TOWARDS GENERALIZED COMBINATORIAL SOLVERS VIA REWARD ADJUSTMENT POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

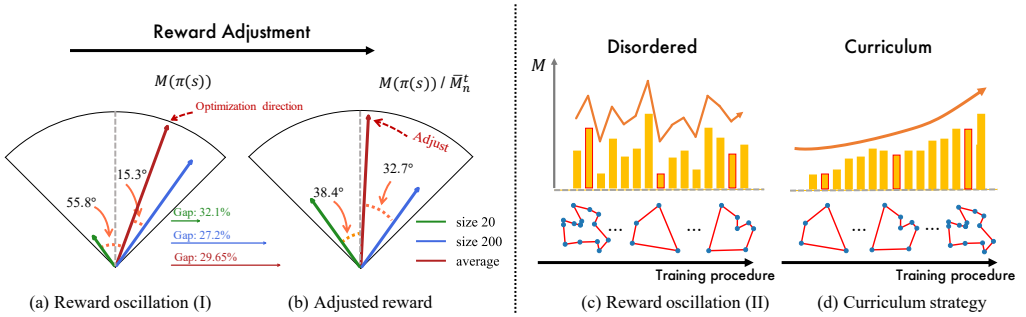
ABSTRACT

Recent reinforcement learning approaches have achieved impressive success in solving combinatorial optimization (CO) problems. However, most existing works focus on evaluating their solvers under a prevalent fixed-size protocol, ignoring generalization to different-size instances. When the solver is confronted with instances of the size it has not been trained on, the performance drops dramatically. In practice, these approaches that lack size-insensitive generalization capacities are unacceptable since an additional training period is repeated for each new instance size. We observe the main obstacle preventing us from training a generalized combinatorial solver is oscillating reward signals. Reward oscillation mainly includes two sides: 1) The conventional reward fails to depict the actual performance of solvers for different instance sizes. 2) The inherent difficulties varying across different sizes worsen training stability. Thus, we present Reward Adjustment Policy Optimization (RAPO), an end-to-end approach to building combinatorial solvers for a wide range of CO problems. RAPO contains a reward adjustment method across instances with variable sizes to address the first side of reward oscillation, along with a promising curriculum strategy to alleviate another side. We conduct experiments on three popular CO problems, namely, the traveling salesman problem (TSP), the capacitated vehicle routing problem (CVRP), and the 0-1 knapsack problem (KP). RAPO exhibits significant improvement in generalization to instances with variable sizes consistently on all benchmarks. Remarkably, RAPO even outperforms its fixed-size counterparts in its well-trained size by a clear margin.

1 INTRODUCTION

Combinatorial optimization (CO) problems are frequently encountered in modern industry. Typical scenarios include traffic optimization (Böther et al., 2021), supply chain optimization (Eskandarpour et al., 2015), and scheduling (Artigues et al., 2013). Many CO problems are NP-hard, which means it is considered unlikely to propose a deterministic algorithm that can solve these problems within polynomial running time. NP-hard CO problems have been studied extensively by the traditional Operations Research (OR) community. Still, real-world CO problems tend to have unique formulations and complicated constraints that vary case-by-case. Developing a powerful and efficient approximate algorithm to solve these problems requires careful hands-on design and extensive experience of domain experts (Affenzeller & Mayrhofer, 2002; Karp, 1972; Helsgaun, 2017), which motivates us to develop a data-driven algorithm that automatically builds combinatorial solvers to save labor costs.

Recently, deep reinforcement learning based combinatorial solvers have made incredible progress. These methods present promising results on various classical NP-hard CO problems such as the traveling salesman problem (TSP) and the capacitated vehicle routing problem (CVRP) at a superior speed (Bello et al., 2016; Kool et al., 2019; Kwon et al., 2020; Joshi et al., 2022). However, most existing works focus on evaluating their solvers under a prevalent fixed-size protocol, ignoring the generalization to different-size instances. This rough evaluation covers up their fragile out-of-distribution performance. Traditional solvers have the inherent capacities to handle combinatorial optimization problems with different input sizes, called size-insensitive generalization. However, it is not the case with existing learning-based solvers. When the solver is confronted with instances of the size it has not been trained on, the performance drops dramatically. Meanwhile, a straightforward



(a) Reward oscillation (I) (b) Adjusted reward (c) Reward oscillation (II) (d) Curriculum strategy

Figure 1: The overview of the reward oscillations. (a) shows the oscillation caused by the conventional reward. The green arrow indicates the optimization direction for size-20 instances under the optimality gap 32.1% which is larger than the optimality gap 27.2% of the blue arrow corresponding to size-200 instances. However, the rewards of these two sizes are -5.02 and -14.00 respectively, leading the final direction of the red arrow substantially tending to the blue one whose optimality gap is smaller. (b) is how the reward adjustment alleviates the oscillation. (c-d) show how a curriculum alleviates the second aspect of the reward oscillations

remedy is to train instances with variable sizes simultaneously, but leading to overall poor performance on each size. In practice, these approaches that lack size-insensitive generalization are unacceptable since an additional training period is repeated for each new instance size. This dilemma of existing neural combinatorial solvers highly motivates us to develop a new approach that allows us to train solvers to handle instances of different sizes efficiently.

We observe the main obstacle preventing us from training a generalized combinatorial solver is oscillating reward signals. Reward oscillation mainly includes two aspects. First, the conventional reward that is directly defined by the negative measure function fails to depict the actual performance of solvers for different instance sizes. As shown in Figure 1(a), the red arrow, denoting optimization direction, tends to the blue arrow with a lower reward but neglecting the green arrow with a larger gap. Second, the inherent difficulties varying across different sizes worsen training stability. The complexities of NP-hard CO problems increase dramatically along with the instance size growing. As shown in Figure 1(c), rapidly switching the instances of different sizes in stochastic training as a straightforward approach makes the optimization struggle.

Thus, we present Reward Adjustment Policy Optimization (RAPO), an end-to-end approach to building combinatorial solvers for a wide range of CO problems. On the one hand, we design a new adjusted reward that can fairly reflect the actual performance of solvers for different sizes and provide a stable signal throughout the training process. On the other hand, we propose a curriculum strategy that gradually increases the scope of variable sizes. Besides, for problems defined in the Euclidean space, we propose a Euclidean Transformer that incorporates Euclidean structural information of the instance into the model, which is shown effective in our experiments with a low cost.

Finally we conduct comprehensive experiments to prove the superiority of RAPO, including three classical combinatorial optimization problems, TSP, CVRP and KP. Remarkably, RAPO outperforms its fixed-size counterparts in their fixed-size specialized paradigm by a clear margin and has additional generalization capabilities for instances with variable sizes at no additional cost. For TSP and CVRP, besides the standard uniformly generated instances, we build a new benchmark based on sampling locations from real USA cities, proving the extensibility of RAPO to complex real-world distributions. These results show that RAPO largely closes the optimality gap in instances with variable sizes and some more cases with larger fixed size and can easily apply to practical applications.

The contributions of this paper are summarized as follows.

- We observe reward oscillations when training a neural combinatorial solver with variable size instances. To address the issues, we propose Reward Adjustment Policy Optimization (RAPO), an end-to-end approach to building a generalized combinatorial solver for a wide range of CO problems. RAPO includes two components, namely, a new adjusted reward based on the reinforcement learning procedure and a curriculum strategy.

- We conduct comprehensive experiments, including three classic combinatorial problems, the traveling salesman problem (TSP), the capacitated vehicle routing problem (CVRP), and the 0-1 knapsack problem (KP). RAPO exhibits significant improvement in generalization to instances with variable sizes consistently on all benchmarks. Remarkably, RAPO even outperforms its fixed-size counterparts in its well-trained size by a clear margin.

2 Related Work

Neural combinatorial optimization. The use of deep learning models to obtain solutions to combinatorial optimization problems has been extensively explored in the last few years. Bello et al. (2016) serve as one of the earliest reinforcement learning approaches based on pointer network framework (Vinyals et al., 2015). Compared with supervised learning, reinforcement learning methods do not require ground-truth optimal solutions as training labels and are more suitable for solving combinatorial optimization problems. Since then, a series of works (Nazari et al., 2018; Kool et al., 2019; Kwon et al., 2020) continues to improve this line, and they build solutions in a greedy style. As a result, these methods are very fast in one run. Our work also belongs to this line. Another path of neural combinatorial solvers is to build solution in an iterative manner. These methods improve solutions based on previous ones continuously (Wu et al., 2021; Chen & Tian, 2019; d O Costa et al., 2020; Hottung & Tierney, 2019). The running time of these local search style methods is related to the predefined computational budget.

Model architecture. Pointer network (Vinyals et al., 2015) is built upon long short-term memory (LSTM) networks (Hochreiter & Schmidhuber, 1997). Nazari et al. (2018) introduce a permutation invariant encoder and then Kool et al. (2019) extend the idea and adopts Transformer models (Vaswani et al., 2017) without positional encodings. Khalil et al. (2017) build their model on Struct2Vec Dai et al. (2016). For those problems with fully connected graphs as input, the permutation or structure of the input nodes does not seem to be important. Even so, our work demonstrates that it is still beneficial to incorporate data structure information into the model as a prior information.

Size generalization of neural combinatorial solvers. Previous literature (Vinyals et al., 2015; Bello et al., 2016; Kool et al., 2019; Kwon et al., 2020) mainly follows a common fixed-size protocol where training and test data are generated by uniformly sampling a fixed number of points from a unit square. Only very few works are on the generalization ability of neural combinatorial solvers. Recent discussions (Joshi et al., 2022; Garmendia et al., 2022; Manchanda et al., 2022) point out that these fixed input-size models fail to generalize to different problem sizes, which limits the applicability of these methods. Zhang et al. (2022) and Jiang et al. (2022) study the mismatch in training and test distributions within a fixed input-size for a learned TSP solver. It is another meaningful angle. Fu et al. (2021) introduces a divide-and-conquer mechanism to generalize a fixed pre-trained neural solver to large-size instances. Wang et al. (2021) and Manchanda et al. (2022) consider another challenging problem in generalizing a model to its unseen distributions. In this paper, we attempt to tackle the size-insensitive generalization problem and unleash the power of learned solvers.

3 Motivation

3.1 Problem Definition

A typical combinatorial optimization instance s includes a group of nodes $\mathcal{V} = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n\}$ as the inputs, where n indicates the problem size. A solution to a CO instance s can be represented as a combination of the input node-set $\mathcal{V}$. Take 2D Euclidean TSP for an example. a valid solution $\tau = (\mathbf{v}_{\tau(1)}, \mathbf{v}_{\tau(2)}, \cdots, \mathbf{v}_{\tau(n)})$ to a TSP instance is a permutation of the node set $\mathcal{V}$, denoting the path traveled through. The measure of the solution can be defined as

$$M(\tau) = ||\mathbf{v}_{\tau(1)} - \mathbf{v}_{\tau(n)}||_2 + \sum_{i=1}^{n-1} ||\mathbf{v}_{\tau(i+1)} - \mathbf{v}_{\tau(i)}||_2. \tag{1}$$

For each instance there is a potential optimal solution τ^* with the minimal measure $M(\tau^*)$. However, no polynomial algorithm to this optimal τ^* is known to date. The object of an approximate solver

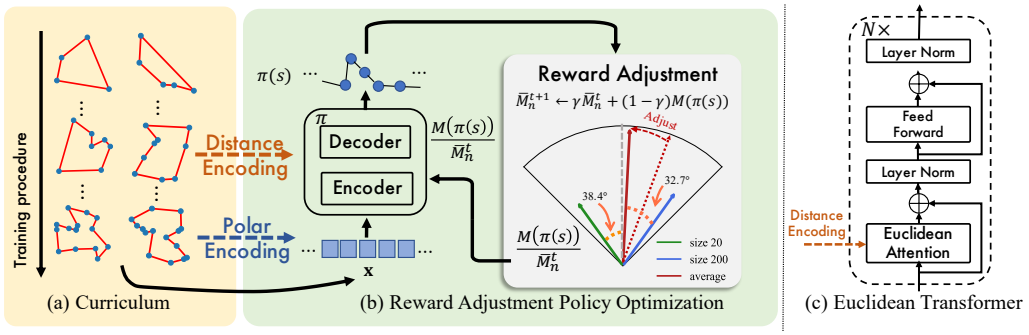


Figure 2: The overview of RAPO. (a) is curriculum strategy; (b) shows how the solver interacts with instances and measure function in RAPO. (c) displays the architecture of Euclidean Transformer.

can be regarded as follows,

$$\min_{\pi} \mathbb{E}_{s\sim D} M(\pi(s)), \tag{2}$$

where the expectation is taken over the problem distribution D. $M(\cdot)$ can be the measure function for arbitrary CO problems (e.g. TSP, CVRP, KP and so on). We denote $\pi(s)$ as a solution τ generated by a solver π for the instance s. Previous learning-based methods (Bello et al., 2016; Kool et al., 2019; Kwon et al., 2020) mainly build upon this optimization objective. They directly set the reward function to $-M(\pi(s))$ by default and learn a solver to minimize the measure function via reinforcement learning.

3.2 Size-insensitive Generalization

Traditional solvers have the inherent capacities to handle combinatorial optimization problems with different input sizes. However, it is not the case with existing learning-based solvers. As shown in Figure 3, solvers trained on instances with a fixed size could hardly generalize to other scales. The prevalent fix-size evaluation protocols (Bello et al., 2016; Kool et al., 2019; Kwon et al., 2020) merely consider training and testing their solvers on some fixed small sizes (e.g. $20, 50, 100$), and thus cover up the fragile out-of-distribution performance of these baseline methods. This phenomenon has also been observed in the literature (Joshi et al., 2022) and proved to be an open problem in this area.

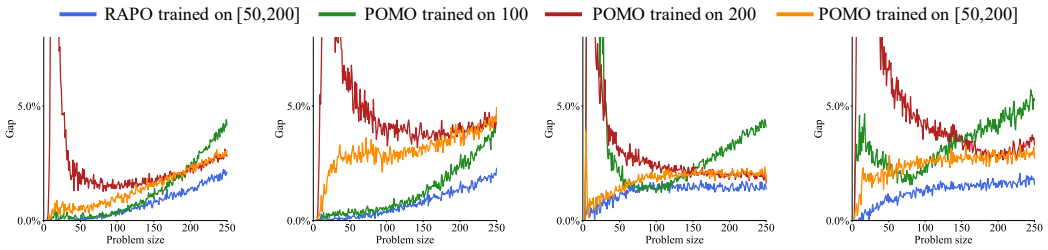
Training on instances with variable sizes simultaneously seems to be a feasible approach to enhance the generalization on variable problem sizes. The modified learning objective can be rewritten as,

$$\min_{\pi} \mathbb{E}_{n\sim U} \mathbb{E}_{s\sim D_n} M(\pi(s)), \tag{3}$$

where the expectation on n denotes that the sampled instance size follows a uniform distribution U over the problem sizes in the target set. However, Figure 3 shows that straightforward optimizing Equation 3 is not a bright way. We can see that even when train-test matching is satisfied, the learned solvers still struggle to achieve the size-insensitive generalization. In practice, these approaches that lack size-insensitive generalization capabilities are unacceptable since an additional training period is repeated for each new instance size. This dilemma of existing neural combinatorial solvers highly motivates us to develop a new approach that allows us to train solvers to handle instances of different sizes efficiently.

4 Method

In the above discussion, we show that the vanilla training with variable sizes simultaneously does not yield the desired size-insensitive generalization capacities. We find that the oscillating reward signals in the training process are the main reason for the failure. As we mentioned above, most previous works adopt the conventional reward function $-M(\pi(s))$ throughout training. However, the scale of this reward would change dramatically according to the size of instance in CO problems. This will cause the reward to fail in revealing the actual performance of solvers for variable-size



(a) Uniform TSP (b) USA TSP (c) Uniform CVRP (d) USA CVRP

Figure 3: The comparison of generalization capacities among RAPO and baselines on various benchmarks. The instances in (a) and (c) are uniformly generated from a unit square $[0, 1]^2$, while the instances in (b) and (d) are uniformly generated from USA cities that present non-trivial topology.

instances. Moreover, the oscillating rewards in the stochastic training will make the optimization struggle. Based on this motivation, we propose Reward Adjustment Policy Optimization (RAPO), a modified learning approach that stabilizes the rewards on variable sizes and eliminates the difficulties of optimization in training.

4.1 Reward Adjustment Policy Optimization

Towards the problem of reward oscillation, the core of our approach is to maintain the reward signals stable across instances of variable sizes. In particular, the optimality gap defined for combinatorial problems is a natural metric that could be applied with instances of variable sizes, which is formulated as

$$G(\pi(s)) = \frac{M(\pi(s)) - M(\tau^*)}{M(\tau^*)}, \tag{4}$$

where the measure function $M(\pi(s))$ is normalized by the optimal measure $M(\tau^*)$. Compared with the naive measure M, the optimality gap G could eliminate the influence of the reward scale and reflect the performance precisely. However, we cannot use the optimality gap as the reward function straightly, since the optimal measure for each instance is not given and is NP-hard to obtain. Hence, we need to find a surrogate to replace $M(\tau^*)$ as the normalization factor.

A lower bound surrogate. Compared with computing $M(\tau^*)$ towards certain instance, estimating the average optimal measure with size n is a more feasible way to reveal the scale. For example, there is a series of theoretical results on the expected $M(\tau^*)$ for TSP problem with uniform node distribution. Suppose $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n$ are n independent random variables uniformly distributed within the square $[0, 1]^2$. The best lower bound to date on the expectation of the $M(\tau^*)$ for this TSP problem given by Steinerberger (2015) has the following form:

$$\mathbb{E}_n[M(\tau^*)] \geq \left(\frac{5}{8} + \frac{19}{5184}\right)\sqrt{n}. \tag{5}$$

This lower bound provides explicit order of $\mathbb{E}_n[M(\tau^*)]$ on size n, which is an ideal surrogate for the intractable $M(\tau^*)$ in the optimality gap $G(\pi(s))$. Besides, there are also many alternatives for TSP problems, e.g., Euclidean minimum spanning tree (Steele & Snyder, 1989). We will discuss these design options in the later section.

However, these lower bounds or numerical estimations are very limited in general use. An elegant lower bound heavily relies on careful theoretical derivation and ideal assumptions, which makes it hard to be adapted to other combinatorial problems and real-world distributions. As a result, we propose a general data-adaptive method to estimate a new surrogate globally in the training procedure.

A momentum updated surrogate. To overcome the disadvantage of the lower bound surrogate, we consider an estimation method that could be embedded into the learning process towards general CO problems or distributions. To match the commonly used stochastic learning setting, the estimation needs to be done within the streaming of sampled mini-batches. Moreover, since the estimated results

are simultaneously used to normalize the reward, we need to keep the numerical stability in the training process. Inspired by the momentum method in the stochastic gradient descent, we propose to apply a momentum updating mechanism to derive the global estimation. Formally, we write the estimated result as vector $\overline{\mathbf{M}}$, where each component $\overline{M}_n$ represents the estimated surrogate for problem size n. We update $\overline{M}_n$ during the training process by

$$\overline{M}_n^{t+1} \leftarrow \gamma \overline{M}_n^t + (1-\gamma) M(\pi_t(s)). \tag{6}$$

Here $\gamma \in [0, 1)$ is a momentum coefficient. With this momentum updating mechanism, the element $\overline{M}_n$ could describe the expected measure of the solver on distribution of instance with size n in training process. Similar with the average optimal measure $\mathbb{E}_n[M(\tau^*)]$, the surrogate $\overline{M}_n$ is also a proper statistic to reveal the global measure scale that could be estimated with higher precision. Furthermore, the slow evolving property of the momentum update also promises the optimization to be as stable as the constant surrogate such as the lower bound surrogate.

After the design of the sophisticated momentum updated surrogate, our training objective function with the policy gradient algorithm (the baseline version) can be formulated as follows:

$$\nabla_\theta J^t(\theta) \approx \mathbb{E}_{n\sim U, s\sim D_n} \left(\frac{M(\tau)}{\overline{M}_n^t} - b(s) \right) \nabla_\theta \log p_\theta(\tau|s), \tag{7}$$

where $\nabla_\theta J^t(\theta)$ is the policy gradient. In the objective function, we replace the conventional reward $M(\tau)$ to the adjusted reward $M(\tau)/\overline{M}_n^t$ where we ignore the constant -1 in the optimality gap. We also minus the shared baseline $b(s)$ following Kwon et al. (2020). The training procedure with this optimization objective is shown in Figure 2(b). With the introduction of the new surrogate, the adjusted reward is balanced and instructive on instances with variable sizes and could depict the actual performance of the solver. We find that compared with the theoretical lower bounds or other numerical estimation methods, the momentum updated surrogate is a more efficient and extensible choice in our empirical study. Table 4 displays the detailed comparison.

Curriculum strategy. While the above adjusted reward addresses the first side of the reward oscillation, there remains the other side that the inherent difficulties varying across different sizes would worsen training stability. Inspired by the observation that the difficulty of NP-hard problems grows dramatically with problem size, we introduce a curriculum strategy as the crucial part of Reward Adjustment Policy Optimization (RAPO). When we sample the size of instances, we make the maximum of the uniform distribution gradually grow in the training process. After the curriculum finishes, the instance size will be sampled uniformly on the target size set. The curriculum strategy is

$$n_t \sim U(\text{low}, \text{up}), \text{ where up} = \begin{cases} \left(\frac{n_{\max} - \text{low}}{t_{\text{cur}}}\right) t + \text{low} & \text{if } t \le t_{\text{cur}}, \\ n_{\max} & \text{otherwise.} \end{cases} \tag{8}$$

Here low can be assigned to an arbitrarily small size (e.g., 20 in TSP), up grows along the training process. $n_{\max}$ is the max problem size we expect to solve. t_{cur} denotes the current training step.

Lisicki et al. (2020) first evaluate the effect of the curriculum strategy paradigms in TSP scenarios. But they merely obtain tiny gains on the performance (1.6% training data reduction with no performance improvement). This is because they only consider one side of reward oscillation and still adopt imbalanced conventional reward throughout training. Considering our superior performance in the experiments, we think that our reward adjustment plays a key role in releasing the potential of the curriculum strategy.

4.2 Euclidean Transformer

Except for the improvement over learning policy, the architecture modification is another way to enhance the learned solver. Towards CO problems, a promising approach is to properly incorporate structural information of the instances into models to boost the learning. As shown in Figure 2(c), we present the Euclidean Transformer, a modified architecture with special structure design, for large group of CO problems that are defined on Euclidean space.

Polar positional encoding. Unlike vision or language tasks, there is no explicit order relation of the input nodes in routing problems. For this reason, the original Transformer for CO problems proposed in (Kool et al., 2019) erases the positional encoding among the input tokens. Since the polar transformation is a fundamental operation in many computational geometry algorithms such as convex hull, we incorporate the polar information as an effective structural prior into the model. In the Euclidean Transformer, we develop a polar positional encoding that is formulated as follows:

$$h_i^{(0)} = x_i + \text{Embed}(\sin\varphi_i, \cos\varphi_i, r_i). \tag{9}$$

Here, φ_i denotes the polar angle of the node and r_i denotes the radius. Embed is a linear layer. In this way, the Euclidean Transformer can utilize the polar information explicitly.

Relative distance encoding. For combinatorial optimization defined in the Euclidean space, the pairwise distances always provide crucial relation information. We present relative distance encoding that explicitly incorporates pairwise distance information into the attention layers of Euclidean Transformer. The following describes how the relative distance encoding influences the attention computation for each block,

$$A_{ij} = \frac{(h_i \cdot W_Q)(h_j \cdot W_K)^T}{\sqrt{d}} + \alpha \text{dist}(i,j) + \beta, \tag{10}$$

where $\text{dist}(i,j)$ denotes the relative distance between nodes i and j, h denotes the hidden state, and W_Q and W_K are weight matrices in the multi-head self-attention mechanism. The additional parameters α and β are learnable and are distinct between every attention head and layer. In the end, a softmax function is performed on A to produce the attention values. With the additional knowledge from the distance, it would be easier for the Euclidean Transformer to capture the topological structure and measure the length of path for easing policy optimization in RAPO.

5 Experiments

We conduct our experiments on three well-known combinatorial optimization problems: travelling salesman problem (TSP), capacitated vehicle routing problem (CVRP) and 0-1 knapsack problem (KP). TSP and CVRP are typical routing problems, which are commonly adopted by previous work. Beyond routing problems, our evaluation on the 0-1 knapsack problem aims to prove that our proposed RAPO is general, and can be easily applied to different kinds of combinatorial optimization problems. Finally, We ablate the important design elements in RAPO.

5.1 Routing Problems: TSP and CVRP

In this paper, we deal with Euclidean TSP and CVRP, where cities or nodes are generated from Euclidean space and the distance between each pair of nodes equals to their Euclidean distance.

Benchmark. We generate instances of routing problems following two different benchmarks: a uniform benchmark and the real-world USA benchmark. For the uniform benchmark, each node of an instance is uniformly sampled from the unit square $[0,1]^2$. Most of the existing works (Bello et al., 2016; Kool et al., 2019; Kwon et al., 2020; Joshi et al., 2022) on TSP and CVRP are evaluated with this benchmark. However, in real-world applications, the node distributions for CO problems are hardly uniform. Therefore, we propose a real-world USA benchmark, where the node is sampled from the distribution of 115475 cities or towns on a real USA map provided by the USA TSP challenge (Cook, 2012). Details about this benchmark can be found in Appendix B.

Evaluation protocol. We evaluate trained models on five different protocols w.r.t. problem sizes: fixed-size protocols $n = 100$, $n = 200$, $n = 500$, as well as varying-size protocols $n \in [50, 250]$ and $n \in [50, 500]$. For the varying-size ones, we sample the size of each problem uniformly from the range $[50, 250]$ and $[50, 500]$, in order to assess the model's ability to generalize to different sizes. For each protocol, we sample 1000 problems with the uniform benchmark or the USA benchmark and report averaged evaluation results. Evaluation results of RAPO reported in fixed-size protocols, namely $n = 100$, $n = 200$ and $n = 500$, are produced by models trained on size intervals $[50, 100]$, $[50, 200]$ and $[50, 500]$, respectively. As for the solution sampling strategy, POMO and RAPO both

Table 1: Experiment results on TSP

	Dataset	$n = 100$		$n = 200$		$n = 500$		$n \in [50, 250]$		$n \in [50, 500]$	
	Metric	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
uniform	Concorde	0.00%	4m	0.00%	22m	0.00%	4h	0.00%	14m	0.00%	3h
	LKH3	0.00%	7m	0.00%	9m	0.21%	17m	0.00%	8m	0.08%	20m
	AM	2.27%	45m	8.16%	2h	12.06%	12h	8.07%	1h	10.78%	8h
	POMO	0.39%	25s	2.75%	1m	7.13%	18m	2.56%	1m	4.10%	12m
	RAPO	**0.30%**	25s	**1.52%**	1m	**5.36%**	18m	**1.05%**	1m	**2.98%**	12m
USA	Concorde	0.00%	5m	0.00%	18m	0.00%	3h	0.00%	9m	0.00%	2h
	LKH3	0.00%	7m	0.00%	9m	0.07%	17m	0.00%	8m	0.02%	22m
	AM	2.14%	45m	8.47%	2h	11.90%	12h	8.23%	1h	10.90%	8h
	POMO	0.53%	25s	3.84%	1m	8.73%	18m	3.28%	1m	5.66%	12m
	RAPO	**0.25%**	25s	**1.45%**	1m	**5.40%**	18m	**0.96%**	1m	**2.86%**	12m

Table 2: Experiment results on CVRP

	Dataset	$n = 100$		$n = 200$		$n = 500$		$n \in [50, 250]$		$n \in [50, 500]$	
	Metric	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
uniform	LKH3	0.00%	13h	0.00%	18h	0.00%	33h	0.00%	17h	0.00%	44h
	AM	4.71%	30m	5.10%	2h	8.13%	16h	5.37%	2h	7.34%	12h
	POMO	**1.40%**	32s	2.07%	2m	3.52%	20m	1.95%	2m	2.60%	14m
	RAPO	1.59%	32s	**1.71%**	2m	**1.81%**	20m	**1.62%**	2m	**1.92%**	14m
USA	LKH3	0.00%	17h	0.00%	26h	0.00%	47h	0.00%	22h	0.00%	69h
	AM	4.94%	30m	4.55%	2h	8.75%	16h	4.96%	2h	7.68%	12h
	POMO	2.34%	32s	2.90%	2m	5.62%	20m	2.75%	2m	3.81%	14m
	RAPO	**1.52%**	32s	**1.74%**	2m	**2.27%**	20m	**1.51%**	2m	**2.04%**	14m

follow the POMO default multiple optima strategy over 8 augmentations for each instance (Kwon et al., 2020) , while AM follows its original 1280-sampling strategy without any augmentation. The *Time* column reported for each protocol is the total time used to solve 1000 problems.

Travelling salesman problem. In Table 1, we compare the performance of RAPO with other baselines on TSP. The best results obtained from the neural solvers are marked in bold. We consider handcrafted solvers Concorde (Applegate et al., 2006) and LKH3 (Helsgaun, 2017), and neural solvers Attention Model (AM) (Kool et al., 2019) and POMO (Kwon et al., 2020). The AM and POMO results for fixed-size protocol are produced by models trained with corresponding fixed size, while results for varying-size protocol are produced by models trained using the objective in Equation 3. RAPO produces high quality solutions under all evaluation protocols on all benchmarks, outperforming POMO by a clear margin especially in varying-size and large-size protocols. In the $n \in [50, 250]$ protocol, RAPO reduces the gap by 60% and 70% on uniform and USA benchmarks respectively, in comparison to its POMO counterpart.

Capacitated vehicle routing problem. For each CVRP instance, we use the solution from the heurstic solver LKH3 as the baseline. The "Gap" values in the table are given relative to LKH3. The results on CVRP are reported in Table 2. RAPO gives superior results over all neural solvers. In large-size protocols, such as $n = 500$, RAPO halves the gap produced by POMO, and can generate solution with a gap only around 2% using only 1% of the time needed for a handcrafted solver. This result shows that RAPO is capable of solving large-scale, complicated combinatorial optimization problems efficiently and can achieve a very good trade-off between solution quality and solving time.

5.2 Beyond Routing Problems: 0-1 Knapsack Problem

We evaluate our model on KP to prove the versatility of our method. The 0-1 knapsack problem can be formulated in a similar manner as TSP, where the x and y coordinates stand for the weight and value of each item. We reuse the model architecture for solving TSP. For each decoding step, the decoder chooses an item to put in the knapsack until all item weights exceed the remaining capacity

Table 3: Experiment results on KP

Dataset	$n = 100$		$n = 200$		$n = 500$		$n \in [50, 250]$		$n \in [50, 500]$	
Metric	Score	Gap	Score	Gap	Score	Gap	Score	Gap	Score	Gap
Oracle	40.343	0.00‰	57.554	0.00‰	91.535	0.00‰	47.982	0.00‰	65.111	0.00‰
AM	40.175	4.22‰	57.113	7.69‰	90.600	10.22‰	47.647	6.93‰	64.546	8.34‰
POMO	40.333	0.25‰	57.535	0.33‰	91.505	0.32‰	47.966	0.33‰	65.089	0.35‰
RAPO	40.334	**0.21‰**	57.538	**0.24‰**	91.513	**0.24‰**	47.969	**0.24‰**	65.095	**0.24‰**

of the knapsack. The comparison between RAPO and baselines in KP is shown in Table 3. The oracle of each problem is obtained via a dynamic programming algorithm. RAPO achieves superior performance over POMO in all evaluation protocols. POMO still suffers from the oscillating reward, as we find the gap of POMO on $n \in [50, 500]$ is mainly caused by weak performance on small instances. RAPO, on the other hand, adjusts the reward and therefore produces better results.

5.3 Model Analysis

In this section, we conduct the ablation study on our model from three key aspects. We choose TSP instances generated from the uniform benchmark as our testbed. The evaluation procedure follows these two protocols $n = 200$ and $n \in [50, 250]$.

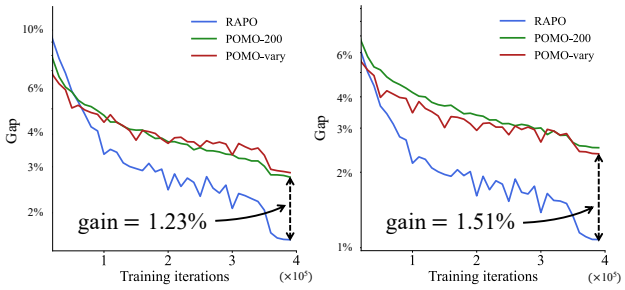
Surrogates for reward adjustment. We compare three design choices of the surrogate discussed in Section 4.1: a numerical solution of lower bound (Steinerberger, 2015), an Euclidean minimum spanning tree (Steele & Snyder, 1989) and our general momentum updated global surrogate. The Euclidean minimum spanning tree (EMST) is an ideal choice for TSP, and the length of the optimal TSP tour can be bounded by EMST, as $|\text{EMST}| \leq |\text{TSP}^*| \leq 2 \times |\text{EMST}|$ (Cormen et al., 2009). We also conduct experiments with the optimality gap as Equation (4), when the optimal measure is computed in advance. Evaluation results are demonstrated in Table 4. The results with original optimality gap and the two surrogates with the lower bound and the EMST supports our motivation of reward adjustment. However, these surrogates are limited and hard to be applied for other CO problems. By contrast, the momentum updated surrogate proposed in this work achieves even better optimality gap in the experiment. These results show that the momentum updated surrogate is a better choice that could also be extensively used on various CO problems.

Curriculum strategy. We present the benefits of the curriculum strategy on both sides, boosting training efficiency and achieving smaller optimality gap. Results in Table 4 support the complementary effect between reward adjustment and curriculum strategy (1.34% to 1.05%). The training curves in Figure 4 highlight that this combination plays an important role in training for solving large TSP instances. Given that Lisicki et al. (2020) observed that curriculum lags the performance in each single size and brings no gains in training efficiency, the importance of our reward adjustment is further emphasized.

Euclidean Transformer. We perform an ablation study on the importance of different module designs in our proposed Euclidean Transformer. The results demonstrate that both polar encoding (PE) and distance encoding (DE) lead to a performance gain compared to the vanilla Transformer and support that a promising structural prior indeed helps to build a strong combinatorial solvers. The detailed results are included in Table 5.

6 Conclusion

This paper studies the open problem that neural combinatorial solvers fail to generalize to instances with variable sizes. We analyze the obstacle that prevents these solvers to tackle different-size CO instances. We provide RAPO, a generalized reinforcement learning based solver enhanced by a reward adjustment mechanism and a curriculum strategy. We have empirically evaluated RAPO with three classic combinatorial optimization problems, namely TSP, CVRP, and KP. We conduct experiments with settings including classic fixed size evaluation but larger problem size, instances with variable sizes, and instances generated from distribution of real USA cities of nonuniform locations. For all settings, RAPO achieves remarkable improvement over existing learned solvers.



(a) $n = 200$ (b) $n \in [50,250]$

Figure 4: Training curves on different models, where RAPO stands for RAPO model trained on $n \in [50, 200]$, POMO-200 for POMO trained on $n = 200$ and POMO-vary for POMO trained on $n \in [50, 200]$.

Table 4: Reward adjustment designs

Designs	$n = 200$	$n \in [50, 250]$
None	2.89%	2.56%
Oracle	2.29%	1.61%
EMST	2.27%	1.61%
lower-bound	2.18%	1.58%
Momentum	1.94%	1.34%
Momentum+Curriculum	**1.52%**	**1.05%**

Table 5: Euclidean Transformer designs

Designs	$n = 200$	$n \in [50, 250]$
None	1.65%	1.15%
PE	1.61%	1.13%
DE	1.57%	1.07%
PE+DE	**1.52%**	**1.05%**

REFERENCES

Michael Affenzeller and Rene Mayrhofer. Generic heuristics for combinatorial optimization problems. In *Proc. of the 9th International Conference on Operational Research*, volume 2002, pp. 83–92, 2002.

David Applegate, Robert Bixby, Vasek Chvatal, and William Cook. Concorde tsp solver, 2006. http://www.math.uwaterloo.ca/tsp/concorde/m.

Christian Artigues, Sophie Demassey, and Emmanuel Neron. *Resource-constrained project scheduling: models, algorithms, extensions and applications*. John Wiley & Sons, 2013.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Maximilian Böther, Leon Schiller, Philipp Fischbeck, Louise Molitor, Martin S Krejca, and Tobias Friedrich. Evolutionary minimization of traffic congestion. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 937–945, 2021.

Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

William Cook. USA TSP challenge. http://www.math.uwaterloo.ca/tsp/data/usa/index.html, 2012.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to algorithms, third edition, 2009.

Paulo R d O Costa, Jason Rhuggenaath, Yingqian Zhang, and Alp Akcay. Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning. In *Asian Conference on Machine Learning*, pp. 465–480. PMLR, 2020.

Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*, pp. 2702–2711. PMLR, 2016.

Majid Eskandarpour, Pierre Dejax, Joe Miemczyk, and Olivier Péton. Sustainable supply chain network design: An optimization-oriented review. *Omega*, 54:11–32, 2015.

Zhang-Hua Fu, Kai-Bin Qiu, and Hongyuan Zha. Generalize a small pre-trained model to arbitrarily large tsp instances. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7474–7482, 2021.

Andoni I Garmendia, Josu Ceberio, and Alexander Mendiburu. Neural combinatorial optimization: a new player in the field. *arXiv preprint arXiv:2205.01356*, 2022.

Keld Helsgaun. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, pp. 24–50, 2017.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

André Hottung and Kevin Tierney. Neural large neighborhood search for the capacitated vehicle routing problem. *arXiv preprint arXiv:1911.09539*, 2019.

Yuan Jiang, Yaoxin Wu, Zhiguang Cao, and Jie Zhang. Learning to solve routing problems via distributionally robust optimization. *arXiv preprint arXiv:2202.07241*, 2022.

Chaitanya K Joshi, Quentin Cappart, Louis-Martin Rousseau, and Thomas Laurent. Learning the travelling salesperson problem requires rethinking generalization. *Constraints*, pp. 1–29, 2022.

Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pp. 85–103. Springer, 1972.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

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

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. POMO: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Michal Lisicki, Arash Afkanpour, and Graham W Taylor. Evaluating curriculum learning strategies in neural combinatorial optimization. *arXiv preprint arXiv:2011.06188*, 2020.

Sahil Manchanda, Sofia Michel, Darko Drakulic, and Jean-Marc Andreoli. On the generalization of neural combinatorial optimization heuristics. *arXiv preprint arXiv:2206.00787*, 2022.

Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems*, 31, 2018.

J Michael Steele and Timothy Law Snyder. Worst-case growth rates of some classical problems of combinatorial optimization. *SIAM Journal on Computing*, 18(2):278–287, 1989.

Stefan Steinerberger. New bounds for the traveling salesman constant. *Advances in Applied Probability*, 47(1):27–36, 2015.

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 (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.

Chenguang Wang, Yaodong Yang, Oliver Slumbers, Congying Han, Tiande Guo, Haifeng Zhang, and Jun Wang. A game-theoretic approach for improving generalization ability of tsp solvers. *arXiv preprint arXiv:2110.15105*, 2021.

Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, and Andrew Lim. Learning improvement heuristics for solving routing problems.. *IEEE transactions on neural networks and learning systems*, 2021.

Zeyang Zhang, Ziwei Zhang, Xin Wang, and Wenwu Zhu. Learning to solve travelling salesman problem with hardness-adaptive curriculum. *arXiv preprint arXiv:2204.03236*, 2022.

A IMPLEMENT DETAILS

Our experiment with RAPO takes POMO as the base model. It is worth mentioning that RAPO is a general method, and we choose to use our method on POMO because it is currently the state-of-the-art solver in end-to-end neural combinatorial solvers for TSP and CVRP. Each of our models is trained for 400K iterations, which takes about 2 days. Adam is used as our optimizer with a learning rate $\eta = 10^{-4}$ and a weight decay $w = 10^{-6}$. We conduct each of our experiments using a single Nvidia TITAN V GPU and a single core of an Intel Xeon Gold 6130 CPU at 2.10GHz. Hyperparameters of the attention model are set the same as Kwon et al. (2020).

A.1 TRAVELLING SALESMAN PROBLEM

Problem setup Given a set of n cities, the goal of Travelling salesman problem (TSP) is to find the shortest possible route that visits a series of city exactly once and returns to the starting city. In our paper, we use 2D Euclidean coordinates (x, y) to represent the location of each city.

Encoder We implement our Euclidean Transformer on top of POMO. The encoder produces a node embedding $\mathbf{h}_i$ for each node input $\mathbf{v}_i$ in a problem instance.

Decoder In the original POMO paper, the decoder uses the average node embedding $\mathbf{h}$, the embedding of the first node $\mathbf{h}^i_{\tau(t-1)}$, and the embedding of the current node $\mathbf{h}^i_{\tau(1)}$ as the query for the decoder. The query, in the form of a concatenation, is defined in Equation (6) and (7) of Kwon et al. (2020).

$$\mathbf{h}^i_{(c)} = \begin{cases} \left[\bar{\mathbf{h}},\ \mathbf{h}^i_{\tau(t-1)},\ \mathbf{h}^i_{\tau(1)}\right] & t > 1 \\ \text{none} & t = 1. \end{cases} \tag{11}$$

$$\mathbf{h}^i_{\tau(1)} = \mathbf{h}_i \qquad \text{for} \quad i = 1, 2, \ldots, N. \tag{12}$$

Here τ is the current trajectory, and $\mathbf{h}^i_{\tau(1)}$ stands for the starting points of multiple trajectories used by POMO.

We adopt a simpler version of the query given by authors of POMO in their source code. This version of the query is the same as the original one, except that the average node embedding is removed.

$$\mathbf{h}^i_{(c)} = \begin{cases} \left[\mathbf{h}^i_{\tau(t-1)},\ \mathbf{h}^i_{\tau(1)}\right] & t > 1 \\ \text{none} & t = 1. \end{cases} \tag{13}$$

Hyperparameters We use the same hyperparameters following previous setups (Kool et al., 2019; Kwon et al., 2020). The node embedding layer maps a node to a $d_{\text{h}} = 128$ dimension embedding. The encoder has $N_{\text{enc}} = 6$ attention layers. The multi-head attention (MHA) module in the attention layer has $M = 8$ heads. Key, value, and query has dimension $d_{\text{k}} = d_{\text{v}} = d_{\text{q}} = 16$. The feed-forward sublayer in each attention layer has hidden dimension $d_{\text{ff}} = 512$. Logit clipping is adpoted in the decoder with the clipping coefficient $C = 10$. The set of hyperparamters is the same for CVRP and KP.

A.2 CAPACITATED VEHICLE ROUTING PROBLEM

Problem setup Capacitated vehicle routing problem (CVRP) is an extension of the TSP that aims to find the optimal route for a vehicle with limited capacity to deliver goods to a given set of customer nodes. The route consists of several sub-routes, at the beginning of which, each vehicle fills its load to its capacity at a given depot and later returns to the depot after a series of delivery. For a CVRP instance with size N, the demand of each customer node δ_i is sampled uniformly from a discrete set $\{1, 2, \cdots, 9\}$ and the capacity of the vehicle is defined as $D = \lceil 30 + \frac{N}{5} \rceil$. In using RAPO to solve the problem, the demand is normalized to $\hat{\delta}_i = \delta_i / D$, and the capacity is normalized to $\hat{D} = 1$.

Encoder An independent encoder is use for the depot node to embed the depot node. This encoder is identical to the encoder for other problem nodes, but the two encoders don't share parameters.

Decoder Similar to TSP, in the original POMO paper, the CVRP decoder uses the average node embedding $\mathbf{h}$, the embedding of the first node $\mathbf{h}^i_{\tau(t-1)}$, and the current load $\hat{D}_t$ as the query for the decoder, formulated below.

$$\mathbf{h}^i_{(c)} = \begin{cases} \left[\bar{\mathbf{h}},\ \mathbf{h}^i_{\tau(t-1)}, \hat{D}_t\right] & t > 1 \\ \text{none} & t = 1. \end{cases} \tag{14}$$

Likewise, we drop the average node embedding term in our implemented query.

$$\mathbf{h}^i_{(c)} = \begin{cases} \left[\mathbf{h}^i_{\tau(t-1)}, \hat{D}_t\right] & t > 1 \\ \text{none} & t = 1. \end{cases} \tag{15}$$

Since the vehicle can return to the depot node at any time to refill the load, different solutions for a certain CVRP instance can differ in lengths. In order to train and evaluate CVRP in batches, we mask all other nodes for trajectories that have finished deliveries so that they can only stay at the depot node and wait for other trajectories to finish. Since actions under this masking has probability 1, these masked steps will not affect the training procedure.

A.3 0-1 Knapsack Problem

Problem setup Given a set of items that each has a weight w and a value v, and a knapsack with capacity D, the goal of the 0-1 Knapsack Problem (KP) is to find the subset of items that has the highest total value while their total weight does not exceed D. Following the setting in POMO, each item $m_i = (w_i, v_i)$ in a KP instance is sampled uniformly from $[0, 1]^2$, and D is set to 25.

Encoder KP can be encoded in the same way as TSP, whereas w and v substitute x and y in the node input. The encoder architecture for KP is exactly the same as for TSP.

Decoder The KP decoder is the same as the TSP decoder except for masking operations. For each decoding step, the already chosen items and items whose weight exceeds the current available capacity are masked. After masking, the decoder chooses an unmasked item to put in the knapsack. Decoding ends when no more items can be chosen. Similar to CVRP, different solutions for the same KP instance may have different lengths, as they choose different number of items. The trajectories that have no available capacity will be forced to choose a dummy node until all trajectories in the batch have finished decoding.

B Details of the USA cities' location dataset

The USA benchmark is motivated from the USA TSP challenge held by University of Waterloo in 2012(Cook, 2012). The organizer of this competition made use of the data collected by the US Geological Survey to propose a 115475-city challenge through nearly all cities, towns, and villages in the contiguous 48 states of USA. The full view of all the points in the dataset is presented in Figure B. We can see from the figure that the points in the dataset are not uniformly distributed.

Since the coordinates take relatively big values (bigger than 10000) representing the latitude and altitude of the cities, we regularized them so that each point falls in the unit square $[0, 1]^2$ for better performance with neural networks. In order to keep the shape of the distribution, the x and y coordinates are divided by the same divisor. In the USA distribution for TSP and CVRP, the points representing cities or customers are sampled at random from this regularized dataset.

C More experiments results

We also proposed a different curriculum than the one we mentioned in 4.1, formulated as below.

$$n_t = \begin{cases} n, \text{where } n \sim U(\text{low}, \left(\frac{n_{\max} - \text{low}}{t_{\text{cur}}}\right) t + \text{low}) & \text{if } t \leq t_{\text{cur}}, \\ n_{\max} & \text{otherwise.} \end{cases} \tag{16}$$

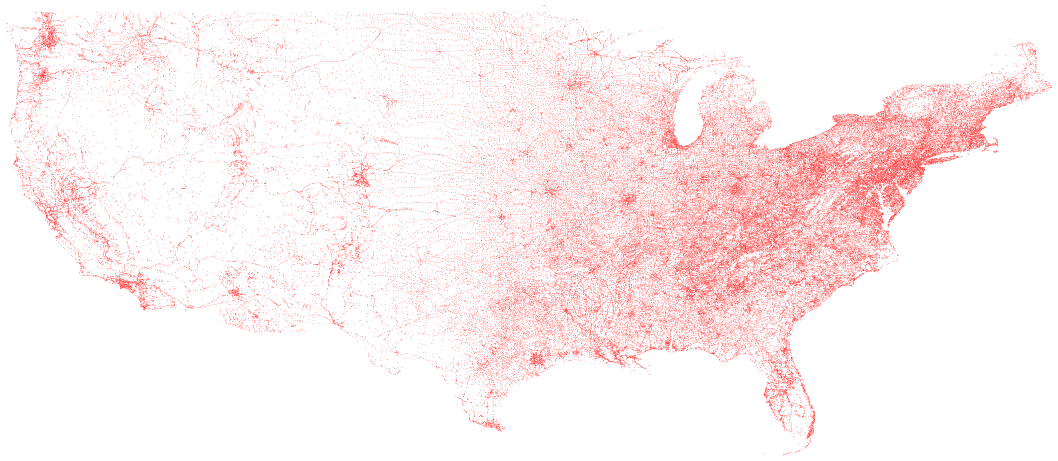


Figure 5: 115475 location points in the USA dataset from Cook (2012)

Table 6: Curriclum experiment results on TSP

	Dataset	$n = 100$		$n = 200$		$n = 500$		$n \in [50, 250]$		$n \in [50, 500]$	
	Metric	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
uniform	Concorde	0.00%	4m	0.00%	22m	0.00%	4h	0.00%	14m	0.00%	3h
	POMO	0.39%	25s	2.75%	1m	7.13%	18m	2.56%	1m	4.10%	12m
	RAPO	0.30%	25s	1.52%	1m	5.36%	18m	**1.05%**	1m	**2.98%**	12m
	RAPO-F	**0.25%**	25s	**1.36%**	1m	**5.24%**	18m	1.10%	1m	3.52%	12m
USA	Concorde	0.00%	5m	0.00%	18m	0.00%	3h	0.00%	9m	0.00%	2h
	POMO	0.53%	25s	3.84%	1m	8.73%	18m	3.28%	1m	5.66%	12m
	RAPO	0.25%	25s	1.45%	1m	**5.40%**	18m	**0.96%**	1m	**2.86%**	12m
	RAPO-F	**0.22%**	25s	**1.34%**	1m	5.53%	18m	1.07%	1m	3.36%	12m

Table 7: Curriclum experiment results on CVRP

	Dataset	$n = 100$		$n = 200$		$n = 500$		$n \in [50, 250]$		$n \in [50, 500]$	
	Metric	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
uniform	LKH3	0.00%	13h	0.00%	18h	0.00%	33h	0.00%	17h	0.00%	44h
	POMO	1.40%	32s	2.07%	2m	3.52%	20m	1.95%	2m	2.60%	14m
	RAPO	1.59%	32s	1.71%	2m	**1.81%**	20m	**1.62%**	2m	**1.92%**	14m
	RAPO-F	**1.43%**	32s	**1.67%**	2m	1.85%	20m	3.15%	2m	2.36%	14m
USA	LKH3	0.00%	17h	0.00%	26h	0.00%	47h	0.00%	22h	0.00%	69h
	POMO	2.34%	32s	2.90%	2m	5.62%	20m	2.75%	2m	3.81%	14m
	RAPO	1.52%	32s	**1.74%**	2m	**2.27%**	20m	**1.51%**	2m	**2.04%**	14m
	RAPO-F	**1.44%**	32s	1.76%	2m	2.51%	20m	3.42%	2m	2.68%	14m

The notations are the same as in Equation 8. The starting size for curriculum low can be assigned an arbitrarily small size, n_t stands for the choice of problem size at training step t, and $n_{\max}$ is the max problem size we expect to solve. The main difference between Equation 8 and Equation 16 is that in this new curriculum setting, the model only trains on the fixed size $n_{\max}$ once training step t reaches t_{cur}. RAPO using this curriculum is denoted as RAPO-F, since the curriculum evolves into fixed-size training.

Experiment results using RAPO-F are presented in Table 6 and Table 7. RAPO-F performs better than RAPO on fixed-size protocols, this is because RAPO-F has more access to the problems with size $n_{\max}$, which matches evaluation. Consequently, RAPO-F has poor performance on varying-size protocols because it gradually forgets small instances during training on $n_{\max}$ size instances.

Table 8: Momentum experiment results

Distribution	uniform TSP		USA TSP		uniform CVRP		USA CVRP	
Protocol	200	$[50, 250]$	200	$[50, 250]$	200	$[50, 250]$	200	$[50, 250]$
$\gamma = 0.85$	2.93%	2.23%	1.63%	1.08%	1.89%	1.77%	1.93%	1.74%
$\gamma = 0.9$	2.20%	1.57%	1.54%	1.03%	1.81%	1.73%	2.00%	1.78%
$\gamma = 0.99$	1.94%	1.34%	1.50%	1.01%	1.90%	1.80%	2.02%	1.77%
$\gamma = 0.999$	2.91%	2.17%	1.85%	1.24%	1.78%	1.69%	1.93%	1.71%

D ABLATION ON THE MOMENTUM COEFFICIENT

The momentum coefficient γ controls the update speed of the momentum updated surrogate, as shown in Equation 6. Table 8 shows the gap of RAPO on $n = 200$ and $n \in [50, 250]$ protocol with different γ values.

For TSP, $\gamma = 0.99$ performs the best. The momentum taking a value too small, such as 0.85, will result in high variance in learning dynamics, causing oscillation. If the momentum takes a relatively large value such as 0.999, the surrogate updates too slowly, causing it to be inaccurate.

For CVRP, where training is more difficult, a larger momentum value $\gamma = 0.999$ performs better, as it can provide a slowly progressing surrogate and make training smoother.

E EFFECT OF INSTANCE AUGMENTATION

Instance augmentation is a common technique for enhancing the model performance. POMO introduced unit square augmentation for Euclidean routing problems in the evaluation phase. The drawback of unit square augmentation is that it only has 8 transformations. As a result, one can only choose from limited 8 trajectories when it comes to sampling from unit square augmentation. We extended the unit square augmentation by adding a rotation of the coordinates of the problem instance around the central point $(0.5, 0.5)$ by angle θ. The transformation is formulated as Equation 17.

$$\begin{cases} x = (x - 0.5)\cos\theta + (y - 0.5)\sin\theta + 0.5 \\ y = -(x - 0.5)\sin\theta + (y - 0.5)\cos\theta + 0.5 \end{cases} \tag{17}$$

This new transformation allows sampling infinitely many augments, as θ can take any value. The rotation augment provides non-linear numerical perturbation, and is therefore highly efficient. We compare evaluation results using 32 rotation augmentations (denoted as $\times 32$) to the 8 unit square augmentations in Table 9 and Table 10. Our new approach takes 4 times more inference time as the unit square augment, and in turn closes the gap by around 10%.

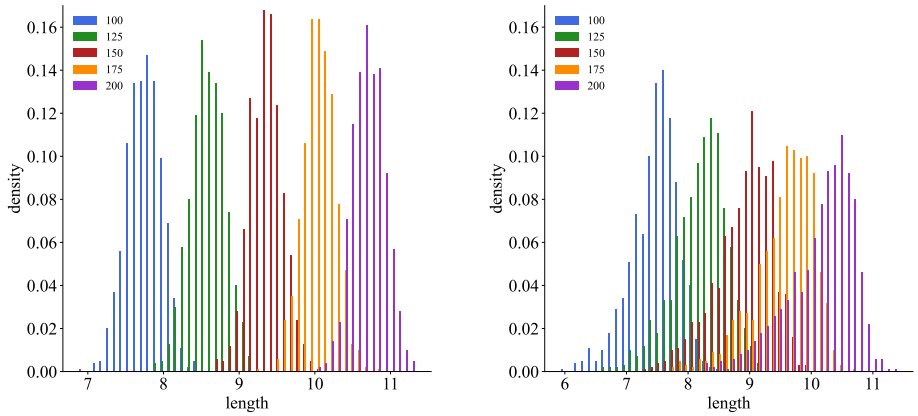
During our experiments, we also find that introducing augmentation in the training phase has a positive effect on evaluation.

Table 9: Augment experiment results on TSP

	Dataset	$n = 100$		$n = 200$		$n = 500$		$n \in [50, 250]$		$n \in [50, 500]$	
	Metric	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
uniform	Concorde	0.00%	4m	0.00%	22m	0.00%	4h	0.00%	14m	0.00%	3h
	RAPO	0.30%	25s	1.52%	1m	5.36%	18m	1.05%	1m	2.98%	12m
	RAPO, ×32	0.23%	3m	1.37%	6m	5.05%	1h	0.91%	4m	2.69%	48m
	RAPO-F	0.25%	25s	1.36%	1m	5.24%	18m	1.10%	1m	3.52%	12m
	RAPO-F, ×32	0.18%	3m	1.24%	6m	4.92%	1h	0.94%	4m	3.13%	48m
USA	Concorde	0.00%	5m	0.00%	18m	0.00%	3h	0.00%	9m	0.00%	2h
	RAPO	0.25%	25s	1.45%	1m	5.40%	18m	0.96%	1m	2.86%	12m
	RAPO, ×32	0.22%	3m	1.37%	6m	5.28%	1h	0.88%	4m	2.73%	48m
	RAPO-F	0.22%	25s	1.34%	1m	5.53%	18m	1.07%	1m	3.36%	12m
	RAPO-F, ×32	0.20%	3m	1.23%	6m	5.38%	1h	0.94%	4m	3.16%	48m

Table 10: Augment experiment results on CVRP

	Dataset	$n = 100$		$n = 200$		$n = 500$		$n \in [50, 250]$		$n \in [50, 500]$	
	Metric	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
uniform	LKH3	0.00%	13h	0.00%	18h	0.00%	33h	0.00%	17h	0.00%	44h
	RAPO	1.59%	32s	1.71%	2m	1.81%	20m	1.62%	2m	1.92%	14m
	RAPO, ×32	1.40%	2m	1.52%	10m	1.73%	1h	1.43%	8m	1.70%	1h
	RAPO-F	1.43%	32s	1.67%	2m	1.85%	20m	3.15%	2m	2.36%	14m
	RAPO-F, ×32	1.32%	2m	1.46%	10m	1.64%	1h	2.69%	8m	2.10%	1h
USA	LKH3	0.00%	17h	0.00%	26h	0.00%	47h	0.00%	22h	0.00%	69h
	RAPO	1.52%	32s	1.74%	2m	2.27%	20m	1.51%	2m	2.04%	14m
	RAPO, ×32	1.36%	2m	1.54%	10m	2.08%	1h	1.38%	8m	1.83%	1h
	RAPO-F	1.44%	32s	1.76%	2m	2.51%	20m	3.42%	2m	2.68%	14m
	RAPO-F, ×32	1.29%	2m	1.56%	10m	2.33%	1h	2.87%	8m	2.37%	1h



(a) Uniform Distribution (b) Explosion Distribution

Figure 6: The oracle length histogram of different sizes for uniform and explosion distribution.