Carnegie Mellon University
Chemical Engineering

REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF Master of Science

TITLE SOLVING VEHICLE ROUTING PROBLEMS WITH DEEP REINFORCEMENT LEARNING

PRESENTED BY **Kiran Prasad Jamuna Prasad**

ACCEPTED BY THE DEPARTMENT OF

Chemical Engineering

Chrysanthos Gounaris

Chrysanthos Gounaris, PROFESSOR AND ADVISOR

12/9/24

DATE

Carl Laird, PROFESSOR AND DEPARTMENT HEAD

DATE

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to those who have supported me throughout the process of completing this thesis. Without their guidance, encouragement, and assistance, this work would not have been possible.

I embarked on my journey at Carnegie Mellon University in September 2023 to pursue a master's in AI Engineering and Chemical Engineering. It was my first time stepping out of my hometown, and this experience has been nothing short of transformative. During this time, I cultivated a passion for machine learning and its applications while navigating significant personal challenges. Being diagnosed with a disorder and undergoing surgery was one of the most difficult periods of my life. I am profoundly grateful to Professor Crysanthos E. Gounaris for his unwavering support during these hardships and for inspiring my interest in applying machine learning to Vehicle Routing Problem (VRP) optimization.

I have been fortunate to collaborate with remarkable individuals in academia and industry. My heartfelt gratitude goes to Dr. Jose M. Pinto and Dr. Jose M. Lainez-Aguirre from Linde PLC for their invaluable mentorship and guidance, which deepened my understanding of real-world optimization and machine-learning applications

I owe everything to my parents. Although my father is no longer with us, his memory continues to guide me, and I hope he will be proud of how far I've come. To my mother, thank you for your unwavering belief in me and your endless support as I became the first in our family to cross the sea in pursuit of education. Your love has been my most significant source of strength.

Every ending marks the beginning of a new journey, and success is not a destination but a continuous process. My time at CMU may have been brief, but it was an incredibly enriching chapter of my life that I will always cherish. One day, I will have the privilege of being part of CMU again.

A B S T R A C T

This thesis introduces an innovative method for addressing the Capacitated Vehicle Routing Problem (CVRP) by employing a deep reinforcement learning technique. The CVRP is a fundamental challenge in logistics, involving optimizing routes for a fleet of vehicles to provide service to customers with known demands. Traditional methods often struggle with scalability, dynamic change, and adaptability, prompting the exploration of machine-learning solutions. This approach leverages a custom reinforcement learning model employing a graph-based attention mechanism to encode and decode the process dynamically. The Graph Attention Encoder uses multi-head attention layers to capture dependencies between nodes, while the multi-head attention layer with Dynamic Decoder captures complex dependencies between nodes, enabling efficient route planning. Implemented an agent-based framework, which simulates the routing environment and facilitates the training of our model through probabilistic reward-based feedback mechanisms. This approach provides a flexible framework for efficiently managing multiple vehicles in complex routing scenarios. Experimental results illustrate that the model can effectively learn optimal routing strategies, producing competitive routing solutions compared to baseline or optimal solutions. This work contributes to the growing body of research on applying deep reinforcement learning to combinatorial optimization problems, providing insights into the integration of attention networks with reinforcement learning to solve vehicle routing problems.

Keywords: Capacitated Vehicle Routing Problem (CVRP); Deep Reinforcement Learning; Graph Attention Networks (GATs); Multi-Head Attention; Dynamic Routing.

INTRODUCTION

The Vehicle Routing Problem (VRP)*(**Error! Bookmark not defined.**)* is a cornerstone challenge in logistics and supply chain management. It involves optimizing routes for vehicles to service a set of customers while minimizing costs and adhering to constraints like vehicle capacity and service time windows. Variants such as the CVRP and the Periodic VRP (PVRP) add layers of complexity with more features and constraints, requiring solutions that address multiple dimensions simultaneously. The VRP's significance extends beyond logistics, with applications in healthcare delivery, ride-sharing systems, and urban planning. However, as problem size and complexity grow, traditional optimization methods, such as exact solvers such as Mixed Integer Programming and Branch and Bound, and heuristics become computationally expensive and need help to scale effectively.[12] Traditional optimization methods struggle or sometimes fail to solve the VRPs when the instance size is larger and has more features and constraints to maintain.

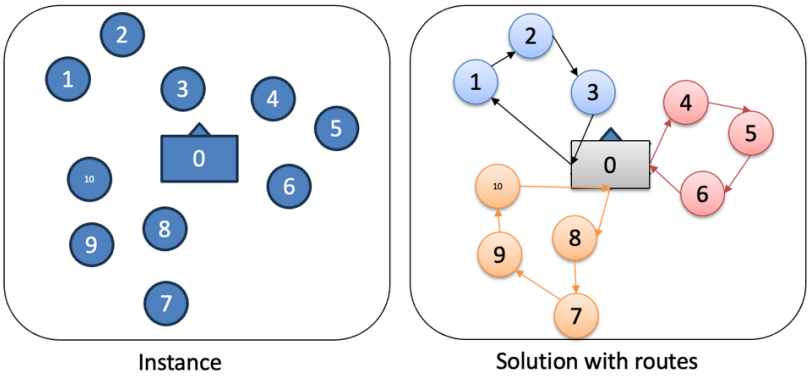

Figure 1: A common vehicle routing problem

This problem is fundamental and highly applicable, influencing the efficiency and cost-effectiveness of distribution systems globally. VRP is an optimization challenge in logistics and transportation. The task entails determining the most efficient paths/routes for a set of vehicles at the depot to provide service to a designated set of locations with selected customers. The aim is to minimize total travel expenses while complying with various constraints, such as vehicle capacity, each location is visited

once and delivery schedules. The difficulty of the VRP grows exponentially with the increase in delivery points, presenting a significant challenge for conventional exact algorithms.[3,4]

In recent years, machine learning has opened new avenues for tackling complex combinatorial optimization problems like CVRP. Among these, reinforcement learning (RL) has emerged as a promising paradigm due to its ability to learn optimal policies through interaction with an environment.[5] This thesis explores applying deep reinforcement learning techniques to solve VRPs, leveraging python's libraries such as Gymnasium, and PyTorch computational power and flexibility.[3,6]

Deep reinforcement learning (DRL) is a very optimistic approach to address complex combinatorial optimization problems. DRL combines the representational power of deep neural networks with the sequential decision-making capability of reinforcement learning, enabling agents to learn optimal or near-optimal solutions through interaction with the environment. Unlike conventional methods, which rely on domain-specific heuristics or exhaustive search, DRL learns directly from data, making it more adaptable to dynamic and uncertain scenarios.[7–9]

Recent research leverages end-to-end neural networks to directly learn heuristics from data, framing VRP as a decision-making problem solvable through RL. In this RL context, the state represents the partial solution and node features, actions are the choice of the next node, and rewards are based on minimizing tour lengths. The policy, parameterized by a neural network, is trained to maximize rewards.[3,6]

Supervised learning (SL) and RL have become prevalent in solving combinatorial optimization problems, particularly routing issues. RL models are subdivided into model-based and model-free approaches, value-based and policy-based strategies, or both, such as actor-critic frameworks. Vinyals et al.[10] introduced a supervised sequence-to-sequence pointer network (PtrNet) to train and solve Euclidean TSP and other optimization problems, employing a SoftMax distribution as a pointer to

select outputs from the input sequence. Bello et al.[11] devised an unsupervised actor-critic RL algorithm to train PtrNet, treating each instance as a sample and using tour length as a cost measure.

GCOMB, a framework utilizing graph convolutional networks and Q-learning, is designed to solve combinatorial optimization problems on large graphs. Also, models have been framed as classifiers;[12] [13]one model employs a message-passing neural network to predict satisfiability in SAT problems, while another uses a graph neural network to tackle decision-making in the Traveling Salesman Problem (TSP). Nowak et al.[14] used deep Graph Convoluted Networks (GCNs) in a supervised learning context to construct effective TSP graph representations, outputting itineraries through highly parallel beam search with non-autoregressive methods.

Several studies have demonstrated the potential of neural networks in learning heuristics for routing problems, especially in DRL and graph neural networks (GNNs), which have shown promise in automatically learning heuristics for complex problems like CVRP, bypassing the need for handcrafted feature engineering. In DRL, the CVRP can be framed as a sequential decision-making problem where the state represents the current solution and remaining customer demands, actions involve selecting the next customer to visit, and rewards are inversely related to the route's total cost. GNNs naturally capture the graph relational structures between customers and depots.[15] In GNNs, node importance is typically determined by the network structure and fixed aggregation functions like sum or mean, which may not effectively capture varying significance among nodes.[2,16] GNNs are suitable for scenarios where the graph structure is relatively uniform but can struggle when node importance varies dynamically.[17–19]

Inspired by the Transformer architecture, Kool et al.[20] proposed an attention model (AM) to solve combinatorial optimization problems, employing a rollout baseline in the policy gradient algorithm to enhance results for small-sized routing problems. Another study[21] introduced a context vector to define the decoding context, training the model with the REINFORCE algorithm and a deterministic

greedy rollout baseline. In graph convolutional networks, estimates of a node's likelihood are trained for models being part of the optimal solution, supplemented with tree search techniques to generate numerous candidate solutions. For work related to VRP, recent developments of a Graph Attention Model (GATs)[22] are particularly relevant. This model utilizes graph attention networks to address VRP challenges, seeking to improve traditional methods.

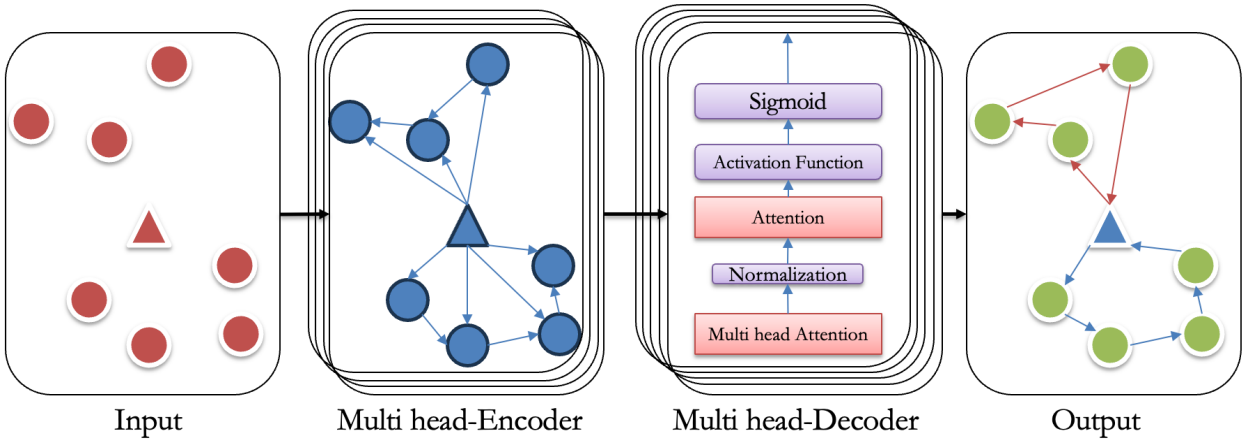

Figure 2: An end-to-end model representation

Based on survey GNNs and reinforcement learning to tackle CVRP without considering dependencies between edges in the graph structure. Inspired by the studies, our research addresses this gap by emphasizing the role of edges and incorporating residual connections between layers. This approach aims to mitigate model degradation caused by vanishing gradients in deep models, enhancing the performance of GAT *(above)*. It includes an encoder based on an improved GAT and a decoder inspired by the Transformer model[23]. Additionally, our framework employs a unique data preprocessing and feature engineering process, incorporating a bagging technique that improves accuracy. These distinctions set our model apart, contributing a novel perspective to solving optimization problems.

GRAPH ATTENTION NETWORK

The CVRP represents a significant logistical challenge in which a set of vehicles must deliver goods to multiple customer locations, constrained by its capacity. The vehicle must return to the depot as needed, ensuring effective and efficient delivery across all nodes. We need to define the problem as a graph-based optimization task. The CVRP is represented as a graph $G = (V, E)$, where, CVRP instance has $V = \{v_0, v_1, \dots, v_n\}$ be a set of nodes, where v_0 represents the depot and v_i for $i = 1, \dots, n$ represents customer locations. E is a set of edges representing possible paths between nodes. Each pair of nodes (v_i, v_j) are called spatial position of the node (s_i). These are the spatial positions of the nodes, typically represented in a 2D Euclidean space. Coordinates help calculate the distances between nodes, which are essential for determining travel costs. Demands d_i represents the quantity of goods required by each customer. The depot's demand is zero $(d_0 = 0)$, as it is only a starting and ending point for routes. The objective is to find a set of routes starting and ending at the depot that minimizes the total travel cost while satisfying vehicle capacity constraints.

$$Total\ cost = minmize \sum_{k=1}^{m} \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij}^{k} \quad (1)$$

where $x_{ij}^{k} = 1$ if vehicle k travels from node i to node j, and 0 otherwise. Constraints for each route, the total demand must not exceed vehicle capacity C *:*

$$\sum_{i=1}^{n} d_i x_{ij}^{k} \leq C, \forall k \quad (2)$$

Each customer must be visited exactly once:

$$\sum_{k=1}^{m} \sum_{j=1}^{n} x_{ij}^{k} = 1, \forall i \quad (3)$$

Equations (2) and (3) represents capacity constraints and visit constraints. The solution $\pi = (\pi_1, \dots, \pi_T)$, where $\pi_T \in \{x_0, \dots, x_n\}$, ensures that each customer node is visited once while the depot may be revisited. The sequence length T varies across solutions. VRP can be framed as a sequential decision-making problem, effectively addressed using an encoder-decoder architecture[24]. The encoder extracts structural features from the input graph, which consists of nodes (representing customer locations and the depot) and edges (representing possible routes). These features are transformed into embeddings that capture the relationships and dependencies within the graph. The decoder then uses these embeddings to construct a solution incrementally. At each step t, it predicts a distribution over nodes, selecting one node to visit based on the partial solution $\pi_{1:t-1}$ (the sequence of nodes visited so far). This process continues until a complete route is formed. The probability of constructing a complete solution π given an input instance X is modeled as follows:

$$p_\theta(\pi \mid X) = \prod_{t=1}^{T} p_\theta\left(\pi_t \mid X, \pi_{1:t-1}\right) \qquad (4)$$

From equation (4), $p_\theta(\pi \mid X)$ represents the probability of generating the entire sequence π (the route) given the input instance X, parameterized by θ. The probability of the complete solution is decomposed into a product of conditional probabilities for each step t. This reflects how each decision (node selection) depends on both the input data and the decisions made in previous steps $(\pi_{1:t-1})$.

- **Encoder**

The encoder employs a Graph Attention Network (GAT) to convert node features into embeddings, drawing a parallel to the transformer architecture utilized in natural language processing. For each node x_i a feature vector of dimension $d_x = 3$, is used, incorporating two spatial coordinates and a demand value. An initial node embedding $h_i(0)$ with a dimension of $d_h = 128$ is calculated by applying a linear transformation. This involves multiplying the node's feature vector by a learnable

weight matrix $W \in \mathbb{R}^{d_h \times d_x}$ and adding a bias vector $b \in \mathbb{R}^{d_h}$. The depot node, denoted as v_0, uses separate transformation parameters (W_0 and b_0) to account for its unique function in the routing scenario. The initial embedding for each node is computed as:[24]

$$h_i^{(0)} = \begin{cases} Wx_i + b & if\ i \neq 0 \\ W_0 x_i + b_0 & if\ i = 0 \end{cases} \quad (5)$$

These features are initially encoded into a higher-dimensional space to facilitate complex relationship modeling. The initial embeddings are processed through a series of three attention layers in the GAT. Each layer consists of two sublayers:

- **Multi-Head Attention Sublayer**

Multi-Head Attention (MHA)[23] sublayer allows the model to focus on different parts of the graph simultaneously by computing multiple attention scores across different "heads." It captures various types of information from neighboring nodes. For each layer $l \in \{1, \dots, N\}$, the node embedding $h_i(l)$ is computed for each node i. The output from layer $l-1$, denoted as $\{h_0^{(l-1)}, \dots, h_n^{(l-1)}\}$, serves as the input for layer l. The multi-head attention[22] vector $MHA_i^{(l)}\{h_0^{(l-1)}, \dots, h_n^{(l-1)}\}$ for each node i is determined through several steps. Initially, query, key, and value vectors as in equation (6) are derived for each node using respective parameters. These vectors are then used to calculate unnormalized attention scores as equation (7), which capture the similarity between nodes. The scores are normalized via the SoftMax function to obtain attention weights as equation (8). These weights are applied to the value vectors, computing a weighted sum that represents the attention output for each node as equation (9). Finally, outputs from all heads are aggregated as equation (10) to form the comprehensive multi-head attention vector, enabling the model to integrate information from different perspectives effectively.

$$q_{im}^{(l)} = W_m^Q h_i^{(l-1)};\ k_{im}^{(l)} = W_m^K h_i^{(l-1)};\ v_{im}^{(l)} = W_m^V h_i^{(l-1)}; \quad (6)$$

$$u_{ijm}^{(l)} = (q_{im}^{(l)})^T k_{jm}^{(l)}, \tag{7}$$

$$a_{ijm}^{(l)} = \frac{e^{u_{ijm}^{(l)}}}{\sum_{j'}^{n} e^{u_{ij'm}^{(l)}}}, \tag{8}$$

$$h_{im}^{\prime(l)} = \sum_{j=0}^{n} a_{ijm}^{(l)} v_{jm}^{(l)}, \tag{9}$$

$$MHA_i^{(l)}\left\{h_0^{(l-1)}, \dots, h_n^{(l-1)}\right\} = \sum_{m=1}^{M} W_m^O h_{im}^{\prime(l)} \tag{10}$$

- **Feed-Forward Sublayer**

Feed-Forward (FF) sublayer applies a fully connected neural network to each node's embedding, allowing for non-linear transformations that enhance feature representation. The update of each node's embedding involves a feed-forward sublayer using skip connections and a fully connected network.[11] Initially, an intermediate representation is computed with a tanh activation function applied to the sum of the node's previous embedding and the multi-head attention output.[23]

$$\hat{h}_i^{(l)} = tanh(h_i^{(l-1)} + MHA_i^{(l)}(h_0^{(l-1)})) \tag{11}$$

Next, this intermediate representation is passed through a two-layer feed-forward network using ReLU activation to process the node's features further.

$$FF\left(\hat{h}_i^{(l)}\right) = W_1^F \left(ReLU\left(W_0^F \hat{h}^{(l)} + b_0^F\right)\right) + b_1^F \tag{12}$$

$$h_i^{(l)} = \tanh\left(\hat{h}_i^{(l)} + FF(\hat{h}^{(l)})\right) \tag{13}$$

Finally, node embedding is updated by combining the intermediate representation and the feed-forward network output, with a tanh activation applied to the result as in equation (13).

$$h_i^N = ENCODE_i^N(h_0, \dots, h_n^0) \tag{14}$$

After processing through multiple attention layers, the final embedding for each node is determined by encoding the initial node features through this attentive mechanism, effectively capturing the complex dependencies and relationships inherent in vehicle routing contexts.[24]

- **Decoder**

Within the decoder, at each step t of constructing the solution, a node is chosen based on the current partial solution $\pi_{1:t-1}$ and the embeddings of each node. A context vector h_c is calculated using a multi-head attention mechanism. Specifically, for the VRP, a new context vector h'_c is formulated such that if $t = 1$, the vector is $h'_c = [\bar{h}_t; h_0^N; D_t]$, and if $t > 1$, it becomes $h'_c = [\bar{h}_t; h_{\pi_{1:t-1}}^N; D_t]$. Here, the operation $[;]$ denotes concatenation, $h_{\pi_{1:t-1}}^N$ is the embedding of the node from the previous step, D_t represents the vehicle's remaining capacity (initially $D_1 = D$), and $\bar{h}_t$ is the average embedding of currently unvisited nodes, including the depot.

The context vector h_c is then computed with a single multi-head attention layer, where a unique query $q_m^{(c)}$ is generated per head, using parameters distinct from the encoders. The query is defined as

$$q_{im}^{(c)} = W_m^Q h'_i, \tag{15}$$

and for each node j, key and value vectors are

$$k_{jm} = W_m^K h_j^N \text{ and } v_{jm} = W_m^V h_j^N, \tag{16}$$

Attention scores are computed for feasible nodes, setting scores to negative infinity for infeasible ones. These scores are normalized using SoftMax to yield attention weights, which are then used to compute a weighted sum of the value vectors, forming the output $h_m^{\prime(c)}$. Finally, the context vector is computed by aggregating outputs from all heads.

This architecture updates node embeddings dynamically at each construction step.[2425] If the vehicle returns to the depot, node embeddings are recalculated to account for changes in the graph's structure.

At each step t, the updated embedding for a node i is recalculated if the vehicle has reached the depot, otherwise the previous embedding is retained.[26]

DATA PREPARATION, FEATURE ENGINEERING AND MODEL TRAINING

To ensure comprehensive training and testing of the reinforcement learning (RL) model, we sourced data from VRPLIB, a repository containing standard benchmark instances for Vehicle Routing Problems (VRPs). The instances used in this study include:

- **A-n61-k9**: 61 customers and nine vehicles.
- **A-n45-k7**: 45 customers and seven vehicles.
- **A-n32-k5**: 32 customers and five vehicles

Each instance keeps the number of customers and vehicles constant. The goal is to generate 1500 data sets per instance by varying customer demands while ensuring they remain whole numbers. Customer demands are systematically varied while ensuring they remain integers and comply with vehicle capacity constraints. Data Generation and Demand Variation involves generating demands for each customer node using a random distribution within a realistic range. A function *`randomize_demands`* is designed to adjust the demands of customer nodes for each instance, ensuring the new demands are feasible (i.e., not exceeding vehicle capacity) and adding diversity to the dataset. To enhance the VRP model's robustness and generalization capability, we applied a series of data augmentation techniques to generate a diverse set of problem instances, 1500 for each.

- **Scaling Node Coordinates**

Each node's coordinates are multiplied by a randomly selected scaling factor in the range of 0.9 to 1.1. This transformation maintains the relative positioning of the nodes within the problem space, thereby preserving the structure of the VRP instance while altering the spatial scale. This will help the model

adapt to varied node densities, reflecting different urban and rural layouts without altering the problem's core structure.

- **Random Rotations**

A rotation matrix is applied to the node coordinates, rotating the entire set around the depot by a random angle between 0 and 2π. This technique ensures that the relative positions remain unchanged while the absolute orientation varies. Rotating the nodes helps the model develop invariance to orientation changes and prevents overfitting to a specific node.

- **Mirroring Node Locations**

Node coordinates are reflected across one or both axes to create mirrored instances. This operation changes the spatial configuration without affecting distances, effectively doubling the variety of scenarios. Mirroring extends the dataset by introducing alternate perspectives without altering the problem constraints, helping to ensure robust learning.

We applied scaling to the node coordinates and precomputed pairwise distances between customers and depots to enhance efficiency. Additionally, customers were grouped based on their geographic proximity to simplify the initial phase of route planning. Demand density was analyzed as the ratio of demand to the size of the customer cluster.

In the training process, a boosting step is introduced to enhance the model's learning efficiency and effectiveness. This step involves adaptive sampling, where the training data is dynamically adjusted to highlight complex instances, thereby broadening the model's exposure to diverse problem settings. The policy gradient method, specifically the REINFORCE algorithm[8], aims to optimize the parameters θ of a policy by minimizing the expected tour length $L(\pi \mid X)$. The gradient of the objective function $J(\theta \mid X)$ is given by:

$$\nabla_\theta J(\theta \mid X) = E_{\pi \sim p_\theta(\cdot|X)}[(L(\pi \mid X) - b(X))\nabla_\theta log p_\theta(\pi \mid X)] \quad (17)$$

where $L(\pi \mid X)$ is the tour length, $b(X)$ is a baseline function used to reduce variance, and $p_\theta(\pi \mid X)$ is the probability of selecting a path π given the input X.

Implementation plan and description for the boosting step:

- Adaptive Sampling: Dynamically adjust the distribution of training data to focus more on difficult instances, effectively changing the data distribution $p(difficult\ instances)$ within the sampling process. This can be modeled by adjusting the weight of instances $w_i \propto \frac{1}{current_{performance(i)}}$
- Ensemble Learning: Create multiple models $\theta_1, \theta_2, \ldots, \theta_K$ and combine their outputs to form a robust ensemble policy $p_{ensemble}(\pi \mid X) = \frac{1}{K}\sum_{k=1}^{K} p_{\theta_k}(\pi \mid X)$ The gradient for each model would be adjusted based on collaborative error reduction.
- Gradient Boosting: Iteratively update the model by combining the gradients from previous iterations to focus on the errors of past models. This can be captured by,

$$\theta^{(t+1)} = \theta^{(t)} - \alpha(\nabla_\theta J(\theta^{(t)} \mid X) + \lambda \sum_{s=1}^{t-1}(\theta^{(t)} - \theta^{(s)}(s)) \qquad (18)$$

  Here, α is the learning rate, and λ is a hyperparameter controlling the influence of previous iterations.

Gradient boosting directs the training focus on rectifying past errors by aggregating historical gradient information. These combined techniques aim to strengthen the policy's ability to minimize tour lengths efficiently in various CVRP scenarios.

EXPERIMENTS

We conducted a series of experiments to gauge the effectiveness of our model. For the conducted experiments, data was sourced from VRPLIB, including instances A-n61-k9, A-n45-k7, and A-n32-k5. A total of 1500 datasets were generated per instance; varying customer demands while preserving problem constraints. This data was divided as follows: 20% was used as development data for hyperparameter tuning and model validation during training, 10% was reserved as test data for final performance evaluation, and the remaining 70% was allocated for nested cross-validation to optimize model parameters.

Nested cross-validation was employed to ensure robust parameter selection and to prevent overfitting. This involved an outer loop where the cross-validation data was divided into $k = 5$ folds. In each iteration, one-fold was reserved for validation, while the remaining $k - 1$ folds were used for training. Within this framework, an inner loop was conducted for additional parameter tuning within the training folds, allowing for the selection of the best-performing hyperparameters. This approach ensured that the validation process in the outer loop remained unbiased.

In configuring the training setups for different instances of the problem, specific strategies were adopted to optimize performance. For Instance, A-n61-k9, experiments with various batch sizes revealed that a batch size of 64 struck the best balance between computational efficiency and model convergence. Smaller batch sizes resulted in slower convergence, while larger ones led to overfitting. The Adam optimizer was chosen for its adaptive learning rate capabilities, which are particularly beneficial in reinforcement learning scenarios where gradients can be sparse. For Instance A-n45-k7, a batch size of 64 was again selected based on empirical results and the complexity of the problem, as it provided effective gradient updates and stable convergence. A learning rate of 10^{-4} was found to be optimal, allowing for a steady decrease in the loss function without causing overshooting. In the case

of Instance A-n32-k5, the smaller problem size enabled the use of a batch size of 32, which balanced computational demands with training stability. A slightly higher learning rate of 2×10^{-4} was employed to accelerate convergence, taking advantage of the reduced complexity in the dataset. These tailored configurations ensured efficient training and robust model performance across different problem instances.

For learning rates, a range from 10^{-5} to 10^{-3} was tested to find the most effective settings. A learning rate of 10^{-4} provided optimal results for larger instances such as A-n61-k9 and A-n45-k7, ensuring stability during training. In contrast, a slightly higher learning rate of 2×10^{-4} was effective for smaller instances like A-n32-k5, where it helped accelerate convergence due to the simpler problem complexity. The training process was structured in batches of 100 epochs, repeated for a total of 600 epochs per instance. This iterative approach allowed for periodic evaluation of the model's performance and ensured its ability to generalize across unseen data.

A-n61-k9	Method	Length/Cost	Training time (hh:mm:ss)	Test time (hh:mm:ss)	Gap
	RL-GAT	1142	13:35:00	00:01:04	5.30%
	OR	1123	-	-	3.60%
	LKH	1093	15:12:00	03:20:00	0.91%
	OPT	1084	-	-	0.00%
A-n45-k7	RL-GAT	1223	07:38:00	00:00:36	6.68%
	OR	1215	-	-	6.03%
	LKH	1160	08:10:00	01:13:48	1.20%
	OPT	1146	-	-	0.00%
A-n32-k5	RL-GAT	797	04:28:00	00:00:05	2.60%
	OR	792	-	-	0.98%
	LKH	789	03:10:00	00:20:23	0.58%
	OPT	784	-	-	0.00%

Table 1:Performance evaluation

RESULTS

We tested our results against baseline optimal solutions, Google OR-tools, and the LKH Solver to evaluate the effectiveness of our approach in solving the Capacitated Vehicle Routing Problem (CVRP). The analysis focused on three instances: A-n61-k9, A-n45-k7, and A-n32-k5.

From Table 1 the training and validation loss curves for Instance A-n61-k9(*below*) demonstrate a significant decrease in loss during the initial epochs, indicating effective learning and adaptation by the model. The training loss stabilizes around epoch 400, which is marked as the optimal point for generalization. The validation loss follows a similar trend but with slightly more variability, reflecting the model's ability to adapt to unseen data. The RL-GAT method achieved a route length of 1141.45 with a gap of 5.30% from the optimal solution, requiring 13 hours and 35 minutes of training and 1 minute 4 seconds for testing. In comparison, LKH produced a closer approximation to the optimal length with a gap of 0.91%, highlighting its efficiency despite longer testing times.

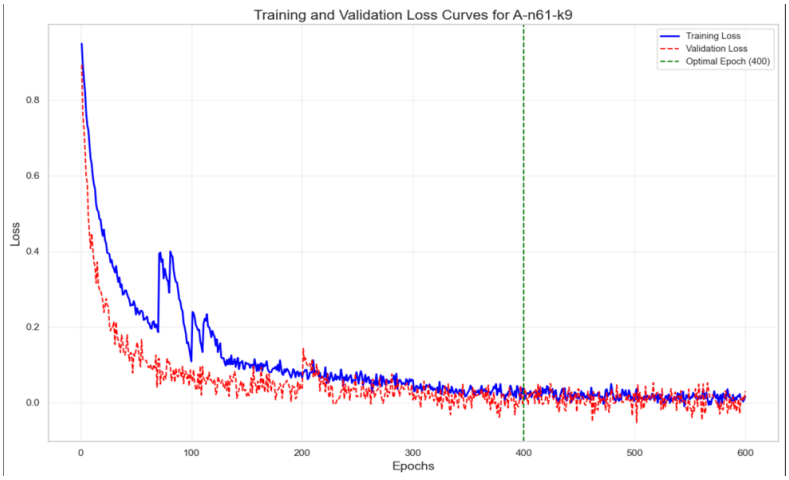

Figure 3:Loss curves for A-n61-k9

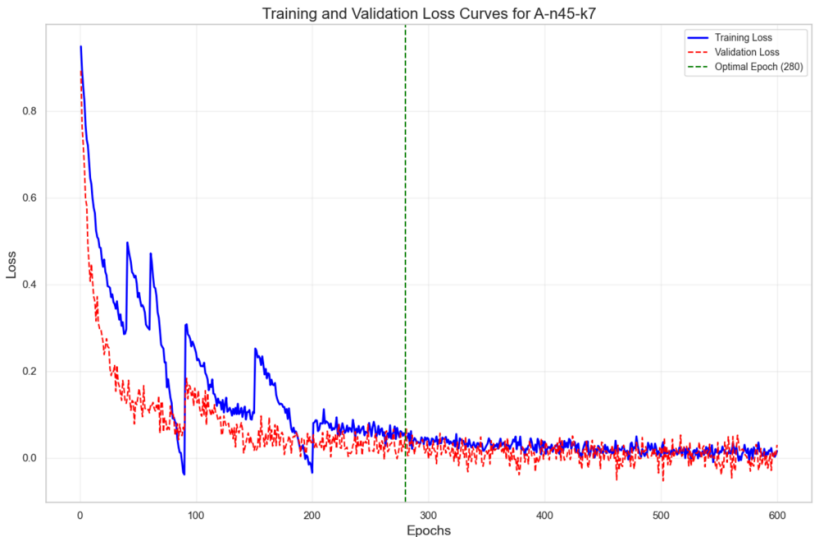

Figure 4:Loss curves for A-n45-k7

For Instance, A-n45-k7(*above*), the loss curves indicate a steady reduction in both training and validation losses, with convergence occurring around epoch 280. This suggests that the model effectively captures the underlying patterns necessary for optimization in this instance. The RL-GAT approach resulted in a route length of 1222.56, with a gap of 6.68%, after 7 hours 38 minutes of training and a quick 36-second test time. LKH again demonstrated superior performance with a significantly lower gap of 1.20%, albeit with longer training and testing durations. These results underscore the trade-offs between computational efficiency and solution accuracy.

In Instance A-n32-k5*(below)*, the model shows rapid convergence with both training and validation losses decreasing quickly and stabilizing around epoch 180. This rapid stabilization is indicative of the simpler problem complexity compared to larger instances. The RL-GAT method achieved a route length of 797.33 with a gap of 2.60%, requiring 4 hours 28 minutes for training and an extended test time of 5 seconds due to its complexity handling smaller datasets. LKH provided a closer approximation to the optimal solution with a gap of only 0.58%, demonstrating its robustness and efficiency in smaller-scale problems.

Figure 5:Loss curves for A-n32-k5

The LKH Solver consistently demonstrated superior performance in terms of proximity to the optimal solution across all instances, albeit with longer computational times compared to RL-GAT and OR-tools. RL-GAT offered faster test times but struggled to achieve the precision seen with LKH, particularly in larger instances like A-n61-k9 and A-n45-k7. OR-tools served as an effective middle ground but did not match the accuracy levels attained by LKH or the optimal solutions.

The performance of our RL-GAT model was evaluated on various CVRP benchmark instances sourced from VRPLIB. Table 2 summarizes the results, including the number of nodes, vehicle capacity, routes, optimal solutions, and the gap between our RL-GAT model and the optimal solutions. The RL-GAT model achieved competitive results across all instances, with optimal gaps ranging from 0.84% to 12.76%. For smaller instances such as A-n32-k5 (31 nodes), the gap was minimal at 0.84%, indicating that the model performs well on less complex problems. However, for larger instances like A-n64-k9 (63 nodes), the gap increased to 12.76%, suggesting potential challenges in scaling to more complex scenarios.

SL.no.	Problem	Nodes	Capacity	Routes	Optimal	Opt. Gap (%)	Output Value
0	A-n32-k5	31	100	5	784.0	0.84	790.5
1	A-n34-k5	33	100	5	778.0	5.20	818.4
2	A-n37-k5	36	100	5	669.0	3.50	692.4
3	A-n38-k5	37	100	5	730.0	2.70	749.7
4	A-n39-k5	38	100	5	822.0	1.63	835.3
5	A-n39-k6	38	100	6	831.0	8.56	902.1
6	A-n44-k6	43	100	6	937.0	7.31	1005.5
7	A-n45-k6	44	100	6	944.0	4.60	987.4
8	A-n45-k7	44	100	7	1146.0	10.67	1268.3
9	A-n46-k7	45	100	7	914.0	8.20	988.9
10	A-n48-k7	47	100	7	1073.0	3.65	1112.1
11	A-n53-k7	52	100	7	1010.0	2.98	1040.1
12	A-n64-k9	63	100	9	1401.0	12.76	1579.8
13	A-n65-k9	64	100	9	1174.0	8.80	1277.3
14	A-n69-k9	68	100	9	1159.0	10.25	1277.8
15	A-n63-k10	62	100	10	1314.0	10.40	1450.7

Table 2: Trained instance A for 70 nodes and solutions

The model generalizes effectively across different problem sizes but struggles with maintaining route efficiency when node count increases significantly. The RL-GAT model scales well for small to medium-sized problems but exhibits performance degradation as problem complexity increases. This is evident from the increasing gaps for larger instances like A-n64-k9 (12.76%) and A-n63-k10

(10.40%). While RL-GAT achieves near-optimal solutions for smaller instances, traditional methods like LKH or OR-Tools, but still outperform traditional methods in terms of achieving exact optimality for larger datasets.

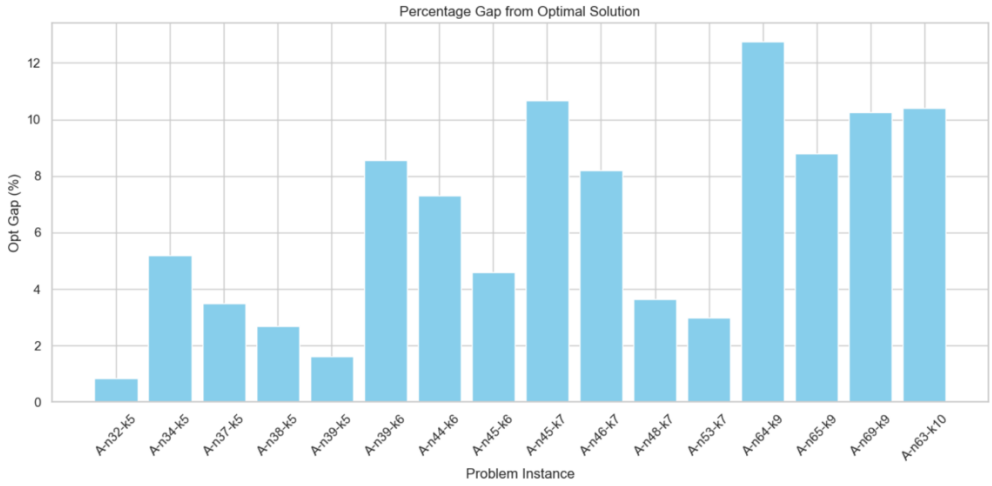

Figure 6:Error Box Plot (Percentage Gap Across Instances)

The error box plot(*above)* highlights the variability in the percentage gap across problem instances. For smaller problems like A-n32-k5 and A-n38-k5, the RL-GAT model achieves low gaps (0.84% and 2.70%, respectively), indicating strong performance on simpler datasets. However, as the problem size increases, such as in A-n64-k9 and A-n63-k10, the gap widens significantly to 12.76% and 10.40%, reflecting challenges in scalability and handling complexity. The spread in gaps is more consistent for smaller instances, while larger datasets exhibit greater variability, suggesting room for improvement in managing complex scenarios.

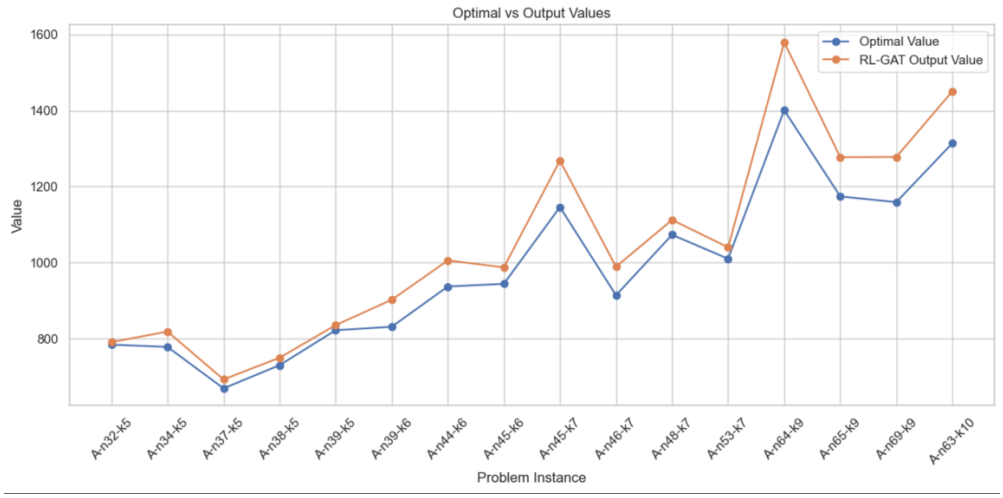

Figure 7: Line Plot (Optimal vs RL-GAT Output Values)

The line plot*(above)*comparing optimal values with RL-GAT output values further emphasizes these challenges. For smaller instances like A-n32-k5 and A-n38-k5, the RL-GAT output values closely align with optimal solutions, demonstrating near-optimal performance. However, for larger instances such as A-n64-k9 and A-n63-k10, significant deviations are observed between RL-GAT outputs and optimal values, highlighting inefficiencies in route optimization. These findings says that while RL-GAT performs well on small to medium-sized problems, additional optimization mechanisms are needed to improve its scalability and generalization for larger datasets.

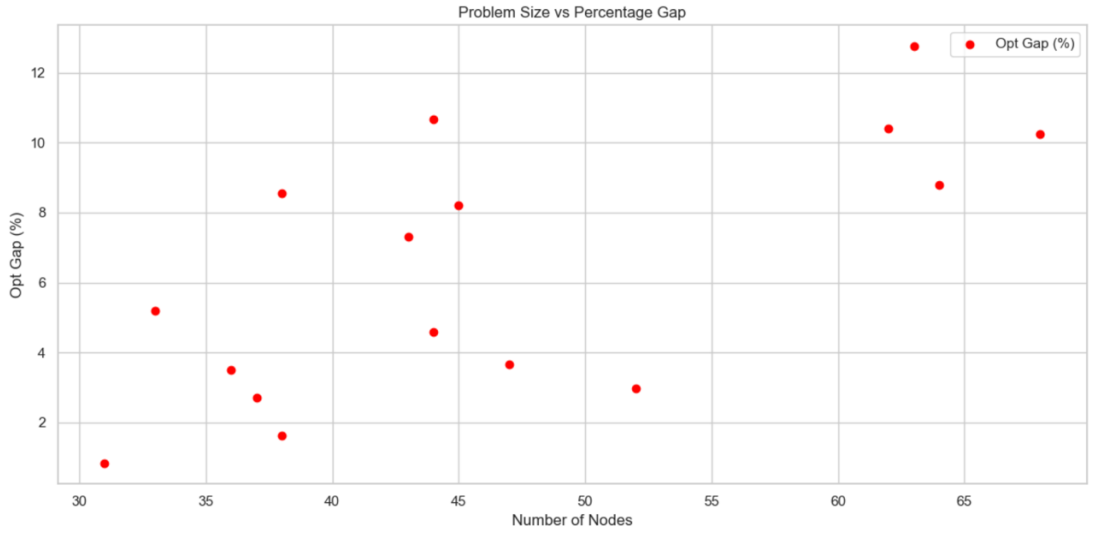

Figure 8: Scatter Plot (Problem Size vs Percentage Gap)

The scatter plot*(above)* demonstrates a clear upward trend in the gap percentage as the number of nodes increases, confirming that problem complexity negatively impacts the RL-GAT model's performance. For problems with fewer than 40 nodes, the gaps remain below 5%, showing that RL-GAT is effective for small-scale CVRP instances. However, for larger problems with more than 60 nodes, gaps exceed 10%, indicating that the model struggles to maintain efficiency and solution quality as problem size grows. This trend highlights the need for scalability improvements to address larger datasets effectively.

CONCLUSION

The experiments conducted in this study demonstrate the potential of reinforcement learning with Graph Attention Networks (RL-GAT) to address the Capacitated Vehicle Routing Problem (CVRP). By leveraging benchmark datasets from VRPLIB and employing tailored configurations for each instance, the RL-GAT model achieved competitive results across varying problem complexities. For smaller instances such as A-n32-k5, the model exhibited minimal gaps from the optimal solutions (0.84%), highlighting its efficiency in handling fewer complex scenarios. However, as the problem size increased, such as in A-n64-k9, the gap widened to 12.76%, indicating challenges in scalability and route efficiency for larger datasets.

The RL-GAT model's ability to generalize across different problem sizes was evident from its consistent performance on medium-sized instances like A-n45-k7, where it achieved a gap of 6.68% with significantly reduced test times compared to traditional solvers like LKH. Despite these advantages in computational efficiency, the model's precision lagged behind LKH, which consistently produced near-optimal solutions with minimal gaps. This trade-off between computational speed and solution accuracy underscores the need for further optimization of RL-GAT to improve its scalability and precision for larger instances.

Future research should explore enhancements to the RL-GAT architecture, such as incorporating hierarchical attention mechanisms or more sophisticated data augmentation techniques to improve scalability and generalization. Additionally, integrating ensemble learning or boosting techniques could further refine the model's ability to handle complex CVRP scenarios. These advancements have the potential to make RL-GAT a more robust and adaptable solution for real-world applications in logistics and supply chain management.

BIBLIOGRAPHY

(1) Raza, S. M.; Sajid, M.; Singh, J. Vehicle Routing Problem Using Reinforcement Learning: Recent Advancements. In *Advanced machine intelligence and signal processing*; Springer, 2022; pp 269–280.

(2) Bahovska, E. Graph Neural Networks in Neighbourhood Selection for a Vehicle Routing Problem Solver. Artificial Intelligence, Utrecht University, 2023.

(3) Kovács, L.; Jlidi, A. Neural Networks for Vehicle Routing Problem. *arXiv preprint arXiv:2409.11290* **2024**.

(4) Wang, A. Optimization Algorithms for Vehicle Routing and Packing Problems. PhD, Carnegie Mellon University, Pittsburgh, 2020.

(5) Nazari, M.; Oroojlooy, A.; Snyder, L. V.; Takáč, M. Reinforcement Learning for Solving the Vehicle Routing Problem. **2018**.

(6) Czuba, P.; Pierzchala, D. Machine Learning Methods for Solving Vehicle Routing Problems. *Proceedings of the 36th International Business Information Management Association (IBIMA), Granada, Spain* **2021**, 4–5.

(7) Pan, W.; Liu, S. Q. Deep Reinforcement Learning for the Dynamic and Uncertain Vehicle Routing Problem. *Applied Intelligence* **2023**, *53* (1), 405–422.

(8) Kalakanti, A. K.; Verma, S.; Paul, T.; Yoshida, T. RL SolVeR pro: Reinforcement Learning for Solving Vehicle Routing Problem. In *2019 1st international conference on artificial intelligence and data sciences (AiDAS)*; IEEE, 2019; pp 94–99.

(9) Chen, B.; Qu, R.; Bai, R.; Laesanklang, W. *A Reinforcement Learning Based Variable Neighborhood Search Algorithm for Open Periodic Vehicle Routing Problem with Time Windows*.

(10) Vinyals, O.; Brain, G.; Fortunato, M.; Jaitly, N. *Pointer Networks*.

(11) Bello, I.; Pham, H.; Le, Q. V; Norouzi, M.; Bengio, S. Neural Combinatorial Optimization with Reinforcement Learning. *arXiv preprint arXiv:1611.09940* **2016**.

(12) Joshi, C. K.; Laurent, T.; Bresson, X. An Efficient Graph Convolutional Network Technique for the Travelling Salesman Problem. *arXiv preprint arXiv:1906.01227* **2019**.

(13) Shehzad, A.; Xia, F.; Abid, S.; Peng, C.; Yu, S.; Zhang, D.; Verspoor, K. Graph Transformers: A Survey. *arXiv preprint arXiv:2407.09777* **2024**.

(14) Nowak, A.; Villar, S.; Bandeira, A. S.; Bruna, J. A Note on Learning Algorithms for Quadratic Assignment with Graph Neural Networks. *Stat* **2017**, *1050*, 22.

(15) Tien, Z. C.; Qi-lee, J. Enhancing Vehicle Routing Problem Solutions through Deep Reinforcement Learning and Graph Neural Networks. *International Journal of Enterprise Modelling* **2022**, *16* (3), 125–135.

(16) Joshi, C. K.; Laurent, T.; Bresson, X. An Efficient Graph Convolutional Network Technique for the Travelling Salesman Problem. **2019**.

(17) Busbridge, D.; Sherburn, D.; Cavallo, P.; Hammerla, N. Y. Relational Graph Attention Networks. *arXiv preprint arXiv:1904.05811* **2019**.

(18) Kool, W.; Van Hoof, H.; Welling, M. Attention, Learn to Solve Routing Problems! *arXiv preprint arXiv:1803.08475* **2018**.

(19) Fellek, G.; Farid, A.; Gebreyesus, G.; Fujimura, S.; Yoshie, O. Graph Transformer with Reinforcement Learning for Vehicle Routing Problem. *IEEJ Transactions on Electrical and Electronic Engineering* **2023**, *18* (5), 701–713.

(20) Kool, W.; van Hoof, H.; Gromicho, J.; Welling, M. Deep Policy Dynamic Programming for Vehicle Routing Problems. In *International conference on integration of constraint programming, artificial intelligence, and operations research*; Springer, 2022; pp 190–213.

(21) Fellek, G.; Farid, A.; Gebreyesus, G.; Fujimura, S.; Yoshie, O. Graph Transformer with Reinforcement Learning for Vehicle Routing Problem. *IEEJ Transactions on Electrical and Electronic Engineering* **2023**, *18* (5), 701–713.

(22) Vaswani, A.; Brain, G.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; Polosukhin, I. *Attention Is All You Need*.

(23) Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. **2017**.

(24) Peng, B.; Wang, J.; Zhang, Z. A Deep Reinforcement Learning Algorithm Using Dynamic Attention Model for Vehicle Routing Problems. In *Artificial Intelligence Algorithms and Applications: 11th International Symposium, ISICA 2019, Guangzhou, China, November 16–17, 2019, Revised Selected Papers 11*; Springer, 2020; pp 636–650.

(25) Lei, K.; Guo, P.; Wang, Y.; Wu, X.; Zhao, W. Solve Routing Problems with a Residual Edge-Graph Attention Neural Network. *Neurocomputing* **2022**, *508*, 79–98.

(26) Wang, Z.; Bai, R.; Khan, F.; Özcan, E.; Zhang, T. Gase: Graph Attention Sampling with Edges Fusion for Solving Vehicle Routing Problems. *Memet Comput* **2024**, *16* (3), 337–353.