Proceedings of the Thirty-Third International Conference on Automated Planning and Scheduling (ICAPS 2023)

Imitation Improvement Learning for Large-Scale Capacitated Vehicle Routing Problems

Viet Bui, Tien Mai

School of Computing and Information Systems, Singapore Management University
tvbui@smu.edu.sg, atmai@smu.edu.sg

Abstract

Recent works using deep reinforcement learning (RL) to solve routing problems such as the capacitated vehicle routing problem (CVRP) have focused on improvement learning-based methods, which involve improving a given solution until it becomes near-optimal. Although adequate solutions can be achieved for small problem instances, their efficiency degrades for large-scale ones. In this work, we propose a new improvement learning-based framework based on *imitation learning* where classical heuristics serve as experts to encourage the policy model to mimic and produce similar or better solutions. Moreover, to improve scalability, we propose *Clockwise Clustering*, a novel augmented framework for decomposing large-scale CVRP into subproblems by clustering sequentially nodes in clockwise order, and then learning to solve them simultaneously. Our approaches enhance state-of-the-art CVRP solvers while attaining competitive solution quality on several well-known datasets, including real-world instances with sizes up to 30,000 nodes. Our best methods are able to achieve new *state-of-the-art* results for several large instances and generalize to a wide range of CVRP variants and solvers. We also contribute new datasets and results to test the generalizability of our deep RL algorithms.

Introduction

We study the vehicle routing problems (VRP), an important class of combinatorial optimization problems which has a wide range of applications in logistics (Laporte 2009). Capacitated vehicle routing problem (CVRP) is a basic variant of VRP, aiming to find a set of routes that minimize the cost and fulfill the demands of a set of customers without violating vehicle capacity constraints. The CVRP is NP-hard (Lenstra and Kan 1981). Both exact and heuristic methods have been developed to solve it (Toth et al. 2014). In recent years, especially after the seminal work of Pointer Networks (Vinyals, Fortunato, and Jaitly 2015) and Graph Neural Networks (Prates et al. 2019), researchers have started to develop new deep learning and reinforcement learning (RL) frameworks to solve combinatorial optimization problems (Kool, van Hoof, and Welling 2018; Chen and Tian 2019). The idea behind the RL algorithms is that a machine learning method could learn better heuristics by extracting useful information directly from data, rather than having an explicitly programmed behavior like heuristic methods. In fact, it has been known that, even though heuristics would get stuck in local optimums, they often offer stable and high-quality solutions, especially for large-scale instances and when the problem involves complex constraints. The RL approach is better in exploring new solutions and escaping from local optimums, but needs time to train and would be unstable with complex constraints. This motivates us to combine heuristics and RL in such a way that RL can learn and benefit from heuristic operators.

In this work, we propose an imitation reinforcement learning algorithm trained via policy gradient to learn improvement heuristics based on k-opt moves and treat advanced heuristics (e.g., VNS or HGS) as experts to *teach* the policy model. Our work aims to enhance the deep RL process via heuristic methods and address scalability by learning from smaller sub-problems simultaneously. That is, for each step, a certain number of nodes are selected in turn clockwise, following by an initial solution, to form sub-problems to our RL policy for ease of learning. In fact, our *clockwise* mechanism offers a good initial solution structure and a natural way to decompose the whole problem into sub-problems that can be solved and learned simultaneously. Moreover, a solution returned by the RL policy model will be fed to a heuristic method to be further improved. Solutions from heuristics are also collected to build an imitation learning model that will be integrated back into the RL policy to persuade the RL to produce similar solutions.

In summary, by combining the clockwise framework, heuristic methods and imitation learning, we bring several advantages to the same place:

(i) Starting with poor quality solutions, we first use heuristics to improve solutions, and heuristic operators are learned and imitated by deep learning networks (i.e., Ho and Ermon (2016)) to generate similar or better solutions at each step. In fact, by letting deep RL and imitation learning work together in an iterative manner, we encourage the algorithm to both explore new solutions (by deep RL) and exploit operators that lead to high-quality solutions (i.e. imitation learning mimicking heuristics).

(ii) The whole network is clustered into smaller sub-problems of similar distributions, allowing our algorithms to process them quickly and scale up.

(iii) Heuristics play as experienced experts (teacher/corrector) that help the deep RL generate more stable and high-quality solutions. Faster convergence is to be expected with the powerful performance of heuristic methods. Moreover, since solutions are always corrected by heuristics to ensure feasibility, the Policy Network does not need to control infeasible solutions.

Altogether, we make the following contributions:

- We propose *Imitation Improvement Learning*, a new learning-based framework using policy gradient with a heuristic method that serves as an expert/teacher to correct and improve any solution to ensure feasibility and teach the policy gradient to generate high-quality solutions.
- We propose *Clockwise Clustering*, a recursive sub-problem decomposition framework to handle large-scale CVRP instances.
- We offer new state-of-the-art solutions for some well-known large-scale CVRP instances.

Our experiments demonstrate the scalability of our *Clockwise Clustering* in solving large-scale real-world instances, the benefits of using imitation learning to get high-quality solutions, and the generalizability of our learning-based model in solving instances of similar distributions. Codes and models are available at https://github.com/vietbt/VRPpp.

Related Work

Heuristics for solving combinatorial optimization problems have been developed for decades. The most powerful methods, such as local search (Crama, Kolen, and Pesch 1995), genetic algorithms (Heiss-Czedik 1997), and ant colony methods (López-Ibáñez 2010), involve iteratively improving solutions in a hand-designed neighborhood search. For example, *move*, *swap* (Wu et al. 2016), and *2-opt* (Croes 1958) are well-known heuristics for the traveling salesman and vehicle routing problems. Examples of the state-of-the-art heuristic algorithms for the CVPR would be the *HGS* (Vidal 2022) that uses a hybrid genetic and local search procedure to achieve state-of-the-art solution qualities on instances of sizes up to 1000, or the *LKH-3* (Helsgaun 2017) that uses the Lin-Kernighan heuristic as a backbone, which involves swapping pairs of sub-routes to create new routes. HGS is considered one of the best heuristic algorithms for CVRP, together with LKH-3, but HGS is more focused on CVRP and LKH-3 was designed for other variants. Our framework can be straightforwardly adapted for other heuristics. For large-scale problem instances, low-level heuristics are often combined with meta-heuristics to achieve good performance, e.g., Pruning and Sequential Search (Arnold, Gendreau, and Sörensen 2019), Spatial Partitioning Strategies (Tu et al. 2017), Constrained Clustering (Alesiani, Ermis, and Gkiotsalitis 2022), and Cluster-First Route-Second (Shalaby, Mohammed, and Kassem 2021).

In recent years, there have been a number of studies focusing on using deep RL to solve combinatorial optimization problems. Those models are categorized into two classes: *construction* and *improvement methods* (Kwon et al. 2020).

- *Construction methods* (Nazari et al. 2018): Starting with an empty solution, a construction method constructs a solution by sequentially assigning each customer to a vehicle until all customers are served. Construction methods still require additional procedures such as beam search, classical improvement heuristics, and sampling to achieve such results.
- *Improvement methods* (Chen and Tian 2019; Hottung and Tierney 2020): Starting with a complete initial solution, the methods select either candidate *nodes* (customers or depot) or heuristic *operators* (or both) to improve and update the solution at each step. This is repeated until termination. Here, if one can learn a policy to improve a solution, such a policy can be used to obtain better solutions from a construction heuristic or even random solutions. Studies have shown that *improvement* methods are able to provide better solutions than *construction* ones (Lu, Zhang, and Yang 2020; da Costa et al. 2021).

Deep RL approaches have achieved competitive results as compared to classical heuristics. For example, Lu, Zhang, and Yang (2020) propose *Learning-to-improve* based on Meta-controller learning, which outperforms LKH-3 but only works on small-scale problems. da Costa et al. (2021) propose *Learning 2-opt* based on learning from local search operations, which also only works with small-scale problem instances. Recently, Li, Yan, and Wu (2021) develop a *learning to delegate* approach in which sub-problem are selected and learned. This method outperforms LKH-3 and works well with uniformly large-scale problems. It is moreover quite scalable and is efficient for generalization. However, since the approach requires a large dataset of instances for training and similar data distributions for testing, the performance on non-uniform large-scale CVRP instances such as CVRPLIB would be poor. In fact, there are very few learning-based experimental studies on very large-scale instances. Although deep RL's learnability is appealing, trajectory collection becomes prohibitively expensive for large-scale problem instances.

Background

Capacitated Vehicle Routing Problems

CVRP can be defined by a fully connected weighted graph $\mathcal{G} = (\mathbb{V}, \mathbb{A})$, where $\mathbb{V} = \{0 \cup \mathbb{I}\}$ stands for a set of nodes and $\mathbb{A} = \{(i,j)|i,j \in \mathbb{V}, i \neq j\}$ denotes a set of arcs connecting these nodes. Set $\mathbb{I}$ denotes the set of customers and 0 denotes the central depot. Each arc of the network is associated with a non-negative value d_{ij} representing the distance between two nodes i and j. In case of Euclidean distance, $d_{ij} = \|x_i - x_j\|_2$, where x_i, x_j are vectors of spatial coordinates of nodes i and j, respectively. Each customer $k \in \mathbb{I}$ is assigned a positive demand $b_k > 0$. At central depot, the demand b_0 is set to 0. We also let $\mathbb{B} = \{b_k|k \in \mathbb{I}\}$ denote the set of demands. The objective function of the CVRP, assuming that the fleet of vehicles is homogeneous, is to seek a set of routes that minimize the total traveling distance. Vehicles start and end at the depot and for every route, the total demand of customers does not exceed the maximal carrying capacity $C > 0$ of the vehicle. In Appendix we provide a

more detailed description and a mixed-integer formulation for the CVRP problem.

k-Opt Heuristic for the CVRP

An improvement heuristic concerns a search procedure that iteratively improves feasible solutions. A procedure can start with an initial solution S_0 and iteratively search a better solution S_{t+1} from a current solution S_t. Local search methods such as Lin-Kernighan-Helsgaun (LKH) (Helsgaun 2017) performs well for CVRP. The procedure seeks for k edge swaps (k-opt moves) that could be replaced by new edges to form a shorter tour. Sequential pairwise operators such as k-opt moves can be decomposed in simpler k'-opt ones ($k' < k$). For instance, sequential 3-opt operations can be decomposed into one, two or three 2-opt operations. However, in local search algorithms, the quality of the initial solution usually affects the quality of the final solution, i.e. local search methods can easily get stuck in local optima (Hansen and Mladenović 2006). To avoid local optima, different meta-heuristics have been proposed, including Simulated Annealing and Tabu Search, which work by accepting bad solutions to enhance exploration on the searching space. Meta-heuristics, nevertheless, still require expert knowledge and rely on sub-optimal rules in their designs.

Clockwise Clustering

In this section we formally introduce the "Clockwise Clustering" framework, our method for solving large-scale CVRPs. Figure 3 illustrates an overview of this framework. From a large-scale instance input $(\mathcal{G}, \mathbb{B}, C)$, we generate an initial solution S_0 by a *clock-hand initializer*, a simple procedure that arranges all nodes in clockwise order and sequentially groups nodes into tours satisfying that the total demand of each tour is not greater than the demand capacity C. Clock-hand initializer basically produces a feasible solution that all tours $T = [t_1, t_2, ..., t_{n_T}]$ are ordered in clockwise direction. Note that sorting tours in clockwise direction means that tours are arranged by the angle between y-axis and the line connecting the centroid point and the depot of each tour. The first u tours (i.e., a proportion of all the tours) $[t_1, t_2, ..., t_u]$ are then selected to form a sub-instance $(\mathcal{G}_{sub}, \mathbb{B}, C)$. Here, only a subset of tours is selected to process, instead of all the tours, to enhance the scalability. After solving this sub-instance by the *Imitation Improvement Learning* framework (described in the next section), we then send the first v tours of the returned sub-solution to *a sub-solution buffer* to keep the results. The unprocessed tours and nodes are then collected and sent back to the beginning of the cycle loop for the next round of the *Imitation Improvement Learning*. When all the nodes are processed, we take the processed tours from the *sub-solution buffer* to build a complete (and feasible) solution S_1 of $\mathcal{G}$. After that, S_1 could be processed same as S_0 to create new solutions $S_2, S_3, S_4, ...$ We accept bad solutions to allow more exploration on the search space. After a certain number of loops, we return the best solution among $\{S_1, S_2, S_3, ...\}$. We provide an overview of the framework in Figure 1.

The *clock-hand initializer* offers a good initial solution and a natural way to decompose the whole problem into sub-problems of similar distributions, allowing heuristics to process them quickly and providing RL with sub-instances of similar distributions for efficient training. The *learning-to-delegate* framework (Li, Yan, and Wu 2021) also endeavors to select sub-problems to improve scalability, but differ from our approach by the fact that the *learning-to-delegate* learns to select sub-problems and uses traditional heuristics (HGS/LKH3) with a huge number of running steps to achieve good sub-instance selections. Instead, our method does not require such a large number of steps as the sub-instance decomposition is embedded as part of the heuristic loop and cooperates with heuristics so both will be improved over iterations.

Algorithm 1: Clockwise Clustering

$S \leftarrow$ Initializer$(\mathcal{G}, \mathbb{B}, C)$ `// Initial solution`
$S^* \leftarrow S$ `// Best solution`
$S' \leftarrow []$ `// Solution buffer`
while repeat a certain number of times **do**
  $S_{sub} \leftarrow S_{:u}, S_{other} \leftarrow S_{u:}$ `// First u tours`
  $S^*_{sub} \leftarrow$ IILSolver$(S_{sub}, \mathbb{B}, C)$
  $S'_{sub} \leftarrow S^*_{sub,:v}$ `// First v tours`
  $S'_{other} \leftarrow S^*_{sub,v:}$ `// Remaining portion`
  $S' \leftarrow S' \oplus S'_{sub}$
  $S \leftarrow S_{other} \oplus S'_{other}$ `// Unprocessed nodes`
  if S is empty **then**
    if Cost(S') < Cost(S^*) **then**
      $S^* \leftarrow S'$ `// Best solution ever`
    end if
    $S \leftarrow S^*$
  end if
end while
return S^*

The pseudo-code of the *Clockwise Clustering* framework is provided in Algorithm 1. We first employ Clock-hand initializer for creating initial solution S and send it into a loop to find the best improvement solution S^*. We denote by $S_{:t}$ and $S_{t:}$ the sub-problems containing nodes from the first t tours and the rest , respectively, in clockwise order. Inspired by the *Divide-and-Conquer* mechanism, we select the first u tours of S to form a sub-problem S_{sub} (e.g., $S_{:u}$) and forward it to the *Imitation Improvement Learning* framework to find an optimal solution S^*_{sub}. We then put its first v tours (e.g., $S^*_{sub,:v}$) to a solution buffer S'. After that, we combine all unprocessed nodes from $S_{u:}$ and $S^*_{sub,v:}$ into S and continue to find improved solutions until S is empty. We describe in detail the selection of u and v in the appendix.

Imitation Improvement Learning

In this section, we present our *Imitation Improvement Learning* (IIL) framework, a main component of the *Clockwise Clustering* for solving sub-instances. Figure 2 provides an overview of the framework. Our framework is iterative in nature; a sub-solution S is improved after each cycle loop. Starting with an initial sub-solution as S, a state

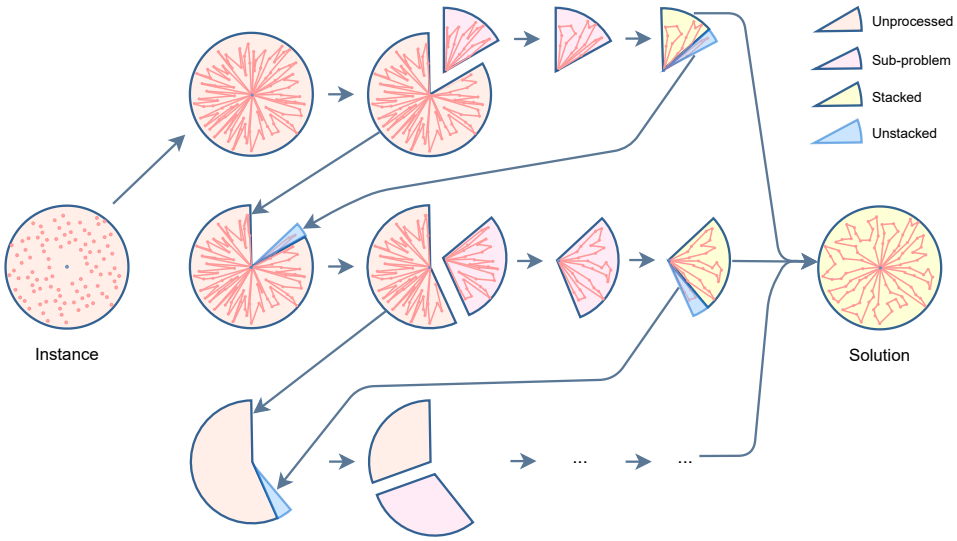


Figure 1: A visualization of the *Clockwise Clustering* framework: The clock-hand initializer is used to construct an initial solution. At each round, a small proportion of the tours are selected to form a sub-problem and sent to *Imitation Improvement Learning* to improve. A majority of the output are kept in a *sub-solution buffer* (*stacked*) and the remaining portion (*unstacked*) is merged with unprocessed tours to start the next round. Once all nodes are processed, we get all the sub-solutions from the sub-solution buffer and build a complete solution, save it and start a new round with the new solution to further improve it.

$s(\mathcal{G}, S, \mathbb{B}, C)$ is forwarded to a neural encoder-decoder network to approximate the stochastic policy $\pi_\theta(a|s)$, where θ are trainable parameters. The value function $V_\phi(s)$ is also a neural network, where ϕ are trainable parameters. This platform uses policy gradient to optimize the parameters of the policy and value functions of the RL network. Here, thanks to the *Clockwise Clustering* platform, the sub-instances sent to IIL are significantly smaller in size, as compared to the original instance, and are of similar numbers of nodes. This matches well the ability of learning-based RL models to produce high-quality solutions for small-sized instances (Chen and Tian 2019; Hottung and Tierney 2020). Intuitively, a good RL policy model would provide the heuristics with better local search space, leading to better expert solutions to further teach and improve the RL through imitation learning. Figure 3 provides an overview of our frameworks, showing how the *Clockwise Clustering* and *Imitation Improvement Learning* frameworks work together. Detailed steps of the *Imitation Improvement Learning* framework are provided in Algorithm 2.

Imitation Learning with Experts

The key component of the IIL framework is an imitation learning model that uses heuristics' solutions to teach the RL policy to generate high-quality solutions. More precisely, the RL policy model acts as a student who wants to learn from experts. Classical heuristics are ideal experts, which are also iterative and are able to generate good solutions quickly. Combining the loop of RL policy (student) and heuristics (expert), we have a closed loop where the imitation learning model encourages the RL policy to mimic the heuristics' policy. There are two main parts of RL policy: *Encoder* (extracting features from input states to representation vectors) and *Decoder* (generating actions based on representation vectors). As the heuristic agents only give solutions and there is no direct policy associated with them, we propose to mimic the encoder network of the heuristics To this end, we adapt the *generative adversarial imitation learning* (GAIL) algorithm (Ho and Ermon 2016), which is based on a discriminative neural model D_δ to distinguish between solutions generated by the RL policy and the expert (i.e. heuristics), where δ are trainable parameters. Modifying GAIL with encoder feature extraction, our objective is $\max_\theta \min_\delta \mathbb{E}_{s\sim Expert}[\log(D_\delta(\text{Encoder}_\theta(s)))] + \mathbb{E}_{s\sim Student}[\log(1 - D_\delta(\text{Encoder}_\theta(s)))]$, where $\text{Encoder}_\theta(s)$ refers to the representation vector of state s. By solving the max-min problem, one can force the RL policy to generate solutions whose encoder feature extractions are similar to those from expert's solutions.

Improvement Learning with k-Opt Moves

Inspired by the *Learning 2-opt* (da Costa et al. 2021), our algorithm will continuously select k different nodes and swap its edges in the selection order to form a student solution. It is to be expected that RL will learn to find out which order would be good for the learning. The RL policy samples an action a which contains k nodes in $\mathbb{I}$ used for k-opt moves to generate a student solution $S_{student}$. After that, this solution is forwarded to the *Local Search* component of classical heuristics to get an expert solution S_{expert}. The heuristics do not need to be refreshed after generating this solution. Expert solution S_{expert} continues to be forwarded to RL policy

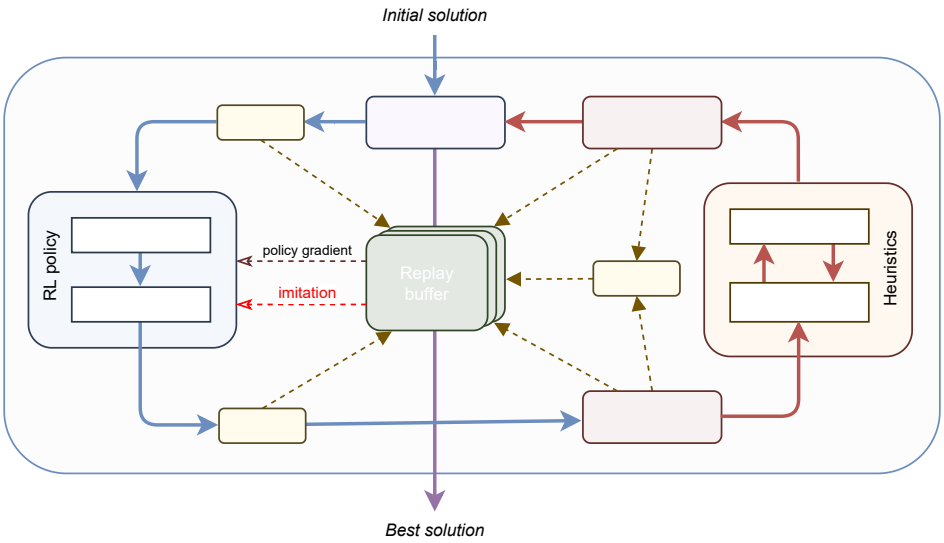


Figure 2: An overview of our *Imitation Improvement Learning* framework: Starting with an initial sub-solution S, the input is forwarded to an imitation cycle loop between RL Policy (student) and Heuristics (expert) to be improved after each iteration. At each iteration, the RL Policy, together with k-opt operators, generate a student solution and brings it directly to the traditional heuristics's local search to produce an expert solution. These two solutions are used to calculate rewards and imitation loss for training RL policy with policy gradient.

for a new loop. As $S_{student}$ is not always feasible, rewards are computed by the gap between input and output solutions, e.g., $r = \text{Cost}(S) - \text{Cost}(S_{expert})$. For each step, replay buffer $\mathcal{D}$ collects states, actions, rewards and expert/student solutions for updating model parameters via policy gradient with RL loss and imitation learning loss.

Network Architecture

Our neural network is based on an encoder-decoder architecture (a detailed description is given in appendix). The encoder learns representations that embed graph topology. We create node features $\boldsymbol{X} \in \mathbb{R}^{|\mathbb{V}|\times 3}$ (x-coordinate, y-coordinate, and demand rate), edge features $\boldsymbol{E} \in \mathbb{R}^{|\mathbb{A}|\times 2}$ (euclidean distance and radian angle between each edge and x-axis), and import them into the Residual E-GAT (Lei et al. 2021) for feature extraction. Given these representations, the policy decoder samples action indices $a_1, a_2, ..., a_k$ sequentially for k-opt. We aim to learn the parameters of a stochastic policy $\pi_\theta(a|s)$ that, given a state s, assigns high probabilities to moves that reduce the cost of a tour. Our architecture uses a chain rule to factorize the probability of a k-opt move as $\pi_\theta(a|s) = \prod_{i=1}^{k} p_\theta(a_i|s)$. We use a pointing mechanism (da Costa et al. 2021) to predict a distribution over node outputs given encoded actions (nodes) and a state representation (query vector). The value decoder operates on the same encoder outputs but outputs real-valued estimates of state values. We give more details of network architecture in Appendix .

Loss Function

We use PPO (Schulman et al. 2017) for policy gradient optimization with the loss function $\mathbb{E}_{\pi_\theta}[\mathcal{L}^{PPO}_{\pi_\theta}]$. We also add an imitation loss to for imitation learning, leading to the following overall loss

$$\mathcal{L}^{IIL}_{\pi_\theta} = \mathcal{L}^{PPO}_{\pi_\theta} + c_{IL}\Big\{\min_{\delta} \mathbb{E}_{s\sim Expert}\left[\log\left(D_\delta\left(\text{En}_\theta(s)\right)\right)\right] + \mathbb{E}_{s\sim Student}\left[\log\left(1 - D_\delta\left(\text{En}_\theta(s)\right)\right)\right]\Big\},$$

where En_θ stands for our Encoder_θ model, the first term is the PPO loss from standard RL policy and the second term (with weight parameter c_{IL}) is from the imitation learning model. By optimizing $\max_\theta \mathbb{E}[\mathcal{L}^{IIL}_{\pi_\theta}]$, we seek for a policy that both maximizes the standard long-term reward function of the RL policy and mimics the heuristics' policy feature extraction. Intuitively, the first term of the loss function is to encourage exploration of new solutions and the second term is to exploit high-quality solutions from heuristics.

Experiments and Results

We provide extensive experimental results based on some large-scale well-known CVRP datasets, targeting the following questions.

(i) By using heuristic methods as an expert/teacher for the policy model, can our *Imitation Improvement Learning framework* outperform the standing-alone heuristics?

(ii) Can the *Clockwise Clustering* and *Imitation Improvement Learning* frameworks help us solve very-large-scale instances with competitive performance?

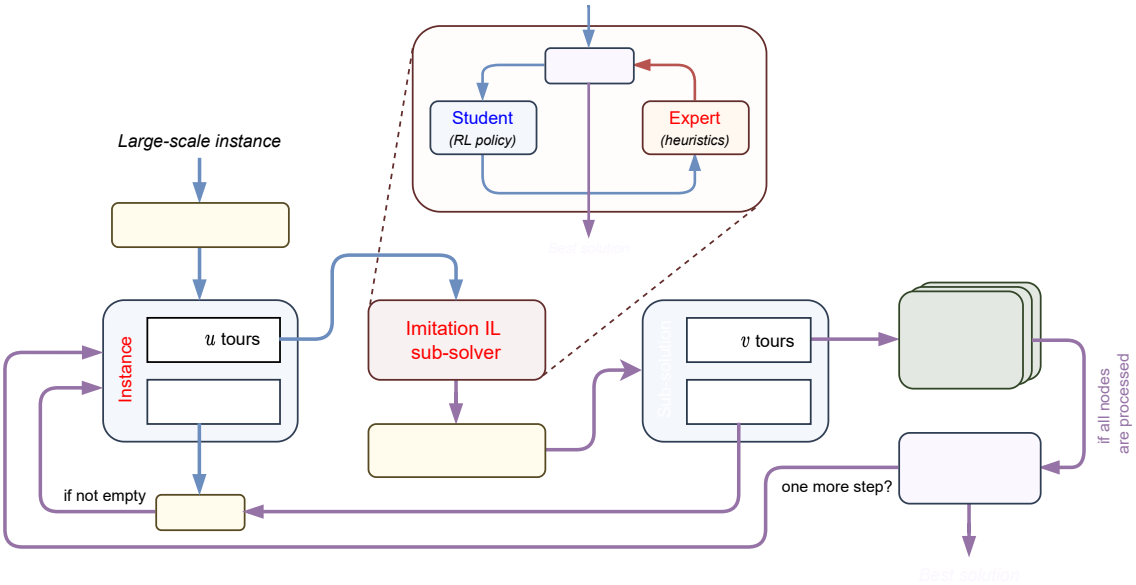


Figure 3: An overview of our *Clockwise Clustering* platform.

Algorithm 2: Imitation Improvement Learning

$S \leftarrow \text{InitSolution}(\mathcal{G}, \mathbb{B}, C)$ `// Global solution`
$\mathcal{D} \leftarrow []$ `// Replay buffer`
while repeat a certain number of times **do**
  $s \leftarrow \text{InputFeatures}(S, \mathcal{G}, \mathbb{B}, C)$ `// State`
  $a \leftarrow \text{SampleNodesFromPolicy}(s)$ `// Action`
  $S_{student} \leftarrow \text{ProcessOperators}(S, a)$ `// k-opt`
  $S_h \leftarrow S_{student}$ `// Heuristic solution`
  while repeat a certain number of times **do**
    $S'_h \leftarrow \text{LocalSearch}(S_h)$
    $S^*_h \leftarrow \text{Perturbation}(S'_h)$
    $S_h \leftarrow \text{AcceptanceCriterion}(S'_h, S^*_h)$
  end while
  $S_{expert} \leftarrow S_h$
  $r \leftarrow \text{Cost}(S) - \text{Cost}(S_{expert})$ `// Reward`
  $\mathcal{D} \leftarrow \mathcal{D}.\text{append}((s, a, r, S_{student}, S_{expert}))$
  $S \leftarrow S_{expert}$ `// Update global solution`
  if $|\mathcal{D}| > n_{\mathcal{D}}$ **then**
    $\text{UpdateRLWithImitation}(\mathcal{D})$
    $\mathcal{D} \leftarrow []$ `// Clear replay buffer`
  end if
end while
return S

(iii) Can our frameworks *generalize* to other instance sets of similar distributions?

(iv) Can our algorithms offer new *state-of-the-art (SOTA)* solutions for some popular CVRP instances?

As HGS is considered one of the best heuristic algorithms for CVRP, we use HGS as the expert heuristic of our IIL framework. Besides, VNS is a popular heuristic method. The use of VNS is to explore the impact of different heuristics on our approach. Below, we present our datasets and our comparison results. We trained each dataset separately and evaluated on the test set if the dataset provides one, or the training set otherwise. Other details can be found in the appendix.

Dataset

We benchmark our frameworks using large-scale instances from three recent datasets from CVRPLIB (http://vrp.atd-lab.inf.puc-rio.br/index.php/en/), which are known to be challenging for both heuristic and learning-based methods. To test the generalizability of our clockwise clustering (CC) and IIL method, we experiment on an uniform dataset from Li, Yan, and Wu (2021). We also benchmark on *constrained electrical vehicle routing* (CEVRP) datasets, a CVRP variant for electrical vehicles with battery constraints. For this, Mavrovouniotis et al. (2020) contribute a large-scale CEVRP dataset containing 17 instances. Same as CVRP, we test the generalizability of our approaches in CEVRP by using this dataset to train the RL policy and uniformly generating 238 new instances to serve as a test set.

Six datasets used for benchmarking are listed in Table 1. First, we try to test the efficiency of our algorithms, compared to the heuristic counterparts, using Dataset 1. Next, we test the generalizability of our model by learning with a train set and evaluating with another test set of Dataset 2. We then benchmark our algorithms with two real-world large-scale datasets, called as Dataset 3&4, which are collected in Brazil and Belgium, respectively. Although most of the CVRP datasets use 2D euclidean distances, the DIMACS dataset (Dataset 3) use weights specifically defined for pairs of nodes. For CEVRP, we benchmark with instances from Mavrovouniotis et al. (2020) (named as Dataset 5) and our newly generated instances (Dataset 6).

Dataset	Source	VRP Type	Distance Space	# instances	# customers
1	Uchoa et al. (2017)	CVRP	Euclidean	100	100-1000
2	Li, Yan, and Wu (2021)	CVRP	Euclidean	2000/40/40	500-2000
3	DIMACS[1]	CVRP	Explicit	12	241-1000
4	Arnold, Gendreau, and Sörensen (2019)	CVRP	Euclidean	10	3000-30000
5	Mavrovouniotis et al. (2020)	CEVRP	Euclidean	17	21-1000
6	Ours	CEVRP	Euclidean	238	21-1000

Table 1: Datasets.

Experimental Results and Analysis

In this section, we use GAP (percentage) scores defined below as the main criteria for comparison. A GAP score is calculated as the percentage gap w.r.t the corresponding BKS (best-known-solutions), i.e.,

$$GAP(X, BKS) = \frac{Cost(S_X) - Cost(S_{BKS})}{Cost(S_{BKS})}.$$

For Datasets 1-4, the BKS were taken from the datasets' authors (continuously updated on their website[2]), which are found by any method with unknown computing resources. We also use BKS from the recent CEC-12 competition for Dataset 5, and VNS as a baseline for Dataset 6 (our new dataset). We do not aim to compare our approach with BKS; rather, they are only used to compare our frameworks against other well-known baseline methods.

Method	GAP vs BKS
OR-tools	-4.01%
VNS	-3.08%
HILS	-1.00%
KGLS	-0.66%
SISR	-0.54%
HGS (30k)	-0.30%
HGS (95%)	-0.48%
RL+VNS (ours)	-2.15%
RL+HGS (ours)	-0.31%
IIL+VNS (ours)	-1.79%
IIL+HGS (ours)	**-0.27%**

Table 2: Experimental results for Dataset 1.

With Dataset 1, we compare our methods with OR-tools, VNS, HILS (Subramanian, Uchoa, and Ochi 2013), KGLS (Arnold and Sörensen 2019), SISR (Christiaens and Vanden Berghe 2020), and HGS (Vidal 2022). Instead of running HGS with full settings, we compare with HGS 30k steps and HGS 95% solution quality, similarly to Li, Yan, and Wu (2021). Table 2 reports the results from previous works and our results, compared to the BKS by GAP scores, where IIL stands for our *Imitation Improvement Learning* method and RL stands for our learning-based framework but without the imitation learning loss. Our methods are clearly better than the corresponding standing-alone heuristics (i.e., RL+VNS and IIL+VNS versus VNS, and RL+HGS and IIL+HGS versus HGS). For instance, with HGS, our IIL method get -0.27% GAP vs BKS, better than -0.30% of HGS 30k steps. In terms of running time, our algorithms need about 10 hours while HGS takes about 16 to 40 hours to finish. We do not report the running times of the other methods because they are not reported in their respective papers, and these methods are clearly outperformed by our algorithms in terms of solution quality. The results also indicate that our IIL framework works the best with HGS subsolver.

Table 3 shows a comparison of our best method IIL+HGS with previous results reported for Dataset 2 (Li, Yan, and Wu 2021). Note that all cost scores are divided by $1e5$. Our costs are better than the previous SOTA results obtained by Learning-to-delegate (L2D) (Li, Yan, and Wu 2021) for all the three sub-datasets.

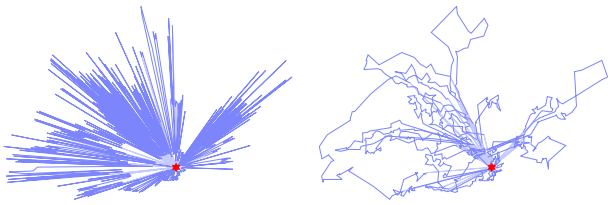
Method	Dataset 2		
	N = 500	N = 1000	N = 2000
LKH-3 (95%)	62.00	120.02	234.89
LKH-3 (30k)	61.87	119.88	234.65
OR-tools	65.59	126.52	244.65
AM sampling	69.08	151.01	356.69
AM greedy	68.58	142.84	307.86
NeuRewriter	73.60	136.29	257.61
Random	61.99	120.02	234.88
Count-based	61.99	120.02	234.88
Max Min	61.99	120.02	234.89
L2D (short)	61.99	119.87	234.89
L2D (long)	61.70	119.55	233.86
IIL+HGS (**ours**)	**60.49**	**118.37**	**225.43**

Table 3: Average costs for instances of different sizes $N \in \{500, 1000, 2000\}$ of Dataset 2.

Table 4 shows the results for Dataset 3 & 4. For Dataset 3, the classical HGS's score is not significantly better than that of the IIL+HGS. Note that it is a non-euclidean dataset so we are not able to set correctly the node and edge features for our E-GAT encoder. It might affect the performance of our framework. In addition, initial solutions created by our *clock-hand* could be worse, making it difficult for our *Clockwise Clustering* framework to split nodes into sub-instances. Nevertheless, we achieved a new SOTA result for *Loggi-n501-k24* instance; our solution cost is 177078, better than the previous one 177176 reported by the CVRPLIB authors

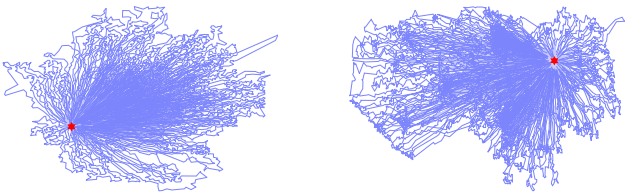
[1]The 12th DIMACS Implementation Challenge (http://dimacs.rutgers.edu/programs/challenge/vrp/cvrp/cvrp-competition)

[2]The Capacitated Vehicle Routing Problem Library (http://vrp.atd-lab.inf.puc-rio.br/index.php)



Initial solution - cost: 1406303 | IIL+HGS - cost: 285021

Figure 4: Instance *Loggi-n1001-k31* from *Dataset 3*, which does not support 2D Euclidean space.



Brussels | Flanders

Figure 5: Solutions generated by IIL+HGS for two very-large-scale instances of *Dataset 4*, which represent the real-world maps of Brussels&Flanders, Belgium with sizes up to 30k.

(http://vrp.galgos.inf.puc-rio.br/index.php/en/updates/). For Dataset 4, even-though it contains very-large-scale instances of sizes up to 30k, our IIL+HGS performs better than the classical HGS and any other methods. We illustrate initial solutions (from the Clock-hand initializer) and our best solutions for two large instances from Dataset 3&4 in Figure 4 and 5.

Methods	Dataset 3	Dataset 4
CW	-	-8.00%
MDM	**-0.10%**	-3.63%
HGS	**-0.10%**	-4.89%
IIL+VNS (ours)	-0.80%	-6.71%
IIL+HGS (ours)	**-0.10%**	**-2.95%**

Table 4: GAP vs BKS (percentage) for two real-world datasets.

Table 5 reports results for large-scale CEVRP instances from Dataset 5&6, where "-" indicates that the information is not available from previous works. For these instances, battery information is added to the node features. IIL+HGS outperforms other classical heuristics. Specifically, IIL+HGS gets a new SOTA results with 1.88% GAP vs BKS. Note that HGS does not support the CEVRP variant with battery constraints, so we are not able to embed directly HGS library to our IIL framework. To make it work, we add a post-processing step, similarly to the GRASP (Woller, Kozák, and Kulich 2021), to rebuild feasible solutions that fit the battery constraints. To test the generalizability of our approach, we run our IIL algorithms on Dataset 6 and compare the results with VNS and BACO (i.e., SOTA algorithms for the CEVRP instances), which clearly shows that IIL+HGS is much better than the heuristics in terms of both solution quality and running time.

Methods	Dataset 5	Dataset 6
GA	-4.57%	-
SA	-2.65%	-
VNS	-1.08%	0.00%
BACO	-0.43%	0.13%
IIL+VNS (ours)	0.43%	1.36%
IIL+HGS (ours)	**1.88%**	**2.15%**

Table 5: GAP vs BKS (percentage) for CEVRP datasets

Conclusion

We proposed a new learning-based framework for CVRP, which employs heuristic methods as an expert to teach the RL policy model to generate high-quality solutions. To enhance the scalability and take the advantage of the RL approaches in learning from similar instances, we propose the *Clockwise Clustering* framework that offers good initial solutions and a nature way to decompose the whole instances into sub-instances having similar numbers of nodes. We benchmarked on several popular large-scale CVRP instances of sizes up to 30k, showing that our proposed algorithms outperform the respective standing-alone heuristics and offer competitive solutions, compared to previous SOTA algorithms. Our methods also archive new best solutions for several instances and generalize for a wide range of CVRP distributions and solvers. In general, we highlight the effectiveness of using generative adversarial imitation learning to help RL and heuristic methods work together in an iterative manner. Our clockwise clustering can be used with other VRP solvers. Exploring the use of this framework for other RL and heuristic algorithms would be an interesting direction for future work. It is also interesting to see how our frameworks can be extended to solve other challenging combinatorial optimization problems.

Acknowledgments

This research/project is supported by the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-017).

References

Alesiani, F.; Ermis, G.; and Gkiotsalitis, K. 2022. Constrained Clustering for the Capacitated Vehicle Routing Problem (CC-CVRP). *Applied Artificial Intelligence*.

Arnold, F.; Gendreau, M.; and Sörensen, K. 2019. Efficiently solving very large-scale routing problems. *Computers & Operations Research*, 107: 32–42.

Arnold, F.; and Sörensen, K. 2019. What makes a VRP solution good? The generation of problem-specific knowledge for heuristics. *Computers & Operations Research*, 106: 280–288.

Chen, X.; and Tian, Y. 2019. Learning to Perform Local Rewriting for Combinatorial Optimization. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*.

Christiaens, J.; and Vanden Berghe, G. 2020. Slack Induction by String Removals for Vehicle Routing Problems. *Transportation Science*, 54(2): 417–433.

Crama, Y.; Kolen, A. W. J.; and Pesch, E. J. 1995. Local search in combinatorial optimization. In *Artificial Neural Networks: An Introduction to ANN Theory and Practice*, 157–174.

Croes, G. A. 1958. A Method for Solving Traveling-Salesman Problems. *Operations Research*, 6: 791–812.

da Costa, P.; Rhuggenaath, J.; Zhang, Y.; Akcay, A.; and Kaymak, U. 2021. Learning 2-Opt Heuristics for Routing Problems via Deep Reinforcement Learning. *SN Computer Science*, 2(5): 388.

Hansen, P.; and Mladenović, N. 2006. First vs. best improvement: An empirical study. *Discrete Applied Mathematics*, 154(5): 802–817.

Heiss-Czedik, D. 1997. An Introduction to Genetic Algorithms. *Artificial Life*, 3: 63–65.

Helsgaun, K. 2017. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, 24–50.

Ho, J.; and Ermon, S. 2016. Generative Adversarial Imitation Learning. In *Proceedings of the 29th International Conference on Neural Information Processing Systems*, volume 29.

Hottung, A.; and Tierney, K. 2020. Neural Large Neighborhood Search for the Capacitated Vehicle Routing Problem. 325: 443–450.

Kool, W.; van Hoof, H.; and Welling, M. 2018. Attention, Learn to Solve Routing Problems! In *Proceedings of the 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. 33: 21188–21198.

Laporte, G. 2009. Fifty years of vehicle routing. *Transportation science*, 43(4): 408–416.

Lei, K.; Guo, P.; Wang, Y.; Wu, X.; and Zhao, W. 2021. Solve routing problems with a residual edge-graph attention neural network. *Neurocomputing*, 508: 79–98.

Lenstra, J. K.; and Kan, A. H. G. R. 1981. Complexity of vehicle routing and scheduling problems. *Networks*, 11: 221–227.

Li, S.; Yan, Z.; and Wu, C. 2021. Learning to delegate for large-scale vehicle routing. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*.

López-Ibáñez, M. 2010. Ant Colony Optimization. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, 2353–2384.

Lu, H.; Zhang, X.; and Yang, S. 2020. A Learning-based Iterative Method for Solving Vehicle Routing Problems. In *Proceedings of the International Conference on Learning Representations*.

Mavrovouniotis, M.; Menelaou, C.; Timotheou, S.; Panayiotou, C. G.; Ellinas, G.; and Polycarpou, M. M. 2020. Benchmark Set for the IEEE WCCI-2020 Competition on Evolutionary Computation for the Electric Vehicle Routing Problem.

Nazari, M.; Oroojlooy, A.; Snyder, L.; and Takac, M. 2018. Reinforcement Learning for Solving the Vehicle Routing Problem. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, volume 31.

Prates, M.; Avelar, P.; Lemos, H.; Lamb, L.; and Vardi, M. 2019. Learning to Solve NP-Complete Problems: A Graph Neural Network for Decision TSP. 33: 4731–4738.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347.

Shalaby, M.; Mohammed, A.; and Kassem, S. 2021. Supervised Fuzzy C-Means Techniques to Solve the Capacitated Vehicle Routing Problem. *The International Arab Journal of Information Technology*, 18(34A): 452–463.

Subramanian, A.; Uchoa, E.; and Ochi, L. 2013. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40: 2519–2531.

Toth, P.; Vigo, D.; Toth, P.; and Vigo, D. 2014. Vehicle Routing: Problems, Methods, and Applications, Second Edition.

Tu, W.; Li, Q.; Li, Q.; Zhu, J.; Zhou, B.; and Chen, B. Y. 2017. A spatial parallel heuristic approach for solving very large-scale vehicle routing problems. *Transactions in GIS*, 21: 1130 – 1147.

Uchoa, E.; Pecin, D.; Pessoa, A.; Poggi, M.; Vidal, T.; and Subramanian, A. 2017. New benchmark instances for the Capacitated Vehicle Routing Problem. *European Journal of Operational Research*, 257(3): 845–858.

Vidal, T. 2022. Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. *Computers & Operations Research*, 140: 105643.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer Networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, 2692–2700.

Woller, D.; Kozák, V.; and Kulich, M. 2021. The GRASP Metaheuristic for the Electric Vehicle Routing Problem. In *Modelling and Simulation for Autonomous Systems*, 189–205.

Wu, C.; Shankari, K.; Kamar, E.; Katz, R. H.; Culler, D. E.; Papadimitriou, C. H.; Horvitz, E.; and Bayen, A. M. 2016. Optimizing the diamond lane: A more tractable carpool problem and algorithms. *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 1389–1396.