Scaling Combinatorial Optimization Neural Improvement Heuristics with Online Search and Adaptation

Federico Julian Camerota Verdù [1], Lorenzo Castelli[2], Luca Bortolussi[1]

[1]Dipartimento di Matematica, Informatica e Geoscienze, Università degli Studi di Trieste, Italy
[2]Dipartimento di Ingegneria e Architettura, Università degli Studi di Trieste, Italy
federicojulian.camerotaverdu@phd.units.it, lorenzo.castelli@dia.units.it, lbortolussi@units.it

Abstract

We introduce Limited Rollout Beam Search (LRBS), a beam search strategy for deep reinforcement learning (DRL) based combinatorial optimization improvement heuristics. Utilizing pre-trained models on the Euclidean Traveling Salesperson Problem, LRBS significantly enhances both in-distribution performance and generalization to larger problem instances, achieving optimality gaps that outperform existing improvement heuristics and narrowing the gap with state-of-the-art constructive methods. We also extend our analysis to two pickup and delivery TSP variants to validate our results. Finally, we employ our search strategy for offline and online adaptation of the pre-trained improvement policy, leading to improved search performance and surpassing recent adaptive methods for constructive heuristics.

Code — https://github.com/federico-camerota/LRBS

1 Introduction

Combinatorial Optimization (CO) problems can be found in several domains ranging from air traffic scheduling (Bertsimas, Lulli, and Odoni 2011) and supply chain optimization (Singh and Rizwanullah 2022) to circuit board design (Barahona et al. 1988) and phylogenetics (Catanzaro et al. 2012). Although general-purpose solvers exist and most CO problems are easy to formulate, in many applications of interest getting to the exact optimal solution is NP-hard and said solvers are extremely inefficient or even impractical due to the computational time required to reach optimality (Toth 2000; Colorni et al. 1996). Specialized solvers and heuristics have been developed over the years for different applications. However, the latter are often greedy algorithms based on hand-crafted techniques that require vast domain knowledge, thus they cannot be used on different problems and may get stuck on poor local optima (Applegate, Cook, and Rohe 2003; Helsgaun 2009; Gasparin et al. 2023).

CO problems have gained attention in the last few years within the deep learning community where neural networks are used to design heuristics that can overcome the limitations of traditional solvers (Lombardi and Milano 2018; Bengio, Lodi, and Prouvost 2021). In particular, extensive literature has been developed on methods to tackle the travelling salesperson problem (TSP) due to its relevance and particular structure that allows to easily handle constraints with neural heuristics. Deep learning approaches for CO problems can be divided into constructive and improvement methods. The former follows a step-by-step paradigm to generate a solution starting from an empty one and sequentially assigning decision variables (Vinyals, Fortunato, and Jaitly 2015; Nazari et al. 2018; Kool, van Hoof, and Welling 2019). Instead, improvement approaches iteratively improve a given initial solution using an operator to turn a solution into a different one (Zhang et al. 2020; de O. da Costa et al. 2020; Wu et al. 2021; Hottung and Tierney 2022). Moreover, deep learning solvers can be classified based on their learning strategies: supervised learning (Khalil et al. 2017; Joshi, Laurent, and Bresson 2019; Hottung, Bhandari, and Tierney 2020; Li, Yan, and Wu 2021; Xin et al. 2021; Sun and Yang 2023) and deep reinforcement learning (DRL) (Bello et al. 2017; Khalil et al. 2017; Deudon et al. 2018; Kool, van Hoof, and Welling 2019; Ma et al. 2020; Barrett et al. 2020; Kim, Park et al. 2021; Ma et al. 2021; Qiu, Sun, and Yang 2022; Kim, Park, and Park 2022; Ye et al. 2023; Ma, Cao, and Chee 2023).

Many recent advancements in neural solvers for CO primarily lie within the constructive framework. This approach eliminates the necessity for manually crafted components, thereby providing an ideal means to address problems without requiring specific domain knowledge (Lombardi and Milano 2018). However, improvement heuristics can be easier to apply when complex constraints need to be satisfied and may yield better performance than constructive alternatives when the problem structure is difficult to represent (Zhang et al. 2020) or when known improvement operators with good properties exist (Bordewich et al. 2008). Still, generalization, i.e., scaling from training sets with small problems to large instances while retaining good performance, is an open issue when using DRL neural heuristics in CO, particularly for the TSP (Joshi et al. 2021).

Contributions. While generalization has been studied for constructive methods (Hottung, Kwon, and Tierney 2021; Oren et al. 2021; Choo et al. 2022; Son et al. 2023; Jiang et al. 2023; Li et al. 2023), to the best of our knowledge no prior work has been done on improvement heuristics. In

this paper, we focus on improvement heuristics for the TSP based on DRL policies and propose an inference-time beam search approach, Limited Rollout Beam Search (LRBS), which allows tackling problems 10 times larger than those seen at training time. Using pre-trained models, on instances of the same size as those used for training, our search scheme achieves state-of-the-art results among similar improvement methods and shows comparable performance to constructive heuristics. Generalization to instances up to 10 times larger than those seen while training is improved considerably compared to sampling from the original policy, mitigating the gap with constructive solvers. Moreover, our approach allows the integration of online adaptation within the search to overcome the limitations posed by the pre-trained model in large-scale generalization. We also investigate the effectiveness of LRBS as an exploration strategy in fine-tuning the pre-trained models on a limited dataset of instances of the same size as those in the test set. In this setting, our experiments show competitive performances with constructive approaches that use online instance-based adaptation. Finally, we validate LRBS on two pickup and delivery TSP variants and show the advantage of our search approach with respect to more specialized problem-specific solutions. In conclusion, our analysis indicates that solvers utilizing improvement heuristics and a robust exploration approach may offer a viable alternative to adaptive constructive methods, displaying enhanced scalability for larger problem instances in terms of computational times.

2 Preliminary and Related Work

Improving TSP Solutions with DRL

A TSP instance is defined by a graph $G = (V,\ E)$ and the objective is to find a tour δ, i.e. a sequence of nodes $x_i \in V$, such that each node is visited only once, the tour starts and finishes in the same node and minimizes the tour length

$$L(\delta) = w_{\delta_N,\delta_1} + \sum_{i=1}^{N-1} w_{\delta_i,\delta_{i+1}},$$

where $N = |V|$, $w_{ij} \in \mathbb{R}^+$ and $(i,\ j) \in E$ are edges in the graph. In this work, we consider instances of the Euclidean TSP (Arora 1996) where $w_{ij} = \|x_i - x_j\|$.

To solve TSP instances in the improvement framework we start from a given randomly generated initial solution δ^0 and use a policy π_θ, parametrized by learnable weights θ, to sequentially improve δ^0. The policy selects actions in the neighbourhood defined by an operator g that, given a solution and an action, returns another solution to the problem. We formulate the DRL framework as follows.

State. The state is given by the current solution δ^T and the best solution found so far $\delta^{*T} = \text{argmin}_{i \le T} L(\delta^i)$.

Action. Actions are elements within the neighbourhood defined by the operator g. For TSP, we consider 2-opt moves (Lin and Kernighan 1973) that consist of selecting a tuple of indices $(i,\ j)$ and reversing the order of the nodes between δ_i and δ_j. In Figure 1 we illustrate an example of 2-opt move with $(i = 3,\ j = 6)$, assuming zero-based numbering, where the order of all the nodes between $\delta_3 = 2$ and $\delta_6 = 7$ is reversed to obtain the new tour.

Reward. At each step T, the reward is computed as $r^T = L(\delta^{*T}) - L(\delta^{*T-1})$, hence the agent is rewarded only when improving on the best solution found.

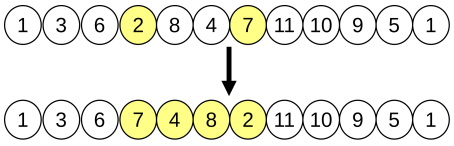

Figure 1: Example of 2-opt move with $(i = 3,\ j = 6)$.

The above elements with the state transitions derived by the operator neighbourhood define a Markov Decision Process (MDP) (Bellman 1957; Puterman 1990) that we call improvement MDP that terminates after T_{max} the number of steps in an episode. Although this work is focused on the Euclidean TSP and its variants, the framework described here easily extends to other routing and CO problems.

Search in Neural CO

Beam Search (BS) and Monte Carlo Tree Search (MCTS) (Coulom 2006) have been widely used in neural CO (Joshi, Laurent, and Bresson 2019; Oren et al. 2021; Choo et al. 2022). Typically, they are used online with autoregressive constructive methods to boost their performance at inference time. However, many of the search techniques that work well for constructive heuristics are difficult to extend efficiently to the improvement setting. This is because constructive methods work on a short horizon, i.e. the number of steps required to obtain a solution, which is defined by the number of variables in the problem. On the contrary, improvement policies often require many more iterations to achieve good performance, see e.g. Ma et al. (2021). Although MCTS has been widely applied with DRL policies, yielding impressive results (Silver et al. 2016), a notable drawback lies in the computational cost associated with its backpropagation procedure (Choo et al. 2022). This limitation renders MCTS less suitable for the context of CO, particularly when dealing with large, difficult-to-explore search spaces. In the literature on neural CO, BS has emerged as a practical alternative to MCTS. This approach strikes a favourable balance between search capability and runtime complexity, making it a promising choice for addressing the challenges inherent in CO scenarios (Vinyals, Fortunato, and Jaitly 2015; Nazari et al. 2018; Kool, van Hoof, and Welling 2019; Joshi, Laurent, and Bresson 2019).

Adaptive Methods for Neural CO In recent developments within the field of neural CO, a novel trend has emerged in search methods that incorporate techniques for online adaptation of policy parameters during inference. This trend finds inspiration in the work of Hottung, Kwon, and Tierney (2021), who introduced Efficient Active Search (EAS) as an enhanced version of Active Search (AS)(Bello et al. 2017). EAS focuses on training only a small subset

of policy weights, significantly reducing its computational footprint. Simulation Guided Beam Search (SGBS) (Choo et al. 2022) employs "simulations" (i.e. policy rollouts) to assess expanded nodes in BS and seamlessly integrates with EAS for online policy adaptation. SGBS's lookahead capability facilitates informed node selection without the complexities associated with intricate backpropagation techniques as in MCTS. However, it's important to note that SGBS samples from the DRL constructive policy until a leaf node is reached for node evaluation, rendering it too computationally intensive for improvement methods. More recently, Son et al. (2023) proposed Meta-SAGE that uses meta-learning and search to scale pre-trained models to large TSP instances. Introducing a bilevel formulation, the algorithm is made of two components: a scheduled adaptation with guided exploration (SAGE) that updates parameters at test time and a scale meta-learner that generates scale-aware context embeddings.

3 Searching with LRBS

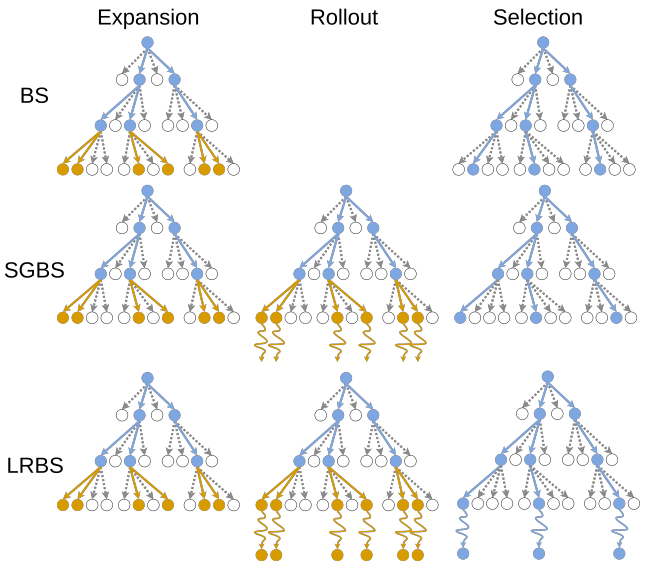

Figure 2: Comparison of BS, SGBS, and LRBS. On the left, is the "Expansion" step which shares similarities among the three algorithms. Highlighted in blue are the β paths of the active beam nodes. Yellow nodes represent the $\beta \times \alpha$ children selected for expansion where SGBS and LRBS apply the DRL policy in the "Rollout". Finally at the "Selection" step the beam is updated and grown down the search tree. While SGBS uses rollouts to evaluate the selected children discarding the trajectory, LRBS keeps the trajectory and selection is done over the states reached in the rollout phase. Illustration inspired by Choo et al. (2022).

In this section, we describe our beam search strategy for CO improvement heuristics. To overcome the limitations of previous methods in the improvement MDP, we propose an effective beam search approach that allows to trade-off between the additional computational cost of search and heuristic performance. Additionally, our approach mitigates the effect of the longer episodic horizon in the improvement MDP by reducing the effective horizon on which the DRL policy works.

Algorithm 1: Limited Rollout Beam Search

Input: initial solution δ^0, pre-trained policy π, parameters $(\alpha, \beta, n_s, T_{\max})$, objective function f
Output: best found tour δ_{best}
$\delta_{\text{best}} \leftarrow \delta^0$
$R \leftarrow \{\}$
$B \leftarrow$ **sample** $\alpha \times \beta$ tours from $\pi(\cdot|\delta^0)$
for δ_i^1 **in** B **do**
  $\delta_i^{n_s} \leftarrow$ rollout π for n_s steps starting at δ_i^1
  add $\delta_i^{n_s}$ **to** R
  update δ_{best}
end for
$B \leftarrow$ select the best β elements in R according to f
$t \leftarrow n_s$
while $t < T_{\max}$ **do**
  $R \leftarrow \{\}$
  $B \leftarrow$ **for each** δ_i^t **in** B **sample** α tours from $\pi(\cdot|\delta_i^t)$
  for δ_i^t **in** B **do**
    $\delta_i^{n_s+t} \leftarrow$ rollout π for n_s steps starting at δ_i^t
    add $\delta_i^{n_s+t}$ **to** R
    update δ_{best}
  end for
  $B \leftarrow$ select the best β elements in R according to f
  $t \leftarrow t + n_s$
end while
return δ_{best}

The LRBS algorithm

Solving a CO problem with the DRL framework in Section 2 can be seen as traversing a search tree using policy π to decide the path to follow. Nodes in the tree represent solutions to the problem, with the initial solution δ^0 being the root node, and edges possible improvement actions (e.g. 2-opt moves) that transform one solution into the other. In Algorithm 1 we present LRBS, the algorithm starts at the root node and carries out its search down the tree in a breath-first fashion by keeping a beam of β active nodes for each depth level and exploring α of their children, thus limiting the branching factor (see Figure 2). Contrary to other search problems, there are no terminal nodes to reach in the improvement MDP. Hence, exploration is carried out until the explored paths in the search tree reach a fixed depth ($T_{\max}$) and the best solution found is returned. While SGBS relies on rollouts to evaluate actions, which is impractical in this context due to the long horizon, LRBS uses limited rollouts, effectively reducing the horizon and enabling exploration. In addition, unlike MCTS, LRBS avoids any backpropagation, making it computationally efficient for the improvement MDP. The two main operations in LRBS can be described as follows.

Expansion and Rollout. LRBS introduces into the standard BS expansion step a limited policy rollout. Specifically, for each active node o_k in the beam, α distinct children are sampled according to the probability distribution of $\pi(\cdot|o_k)$ (depicted in the left column in Figure 2) and then, from the resulting $\beta \times \alpha$ states, the policy is rolled out for n_s steps to obtain solutions o_{k+n_s} (as shown in the middle column of Figure 2). Parameters β and α control the degree of exploration in LRBS, so appropriate values need to balance search performance and runtimes as both would increase with larger β and α. For sensitivity analysis of these parameters, we refer the reader to the appendix in the extended version of this paper (Camerota Verdù, Castelli, and Bortolussi 2024).

Selection. To update its beam, LRBS selects the best β solutions according to the objective function f, e.g. L in the improvement MDP described in Section 2, and then the search continues from the new resulting beam front (right column of Figure 2).

In LRBS, the limited length rollouts have a considerable impact on the search capabilities of the algorithm. Within the improvement MDP, the ability of the DRL agent to explore good solutions is highly constrained to the neighbourhood spanned by the used operator g and the derived available actions. This implies that more than one step may be needed to reach a better solution than the current one, and even worse solutions may be observed in the path to an improved solution. By incorporating n_s steps of policy rollout before selection, instead of a single-step look-ahead as in BS, LRBS harnesses the improvement potential of π and enhances its planning capabilities through exploratory actions facilitated by the beam.

4 Adapting Pre-Trained Policies with LRBS

While the search capabilities of LRBS mitigate the effect of distributional shifts when scaling to larger problem instances than those seen while training, its performance is limited by the pre-trained policy. In this section, we introduce an adaptive framework combining LRBS with EAS to update the pre-trained DRL policy. However, it is important to notice that the framework is general and other approaches could be used for adaptation instead of EAS. We study the effectiveness of this approach in two different scenarios: offline fine-tuning (FT) and online adaptation (OA). In EAS, a small set of new parameters ϕ is introduced by adding a few layers into the agent's neural network, that in encoder-decoder architectures are usually placed in the final layers of the decoder. To reduce the computational burden of previous adaptive methods, Hottung, Kwon, and Tierney (2021) proposed to only train the new weights ϕ, making EAS extremely fast. To update ϕ in constructive heuristics, EAS utilizes a loss function consisting of an RL component, aiming to reduce the cost of generated solutions, and an imitation learning component, which increases the probability of generating the best solution seen so far. However, it is not straightforward to apply EAS in the improvement MDP since running multiple times the improvement heuristic for the total number of steps required to achieve a good solution and then adapting ϕ would incur extremely long computational times. Instead, in LRBS we can incorporate easily EAS by updating the new weights on the limited rollouts used in node expansion. To fine-tune the pre-trained policy, we assume a limited set ($\mathcal{S}_{FT}$) of instances in the target problem distribution is available and train ϕ to maximize the reward achieved over the LRBS rollouts with the RL loss function of EAS, leading to the gradient:

$$\nabla_\phi \mathcal{L}_R(\phi) = \mathbb{E}_\pi[(R(\delta_{\text{LRBS}}) - b)\nabla_\phi \log \pi_\phi(\delta_{\text{LRBS}})] \quad (1)$$

where δ_{LRBS} is a rollout of n_s steps and b is a baseline (as in other works, we use the one proposed in Kwon et al. (2020)). This scenario is representative of many domains where similar CO problems have to be solved several times and past instances can be used for fine-tuning. In our experiments, the instances in $\mathcal{S}_{FT}$ are solved only once by the LRBS algorithm and after each policy rollout the new parameters ϕ are updated according to the gradient in Equation 1. Similarly, in the online adaptation scenario, we update the EAS weights at inference time with the approach described above. However, the EAS parameters are reset before solving each batch of test problems, hence, the extra policy weights adapt solely to the instance being solved.

5 Experimental Results

In this section, we report experimental results on the search capabilities of LRBS and its effect on the generalization of pre-trained DRL agents to large TSP instances and two pickup and delivery variants. We use checkpoints of models from de O. da Costa et al. (2020), pre-trained on Euclidean TSP instances with 100 nodes, and from (Ma et al. 2022), pre-trained on PDTSP and PDTSPL instances with 100 nodes. Ma et al. (2021) recently proposed the Dual-Aspect Collaborative Transformer (DACT) architecture for the improvement of TSP solutions with 2-opt moves. Even though DACT performs better than the model from de O. da Costa et al. (2020) in the authors' study, the latter architecture showed much better scalability in our preliminary investigations and even outperformed DACT when both were coupled with LRBS. In all our experiments on TSP, for LRBS, we set a "search budget" such that $\alpha \times \beta = 60$ and fix the other parameters to $n_s = 20$ and $T_{\max} = 5000$, similarly to previous works and balancing solution performance and runtime. On PDTSP and PDTSPL we reduce the budget to 40 and when doing adaptation we use $n_s = 10$ to lower memory consumption. The best values of α and β for each dataset were determined by testing the method on a set of 10 randomly generated instances of the same size as those in the test set. We run all our experiments using a single NVIDIA Ampere GPU with 64GB of HBM2 memory.

Tests datasets. The TSP instances in our experiments are generated as in Kool, van Hoof, and Welling (2019) where the coordinates of nodes are sampled uniformly at random in the unit square. We consider problems with $N = \{100, 150, 200, 500, 1000\}$ nodes. To ensure a fair comparison with the pre-trained policies, for $N = 100$ we use the same $10,000$ test instances of de O. da Costa et al. (2020). For the other problems, we generate datasets with $1,000$

random instances for $N = \{125, 200\}$ and with 128 instances for $N = 500, 1000$. For PDTSP and PDTSPL experiments we generate sets of 128 random instances with 200 and 500 nodes. In the following, we refer to the test dataset with problems with N nodes as TSPN, PDTSPN and PDTSPLN, respectively.

Baselines. We compare LRBS with the pre-trained policy of de O. da Costa et al. (2020) and DACT, also with 8x of the augmentations introduced in Kwon et al. (2020) (A-DACT). Moreover, we include a modification on SGBS (SGBS+C) with limited rollouts (as in LRBS) to work with improvement heuristics, e.g. the pre-trained DRL policy of de O. da Costa et al. (2020). The search approach is similar to LRBS but the new beam front is selected from the direct children of the previous beam front, based on the information of the limited rollouts. Finally, we report the performance of constructive approaches and related algorithms that use search and adaptation (Kwon et al. 2020; Hottung, Kwon, and Tierney 2021; Choo et al. 2022; Son et al. 2023) as well as more recent methods (Luo et al. 2023; Ye et al. 2024). Although these methods have an advantage over improvement heuristics, direct comparison is not straightforward thus we compare our method only with the former. However, they contextualize our results within the broader literature on constructive methods. Recent methods (Drakulic et al. 2024; Sun and Yang 2023; Luo et al. 2023) achieve better generalization than the considered baseline on TSP. However, such methods typically require specialized policy training and cannot be directly used on any pre-trained policy as we do in this work.

	$N = 100$			$N = 150$		
METHOD	OBJ.	GAP	TIME	OBJ.	GAP	TIME
CONCORDE	7.75	0.0%	-	9.35	0.0%	-
POMO[1]	7.77	0.078%	3H	9.37	0.33%	1H
SGBS[1]	7.76	0.058%	0.2H	9.36	0.22%	0.1H
EAS[1]	7.76	0.044%	15H	9.35	0.12%	10H
SGBS+EAS[1]	7.76	0.024%	15H	**9.35**	**0.08%**	10H
DACT [10K]	7.79	0.463%	1.7H	10.20	9.12%	0.4H
A-DACT [10K]	7.76	0.101%	13H	9.86	5.54%	2.9H
COSTA	7.76	0.065%	19H	9.37	0.25%	3.2H
SGBS+C [5K]	7.78	0.335%	193H	9.44	1.00%	31H
BEAM S. [5K]	7.76	0.015%	19H	9.36	0.18%	6.4H
OURS [5K]	7.76	0.014%	19H	9.36	0.16%	3.2H
OURS+OA [5K]	**7.76**	**0.013%**	22H	9.36	0.13%	3.6H

Table 1: Performance evaluation on TSP100 and TSP150. For improvement methods, numbers in brackets indicate the number of steps.

Boosting In-Distribution Performance

In Table 1 we report the results of LRBS on $TSP100$ and $TSP150$ that are close to the training data distribution. Our method outperforms all the considered baselines on TSP100 and has the best results among improvement heuristics on TSP150. On TSP150 the constructive baselines show slightly better gaps than LRBS, but our approach has considerably lower runtime. From our analysis, on test instances close to the training data distribution the best LRBS configuration is ($\beta = 60,\ \alpha = 1$). While such a configuration corresponds to β parallel runs of the policy, introducing limited rollouts allows us to perform online adaption and achieve improved performance.

Out-of-Distribution Exploration with a Pre-Trained Policy

In the first part of Table 2 we show results on the generalization power of LRBS on TSP problems with 200, 500 and 1000 nodes with LRBS ($\beta,\ \alpha$) configurations (30, 2), (15, 4) and (5, 12), respectively. As the test set distribution shifts away from the training distribution we observe that increasing the number of children evaluated for each node in the beam front improves on generalization. While on smaller instances the policy can select good actions and more exploitation with lower α leads to the best performance, on larger instances increasing α allows to compensate for the imprecision of the agent and yields better results. On these test datasets, LRBS scales better than other improvement heuristics achieving optimality gaps close to those of constructive approaches. Our experiments show that the augmentations employed by Ma et al. (2021) considerably improve the policy performance on instances with the same size as the training set. However, when considering larger graphs the benefit of the augmentations becomes less pronounced and the algorithm fails to scale. On the contrary, online exploration with LRBS mitigates the performance degeneration due to distributional shift and our method even improves on the results that the policy of de O. da Costa et al. (2020) would achieve if exploring the solution space for the same time as LRBS and using on average 12x more 2-opt operations. On larger instances, LRBS is not competitive with Meta-SAGE but achieves optimality gaps comparable to those of EAS and SGBS+EAS, even improving its performance as the instances get larger. Turning our attention to the comparison of LRBS to BS, the results in Table 2 present an interesting phenomenon. On the smaller instances with up to 500 nodes, LRBS is faster and achieves much lower optimality gaps, even 6x smaller than BS. However, on the largest problems of the TSP1000 dataset, BS performs better than LRBS. This result strongly suggests that as the distributional shift between the training and test instances gets very large the step-wise greedy selection process of BS is better than the rollouts of LRBS in limiting the performance degradation of the policy. This further motivates the need for adaptive strategies to overcome the limitations posed by the pre-trained model.

Generalization via Adaptation

In the second part of Table 2, we show results on the generalization of LRBS after fine-tuning the DRL policy on a small set of randomly generated instances with the same number of nodes as the test set (FT) and when adapting the policy parameters online (OA). The LRBS configurations are the

[1]Results from Choo et al. (2022)

METHOD	TSP200 OBJ.	TSP200 GAP	TSP200 TIME	TSP500 OBJ.	TSP500 GAP	TSP500 TIME	TSP1000 OBJ.	TSP1000 GAP	TSP1000 TIME
CONCORDE	10.704	0.0%	-	16.530	0.0%	-	23.144	0.0%	-
GLOP[2]	–	–	–	16.91	1.99%	0.1H	23.84	3.11%	0.1H
LEHD[3]	–	0.0182%	0.2H	–	0.167%	1.2H	–	0.719%	7H
EAS[4]	10.736	0.455%	2.4H	18.135	9.362%	4.3H	30.744	32.869%	20H
SGBS+EAS[4]	10.734	0.436%	2.1H	18.191	9.963%	4.2H	28.413	22.795%	19H
META-SAGE[4]	**10.729**	**0.391%**	2.1H	**17.131**	**3.559%**	3.8H	25.924	12.038%	18H
DACT [10K]	15.450	34.346%	0.6H	154.339	833%	0.4H	421.76	1722%	1.8H
A-DACT [10K]	14,345	34.023%	4.4H	147.127	790%	3.3H	412,787	1683%	11.9H
COSTA	10.789	0.796%	4.1H	17.971	8.717%	1.9H	30.439	31.526%	7.8H
SGBS+C [5K]	10.903	1.858%	21.2H	18.455	11.642%	19.1H	47.083	103.44	78.6H
BEAM S. [5K]	11.137	4.051%	6.4H	17.851	7.993%	2.5H	26.507	14.536%	8.7H
OURS [2K]	10.782	0.738%	1.6H	17.684	6.902%	0.8H	32.368	39.970%	3.1H
OURS [5K]	10.771	0.633%	4.1H	17.309	4.633%	1.9H	27.922	20.740%	7.8H
OURS+OA [2K]	10.771	0.629%	1.9H	17.443	5.523%	0.9H	28.468	23.008%	3.3H
OURS+OA [5K]	10.760	0.528%	4.8H	17.187	3.973%	2.0H	**25.895**	**11.889%**	8.1H
OURS+FT [2K]	10.768	0.599%	2.6H	17.303	4.680%	0.9H	28.891	24.838%	3.9H
OURS+FT [5K]	10.757	0.504%	4.6H	17.102	3.463%	2.0H	25.801	11.483%	8.6H

Table 2: Performance evaluation on TSP200, TSP500 and TSP1000. For improvement methods, numbers in brackets indicate the number of steps.

same used for the non-adaptive experiments, with the only exception of the LRBS + FT on the TSP1000 where we use ($\beta = 10,\ \alpha = 6$). For all the considered problems, the FT dataset of randomly generated instances is of size equal to 10% of the test set size and each instance is solved only once using LRBS, running times include also the fine-tuning phase. While LRBS+FT shows the best results for TSP500 and TSP1000, we do not highlight them in bold to keep a fair comparison with the baselines. Our results show that fine-tuning on a limited set of problems allows LRBS to improve considerably on larger instances surpassing all the baselines on the TSP500 and TSP1000 benchmarks, while on the TSP200 the performance of LRBS is close to that of EAS. Even though online adaptation is less effective than fine-tuning, since policy weights are trained only on the instance being solved and reset thereafter, it achieves competitive results on the TSP200 and TSP500 datasets while outperforming constructive baselines on the TSP1000 instances. Although FT achieves lower gaps, we highlight these results in Table 2 for a fair comparison with the baselines. These results show that introducing an adaptive component in the search process of LRBS can overcome the limitations posed by the adopted pre-trained policy. In particular, on the larger TSP1000 problems, the use of LRBS alone fails to achieve the performance of the constructive baselines while both the offline and online adaptive approaches we propose almost halve the optimality gap of LRBS alone and even improve on the baselines.

[2]Results from Ye et al. (2024)

[3]Results from Luo et al. (2023)

[4]Results from Son et al. (2023)

	TSP150	TSP200	TSP500	TSP1000
OURS+FT	0.14%	0.50%	3.46%	11.48%
W/O LRBS FT	0.17%	0.52%	8.43%	81.63%
W/O EXP.	0.28%	1.44%	7.57%	16.06%

Table 3: Optimality gaps on TSP150, TSP200, TSP500 and TSP1000 datasets.

Ablation study. Table 3 reports the analysis of the importance of each element in the fine-tuning experiments on LRBS. In the first case (w/o LRBS FT in Table 3), the training framework of de O. da Costa et al. (2020) is used to fine-tune the policy while in the latter (w/o Exp. in Table 3) we sample from the policy for the same time as the LRBS runtime. To provide a fair comparison, when fine-tuning is done without LRBS the training runs for $n_s \times \beta$ steps to train on the same number of environment interactions. While on the TSP150 and TSP200 datasets w/o LRBS FT yields a gap close to that with LRBS, on TSP500 and TSP1000 there is a considerable performance degeneration. On the contrary, when online exploration is replaced by sampling there is a much smaller effect on the generalization abilities of the model in the larger datasets but a greater decrease in performance in the TSP150 and TSP200 datasets. This shows the strength of LRBS in the fine-tuning phase where exploration allows the policy to better adapt to larger instances, especially for larger instances where the policy can easily get stuck in local optima. Moreover, for smaller problems, we observe that exploration in the fine-tuning phase is less critical but it has a considerable impact when applied online.

	PDTSP200			PDTSP500			PDTSPL200			PDTSPL500		
METHOD	OBJ.	GAP	TIME	OBJ.	GAP	TIME	OBJ.	GAP	TIME	OBJ.	GAP	TIME
LKH	12.913	0.0%	3.4H	20.332	0.0%	20.8H	29.322	0.0%	2.7H	69.922	0.0%	23.1H
N2S-A [1K]	15.321	18.655%	2.7H	114.789	464.668%	34.6H	31.060	5.937%	3.4H	185.862	165.900%	51.0H
N2S-A [2K]	14.875	15.198%	5.5H	95.513	369.723%	69.3H	30.278	3.272%	6.8H	176.543	152.583%	102.3H
N2S-A [3K]	14.716	13.960%	8.3H	89.731	341.330%	103.9H	30.028	2.417%	10.1H	173.354	148.017%	153.5H
OURS [1K]	14.710	13.918%	1.0H	48.907	140.604%	5.8H	30.228	3.103%	1.4H	142.341	103.669%	8.8H
OURS [2K]	14.433	11.773%	2.0H	35.891	76.558%	11.6H	29.869	1.874%	2.8H	108.413	55.087%	17.6H
OURS [3K]	14.320	10.899%	3.0H	33.723	65.889%	17.3H	29.721	1.370%	4.2H	85.762	22.663%	26.5H
OURS+OA [1K]	14.456	11.951%	1.9H	39.885	96.201%	11.8H	30.164	2.884%	2.8H	108.849	55.726%	21.6H
OURS+OA [2K]	14.161	9.664%	3.7H	33.587	65.227%	23.6H	29.839	1.774%	5.6H	81.044	15.942%	43.0H
OURS+OA [3K]	**13.987**	**8.321%**	5.5H	**31.820**	**56.532%**	35.4H	**29.716**	**1.358%**	8.5H	**77.012**	**10.161%**	64.2H

Table 4: Performance evaluation on PDTSP200, PDTSP500, PDTSPL200 and PDTSPL500. Numbers in brackets indicate the number of steps.

Computational efficiency From Table 2 we can also observe that LRBS and its online adaptive variant not only present a lower runtime than the constructive methods but also scale better, i.e. the relative increase in runtime as the test problems get larger is smaller for our algorithms. The reason for this fact is the autoregressive nature of constructive heuristics. With improvement approaches, we keep a fixed number of steps hence the computational cost grows only due to the larger problems to be processed by the neural policy. However, constructive methods, by design, need to perform increasingly more steps to generate solutions for larger instances, thus incurring an additional computational burden as the size of the problems increases

Pickup and Delivery Problems

The pickup and delivery variant of TSP (PDTSP) consists of n one-to-one pickup-delivery requests, where goods at n pickup nodes need to be transported to n corresponding delivery nodes. The objective is to find the shortest Hamiltonian cycle under the *precedence* constraint that every pickup node has to be visited before its corresponding delivery node. We also study PDTSP with the *last-in-first-out* constraint (PDTSPL) that enforces a stack ordering between collected goods and delivery is allowed only for the good at the top of the stack. For these problems, Ma et al. (2022) define a *removal-reinsertion* operator that selects a pickup-delivery request nodes $(\delta_{i^+},\ \delta_{i^-})$, positions $(j,\ k)$ and places node δ_{i^+} after node δ_j and node δ_{i^-} after δ_k. In applying LRBS, we use the same framework described for the TSP but perform the expansion phase only on removal actions. The additional constraints are addressed at the policy and environment level, making LRBS versatile. In Table 4 we report the results of applying LRBS on model checkpoints from Ma et al. (2022) (N2S-A), pre-trained on pickup and delivery instances of size 100, when solving PDTSP and PDTSPL instances with $N = 200$ and 500 nodes. In these experiments, for N2S-A we use the same exploration strategy adopted by the authors where at inference time each instance solved is transformed into $\frac{1}{2}|N|$ different ones, using the augmentations of (Kwon et al. 2020), and the policy is rolled out from each new instance. Our results show that the online exploration approach of LRBS is much more effective than N2S-A when generalizing to larger instances. Not only in terms of pure performance but also computational efficiency. On the smaller instances with 200 nodes, LRBS achieves a good reduction of optimality gaps requiring less time than N2S-A even when performing online adaptation. The PDTSP500 benchmark results are not satisfactory with optimality gaps well above 50% but still, LRBS shows improved generalization compared to N2S-A reducing its gap by almost $6x$. On the much more constrained PDTSPL500 problems instead, online search through LRBS outperforms N2S-A with a gap reduction close to $10x$ when adaption is employed. Overall, the results of Table 4 are still far from being competitive with traditional solvers such as LKH but show the generalization potential of pre-trained policy with online search and adaptation.

6 Conclusion

In this study, we have introduced LRBS, a novel beam search method designed to complement DRL-based improvement heuristics for combinatorial optimization problems enhancing inference time performance and generalization. LRBS offers a tailored approach that enables pre-trained models to efficiently handle problem instances of significantly larger scales, up to ten times bigger than those encountered during the DRL policy initial training phase. To further enhance the generalization of pre-trained models, we integrate LRBS with EAS in offline and online adaptive scenarios. Our experimental evaluation shows that LRBS is superior to existing DRL improvement methods in solving the Euclidean TSP and two pickup and delivery variants. LRBS consistently outperforms alternative approaches proposed both for constructive and improvement heuristics. Moreover, in our analysis, LRBS exhibits superior runtime efficiency when scaling to larger instances compared to established constructive baselines, showing how improvement heuristics coupled with adaptive and search approaches can be a viable alternative to constructive methods.

Acknowledgments

This work has been partially supported by the PNRR project iNEST (Interconnected North-Est Innovation Ecosystem) funded by the European Union Next-GenerationEU (Piano Nazionale di Ripresa e Resilienza (PNRR) – Missione 4 Componente 2, Investimento 1.5 – D.D. 1058 23/06/2022, ECS_00000043).

References

Applegate, D.; Cook, W.; and Rohe, A. 2003. Chained Lin-Kernighan for large traveling salesman problems. *Informs journal on computing*, 15(1): 82–92.

Arora, S. 1996. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In *Proceedings of 37th Conference on Foundations of Computer Science*, 2–11. IEEE.

Barahona, F.; Grötschel, M.; Jünger, M.; and Reinelt, G. 1988. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36(3): 493–513.

Barrett, T.; Clements, W.; Foerster, J.; and Lvovsky, A. 2020. Exploratory combinatorial optimization with reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 3243–3250.

Bellman, R. 1957. A Markovian decision process. *Journal of mathematics and mechanics*, 679–684.

Bello, I.; Pham, H.; Le, Q. V.; Norouzi, M.; and Bengio, S. 2017. Neural Combinatorial Optimization with Reinforcement Learning.

Bengio, Y.; Lodi, A.; and Prouvost, A. 2021. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2): 405–421.

Bertsimas, D.; Lulli, G.; and Odoni, A. 2011. An integer optimization approach to large-scale air traffic flow management. *Operations research*, 59(1): 211–227.

Bordewich, M.; Gascuel, O.; Huber, K. T.; and Moulton, V. 2008. Consistency of topological moves based on the balanced minimum evolution principle of phylogenetic inference. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6(1): 110–117.

Camerota Verdù, F. J.; Castelli, L.; and Bortolussi, L. 2024. Scaling Combinatorial Optimization Neural Improvement Heuristics with Online Search and Adaptation. arXiv:2412.10163.

Catanzaro, D.; Labbé, M.; Pesenti, R.; and Salazar-González, J.-J. 2012. The balanced minimum evolution problem. *INFORMS Journal on Computing*, 24(2): 276–294.

Choo, J.; Kwon, Y.-D.; Kim, J.; Jae, J.; Hottung, A.; Tierney, K.; and Gwon, Y. 2022. Simulation-guided beam search for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 35: 8760–8772.

Colorni, A.; Dorigo, M.; Maffioli, F.; Maniezzo, V.; Righini, G.; and Trubian, M. 1996. Heuristics from nature for hard combinatorial optimization problems. *International Transactions in Operational Research*, 3(1): 1–21.

Coulom, R. 2006. Efficient selectivity and backup operators in Monte-Carlo tree search. In *International conference on computers and games*, 72–83. Springer.

de O. da Costa, P. R.; Rhuggenaath, J.; Zhang, Y.; and Akcay, A. 2020. Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning. In *Asian conference on machine learning*, 465–480. PMLR.

Deudon, M.; Cournut, P.; Lacoste, A.; Adulyasak, Y.; and Rousseau, L.-M. 2018. Learning heuristics for the tsp by policy gradient. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 15th International Conference, CPAIOR 2018, Delft, The Netherlands, June 26–29, 2018, Proceedings 15*, 170–181. Springer.

Drakulic, D.; Michel, S.; Mai, F.; Sors, A.; and Andreoli, J.-M. 2024. Bq-nco: Bisimulation quotienting for efficient neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 36.

Gasparin, A.; Camerota Verdù, F. J.; Catanzaro, D.; and Castelli, L. 2023. An evolution strategy approach for the balanced minimum evolution problem. *Bioinformatics*, 39(11): btad660.

Helsgaun, K. 2009. General k-opt submoves for the Lin–Kernighan TSP heuristic. *Mathematical Programming Computation*, 1: 119–163.

Hottung, A.; Bhandari, B.; and Tierney, K. 2020. Learning a latent search space for routing problems using variational autoencoders. In *International Conference on Learning Representations*.

Hottung, A.; Kwon, Y.-D.; and Tierney, K. 2021. Efficient Active Search for Combinatorial Optimization Problems. In *International Conference on Learning Representations*.

Hottung, A.; and Tierney, K. 2022. Neural large neighborhood search for routing problems. *Artificial Intelligence*, 313: 103786.

Jiang, Y.; Cao, Z.; Wu, Y.; Song, W.; and Zhang, J. 2023. Ensemble-based Deep Reinforcement Learning for Vehicle Routing Problems under Distribution Shift. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Joshi, C. K.; Cappart, Q.; Rousseau, L.-M.; and Laurent, T. 2021. Learning TSP Requires Rethinking Generalization. In *International Conference on Principles and Practice of Constraint Programming*.

Joshi, C. K.; Laurent, T.; and Bresson, X. 2019. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*.

Khalil, E.; Dai, H.; Zhang, Y.; Dilkina, B.; and Song, L. 2017. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30.

Kim, M.; Park, J.; and Park, J. 2022. Sym-nco: Leveraging symmetricity for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 35: 1936–1949.

Kim, M.; Park, J.; et al. 2021. Learning collaborative policies to solve NP-hard routing problems. *Advances in Neural Information Processing Systems*, 34: 10418–10430.

Kool, W.; van Hoof, H.; and Welling, M. 2019. Attention, Learn to Solve Routing Problems! In *International Conference on Learning Representations*.

Kwon, Y.-D.; Choo, J.; Kim, B.; Yoon, I.; Gwon, Y.; and Min, S. 2020. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 21188–21198.

Li, S.; Yan, Z.; and Wu, C. 2021. Learning to delegate for large-scale vehicle routing. *Advances in Neural Information Processing Systems*, 34: 26198–26211.

Li, Y.; Guo, J.; Wang, R.; and Yan, J. 2023. From distribution learning in training to gradient search in testing for combinatorial optimization. *Advances in Neural Information Processing Systems*.

Lin, S.; and Kernighan, B. W. 1973. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2): 498–516.

Lombardi, M.; and Milano, M. 2018. Boosting combinatorial problem modeling with machine learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 5472–5478.

Luo, F.; Lin, X.; Liu, F.; Zhang, Q.; and Wang, Z. 2023. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. *Advances in Neural Information Processing Systems*, 36: 8845–8864.

Ma, Q.; Ge, S.; He, D.; Thaker, D.; and Drori, I. 2020. Combinatorial Optimization by Graph Pointer Networks and Hierarchical Reinforcement Learning. In *AAAI Workshop on Deep Learning on Graphs: Methodologies and Applications*.

Ma, Y.; Cao, Z.; and Chee, Y. M. 2023. Learning to Search Feasible and Infeasible Regions of Routing Problems with Flexible Neural k-Opt. In *Advances in Neural Information Processing Systems*, volume 36.

Ma, Y.; Li, J.; Cao, Z.; Song, W.; Guo, H.; Gong, Y.; and Chee, Y. M. 2022. Efficient Neural Neighborhood Search for Pickup and Delivery Problems. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 4776–4784.

Ma, Y.; Li, J.; Cao, Z.; Song, W.; Zhang, L.; Chen, Z.; and Tang, J. 2021. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. *Advances in Neural Information Processing Systems*, 34: 11096–11107.

Nazari, M.; Oroojlooy, A.; Snyder, L.; and Takác, M. 2018. Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems*, 31.

Oren, J.; Ross, C.; Lefarov, M.; Richter, F.; Taitler, A.; Feldman, Z.; Di Castro, D.; and Daniel, C. 2021. SOLO: search online, learn offline for combinatorial optimization problems. In *Proceedings of the International Symposium on Combinatorial Search*, volume 12, 97–105.

Puterman, M. L. 1990. Markov decision processes. *Handbooks in operations research and management science*, 2: 331–434.

Qiu, R.; Sun, Z.; and Yang, Y. 2022. Dimes: A differentiable meta solver for combinatorial optimization problems. *Advances in Neural Information Processing Systems*, 35: 25531–25546.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587): 484–489.

Singh, G.; and Rizwanullah, M. 2022. Combinatorial optimization of supply chain Networks: A retrospective & literature review. *Materials Today: Proceedings*, 62: 1636–1642.

Son, J.; Kim, M.; Kim, H.; and Park, J. 2023. Meta-SAGE: scale meta-learning scheduled adaptation with guided exploration for mitigating scale shift on combinatorial optimization. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Sun, Z.; and Yang, Y. 2023. DIFUSCO: Graph-based Diffusion Solvers for Combinatorial Optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Toth, P. 2000. Optimization engineering techniques for the exact solution of NP-hard combinatorial optimization problems. *European Journal of Operational Research*, 125(2): 222–238.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. *Advances in neural information processing systems*, 28.

Wu, Y.; Song, W.; Cao, Z.; Zhang, J.; and Lim, A. 2021. Learning improvement heuristics for solving routing problems. *IEEE transactions on neural networks and learning systems*, 33(9): 5057–5069.

Xin, L.; Song, W.; Cao, Z.; and Zhang, J. 2021. NeuroLKH: Combining deep learning model with Lin-Kernighan-Helsgaun heuristic for solving the traveling salesman problem. *Advances in Neural Information Processing Systems*, 34: 7472–7483.

Ye, H.; Wang, J.; Cao, Z.; Liang, H.; and Li, Y. 2023. DeepACO: Neural-enhanced Ant Systems for Combinatorial Optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Ye, H.; Wang, J.; Liang, H.; Cao, Z.; Li, Y.; and Li, F. 2024. GLOP: Learning Global Partition and Local Construction for Solving Large-scale Routing Problems in Real-time. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zhang, C.; Song, W.; Cao, Z.; Zhang, J.; Tan, P. S.; and Chi, X. 2020. Learning to dispatch for job shop scheduling via deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1621–1632.