DEEP LEARNING-DRIVEN METAHEURISTIC FOR ENHANCING VEHICLE ROUTING SOLUTIONS

Georgios TZAGKARAKIS[1]
Theocharis METZIDAKIS[2]
Manolis KRITIKOS[3]

Abstract
This paper investigates the integration of machine learning techniques with classical optimization methodologies to address the Capacitated Vehicle Routing Problem (CVRP), a well-known NP-hard problem in logistics. Traditional approaches leverage heuristics and metaheuristics, such as GRASP and VND, but often struggle with adaptability and efficiency at large scale. Recent advancements in deep learning offer new opportunities for enhanced solution quality and computational efficiency. This study proposes a novel hybrid methodology combining Greedy Randomized Adaptive Search Procedure (GRASP), Variable Neighborhood Descent (VND), and a Graph Isomorphism Network with Edge Features (GINE). The GINE model classifies initial solutions as promising or unpromising, enabling selective local search refinement and potential computational savings. Empirical evaluation on benchmark CVRP instances demonstrates that the hybrid GINE-GRASP-VND approach achieves moderate classification accuracy, generalizes beyond training sizes, and shows stronger improvements on large problems. Results suggest the value of machine learning augmentation in metaheuristics but highlight the challenges of feature design and generalization.

Keywords: vehicle routing, machine learning, metaheuristics, graph neural networks, combinatorial optimization, logistics, hybrid algorithms, deep learning

JEL Classification: C61

1. Introduction

Operations research (OR) provides mathematical models for real-world logistics problems, making use of heuristics, metaheuristics, and, more recently, machine learning (ML) for improved solution quality. A rapidly evolving field in OR is the integration of ML and artificial intelligence (AI) with classical analytical techniques [1]. This synergy is transforming operational efficiency and enabling more strategic planning, as modern algorithms enhance both problem-solving capabilities and data-driven decision-making in

[1] Athens University of Economics and Business, geotzag1997@gmail.com
[2] Management Science Laboratory, Athens University of Economics and Business, metzidakist@gmail.gr, corresponding author
[3] Management Science Laboratory, Athens University of Economics and Business, kmn@aueb.gr

dynamic business environments. ML algorithms are well known for their pattern recognition on data that have no clear mathematical formulation (e.g. images, text and voice). Similarly, combinatorial optimization (CO) tasks involving graph-based data are well-suited to leverage ML methodologies, supporting advanced solution strategies for challenging real-world problems [2].

The Vehicle Routing Problem (VRP) is one of the most fundamental and widely studied combinatorial optimization problems in OR and Logistics. It is concerned with the design of an optimal set of routes between geographically scattered customers that require service and a depot with a fleet of vehicles [3]. Depending on the constraints of the problem (e.g. capacities, time windows, multi-depots, etc.) multiple variants of VRP exist [4]. This paper focuses on the Capacitated Vehicle Routing Problem (CVRP), since they provide a solid start for implementing a new method.

VRPs – being NP-hard problems –traditionally have been tackled using three categories of approaches: exact algorithms, approximation algorithms and heuristics/metaheuristics. An exact method guarantees to find the optimal solution to a CO problem. While being effective, they require exponential computational power in large problems, making them unsuitable to be used. Heuristics and metaheuristics are problem-solving strategies designed to find good solutions quickly, but without guarantees of how close they are to the optimum. Among these approaches, Greedy Randomized Adaptive Search Procedures (GRASP) and Variable Neighborhood Descent (VND), often used within a Variable Neighborhood Search (VNS) framework, have emerged as particularly effective. GRASP is an iterative multi-start metaheuristic, where each iteration consists of a randomized greedy construction phase followed by a local search phase that attempts to improve the constructed solution. VND, on the other hand, is a systematic local search strategy that explores a sequence of neighborhoods, moving to a new solution whenever an improvement is found and restarting the sequence of neighborhoods [5]. The combination of GRASP with VND has proven very competitive for routing problems, as it couples diversified solution construction with an intensive and structured local improvement phase. However, the repeated application of local search to many initial solutions can be computationally demanding, which opens the door to intelligent mechanisms that decide where to invest search effort.

Most recent approaches for facing VRP involve enhancing traditional approximation methods with ML. In end-to-end learning approaches either Supervised Learning (SL) or Reinforcement Learning (RL) is employed to address the problem comprehensively, from initial formulation to final solution. In contrast, hybrid approaches incorporate learning methods either as principal mechanism for generating feasible and efficient solutions – subsequently refined though construction heuristics – or as auxiliary components that assist in resolving subproblems within conventional optimization frameworks [6]. In this paper a

hybrid approach will be implemented between a well-known metaheuristic named GRASP and GINE – an SL method that belongs to the family of Graph Neural Networks (GNNs).

GNNs have emerged as a pivotal paradigm within DL as the generalization of an attention mechanism for the graph domain, designed to operate directly on non-Euclidean data structures where relationships among entities are naturally represented as graphs. Unlike traditional ML models that assume data points are independent and identically distributed, GNNs explicitly model interdependencies between nodes through edges, making them suitable for a wide array of applications from chemistry to transportation systems. The roots of GNNs trace back to recursive neural networks in the 1990s, which were first applied to directed acyclic graphs [7]. Afterwards RNNs and feedforward neural networks introduced for processing arbitrary graph-structured data. These early models relied on iterative message-passing until convergence, which limited scalability. The breakthrough came with the integration of convolutional principles into graph domains, leading to Graph Convolutional Networks (GCNs) that aggregate local neighborhood information in a feedforward manner. This shift not only addressed efficiency but also aligned GNNs with the successful convolutional architectures of image analysis.

Over time, several variants of GNN architectures have emerged [7]. Recurrent GNNs (RecGNNs) represent the earliest family, where node representations are iteratively updated until convergence. GCNs, both spectral-based and spatial-based, dominate current practice thanks to their efficiency and scalability. Graph Attention Networks (GATs) introduce an attention mechanism to weight neighbor contributions adaptively, while Gated Attention Networks (GaANs) refine this idea by incorporating gating mechanisms that regulate multiple attention heads, improving stability and interpretability in complex graph scenarios. Graph Auto-Encoders (GAEs) extend this paradigm for unsupervised representation learning, focusing on link prediction and graph generation. Spatial–Temporal GNNs (STGNNs) add a temporal dimension to capture dynamic interactions in time-evolving systems, with direct relevance to applications such as traffic forecasting and dynamic routing. Applications of GNNs are diverse and extend across both structural and non-structural domains. Importantly, GNNs have begun to transform CO, where problems such as the TSP and the VRP can be modeled as graphs of customers and routes [6]. Despite their versatility, GNNs still face significant challenges [6]. Current research increasingly explores hybridization with reinforcement learning and metaheuristic strategies, particularly in the optimization of vehicle routing and logistics operations.

The remainder of this paper is structured as follows. Section 2 presents the proposed methodology and its main components. Section 3 reports and analyzes the computational results. Section 4 concludes the paper and outlines directions for future research.

2. Proposed methodology

In this section, we present the proposed deep learning approach. More specifically a simple hybrid of GRASP with BVND will be enhanced with a GINE model, which is a better version of GIN, since it adds an edge encoder to each message passing layer. I n the next subchapters each component of the scheme is explained.

2.1 A Hybrid Metaheuristic

Taking inspiration from [8] a GRASP procedure was hybridized with a BVND. The GRASP procedure following the quality-based scheme for the RCL – i.e. selecting elements based on travel cost c_{ij} between two nodes/customers i and j where $c_{min} \leq c_{ij} \leq c_{min} + \alpha(c_{max} - c_{min})$ – will perform a greedy randomized construction. In other words, the method starts from the depot and assigns a route towards a node/customer that is selected randomly from the created RCL list. For the CVRP problem the feasibility of the specific selection is evaluated with capacity constraint. The procedure iteratively assigns customers to a route until the vehicle has no capacity left, so it returns to the depot. That is a closed loop. Then, it starts again with a new vehicle. All customers must be serviced and passed only once from a vehicle. When that happens the total cost of the solution – called initial solution – is computed.

It should be noted that the majority of VRP problems have a 2D Euclidean edge weight type, which means the cost of (undirected) edges are computed by the Euclidean distance of their corresponding nodes. The nodes of a VRP problem are projected in a Cartesian plane with x- and y-axis with only positive values between 0 to 1 or 0 to 100. So, for two nodes i, j with coordinates (x_i, y_i) and (x_j, y_j) that are connected, the distance/cost is

$$c(i,j) = floor\left(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} + 0.5\right)$$

where floor rounds down the value to the nearest integer.

When the construction phase ends the initial solution is improved with a local search algorithm. A BVND was selected, which improves the solution by searching firstly the intra-route neighborhoods – i.e. changes in a route – and then inter-route neighborhoods – i.e. changes between routes. The procedure uses the 2-opt, swap and relocate for intra-route (with that order) and 1-0 exchange (moving a customer from one route to another) and 1-1 exchange (switching customers between routes) for inter-route while respecting the capacity constraint. To be more precise the sequence follows this pattern:

1. Intra-route VND: It runs 2-opt to convergence, then swap and then relocate. If anyone improves it keeps applying the same operator, then moves to the next operator. If there is no further improvement, then it follows the inter-route operators.

2. Inter-route VND: 1-0 exchange is applied with first improvement meaning that as soon as the cost is improved it restarts the specific inter-route operator. If there is no further improvement, then 1-1 exchange follows with the same procedure.
3. After local-search loop, if the total cost is improved, the entire cycle is repeated. Otherwise, it stops.

When the local-search loop breaks the solution is saved. Then the algorithm starts again with the construction of a new solution. If that solution is better, it is the new best solution. The procedure stops when exceeding the maximum iterations or time limit.

2.2 A Hybrid ML Approach

The concept of ISC introduced in [8] is a hybrid approach that predicts if a local-search algorithm should be applied to initial solutions constructed from a greedy algorithm like GRASP. That paper used a Random Forest as an ML classifier.

In this paper a GNN will be used as a deep learning classifier, named edge-aware GIN - also called GINE – which will be trained with several features extracted from graphs of initial solutions. GINE incorporates edge features e_{uv} into the aggregation procedure by projecting them to the node hidden size either by a learned linear layer or an edge encoder. The linear projection that occurs is from a basic MLP. Thus, the new message passing update is expressed as:

$$h_v^{(l+1)} = MLP^{(l)}\left(\left(1+\epsilon^{(l)}\right)\cdot h_v^{(l)} + \sum_{u\in\mathcal{N}(v)}\left(h_u^{(l)} + edge\_MLP^{(l)}(e_{uv})\right)\right)$$

The GRASP-VND metaheuristic will be enhanced with the GINE trained model. Then for each initial solution constructed the model will predict if the local search phase can run otherwise start the initial construction again. In that case some solutions will pass the classification and other will be denied.

2.3 Model Architecture

At the input stage, the model expects graphs, which include node features, edge indices, edge features, and a batch index to support mini-batch training. Before any graph convolutions are applied, node features are projected into a hidden-dimensional representation through a linear transformation. This ensures that the input is embedded into a common latent space that subsequent layers can process effectively. Let $X \in \mathbb{R}^{N\times node\_dim}$ the input data where N is the number of nodes and *node_dim* the number of features per node. Before the convolutions it is transformed into $H^{(0)} \in \mathbb{R}^{N\times hidden\_size}$.

The main body of the network consists of stacked GINE convolutional layers, each parameterized by a small MLP. This MLP follows the structure of a linear transformation, a ReLU nonlinearity, and another linear transformation, thereby allowing the convolution to learn complex transformations of messages passed between nodes. Each GINE layer updates node embeddings by combining information from neighboring nodes and the edges connecting them, which distinguishes it from simpler graph convolutions that only rely on node features. After every convolution, the model applies batch normalization to stabilize training, a ReLU activation to introduce non-linearity, and dropout to regularize and prevent overfitting. By repeating this sequence across multiple layers, the model incrementally enriches node representations with higher-order structural and feature-based information from the graph. Let $e \in \mathbb{R}^{E \times edge\_dim}$ the edge embeddings where E is the number of edges and *edge_dim* the number of features per edge. Then the edges are linearly transformed to $e' \in \mathbb{R}^{E \times hidden\_size}$ to match the dimensions of node embeddings. For a single node its embedding $h_0^{(0)}$ is updated by the message passing layer. More specifically the node embeddings of neighbors of a single node – which are defined by different edge structures – are summed with their corresponding projected edge features (the ones that are connected with the 0 node in the example). Then the sum of all nodes is passed through ReLU and then the message is added to the $h_0^{(0)}$, which is scaled by epsilon ϵ. The new node embedding is passed to a linear transformation *hidden_size →hidden_size,* ReLU and finally another linear transformation *hidden_size →hidden_size.* That's the convolution procedure. A Batch Normalization, another ReLU and Dropout Layer follow that give an embedding passed by one layer $H^{(1)} \in \mathbb{R}^{N \times hidden\_size}$. That means the dimension doesn't change between stacked GINEconv layers.

Once the node embeddings have been updated, they are aggregated into a single graph-level embedding through global mean pooling. This operation compresses all node-level information into a single vector for each graph in the batch. Graph-level attributes are concatenated with the pooled embedding. Let $G \in \mathbb{R}^{global\_dim}$ be the global features where *global_dim* is the number of global features. The single concatenated vector after L layers is as follows

$$g = CONCATENATE\left(\frac{1}{N}\sum_{i=1}^{N} h_i^{(L)}, G\right), g \in \mathbb{R}^{hidden\_size+global\_dim}$$

The resulting graph representation, enhanced by graph-level attributes, is then passed to a final linear readout layer. This layer produces class logits of dimension equal to the number of target classes (2), making the architecture suitable for supervised classification at the graph level.

2.4 Model Training

The neural network requires labeled training data that will try to approximate them. These data are initial solutions generated from a GRASP procedure, which are labeled based on being promising or unpromising solutions for implementing local search algorithm. The labeled scheme can only be addressed by an iterative GRASP procedure. In each iteration the initial solution is saved. Then the VND proceeds with the local search phase and produces an improved solution whose improved cost is also stored with the corresponding initial solution. After some iterations the procedure stops. The improved costs are used to label the initial solutions as promising (1) and unpromising (0). A percentage λ_p of solutions with the lowest improved cost are labeled 1 and a percentage λ_u of solutions with the highest improved cost are labeled 0. The remaining solutions are discarded, so the training dataset includes only labeled data. In figure 1 the Outline of Initial Solution Classification can be seen.

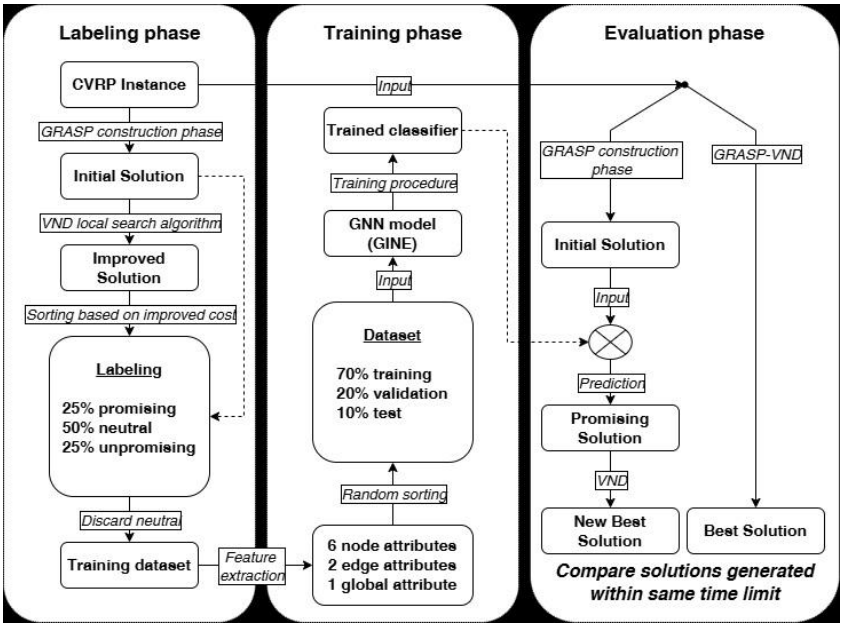

Figure 1. Outline of Initial Solution Classification

The data essentially is a set of routes and a label. It also contains information about the CVRP instance, the node coordinates and demands, and the capacity and number of vehicles. From these data node, edge and global features need to be extracted. Here are the selected features:

- Node Features:
    1. Normalized x coordinate for each node i named $\hat{x}_i \in [0,1]$

$$\hat{x}_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

2. Normalized y coordinate for each node i named $\hat{y}_i \in [0,1]$

$$\hat{y}_i = \frac{y_i - y_{min}}{y_{max} - y_{min}}$$

3. The demand/capacity of vehicle $\in [0,1]$
4. The flag *is_depot* which is 1 id the node is the depot and 0 otherwise
5. Normalized distance to depot

$$dist\_depot_i = \frac{\sqrt{(x_i - x_d)^2 + (y_i - y_d)^2}}{diameter}$$

where $diameter = max\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ is the diameter of the graph and (x_d, y_d) are the coordinates of depot

6. The positional symmetry which addresses the position of a node in a route with being 0 if the node is in the middle of the route and 1 near the depot. More precisely

$$pos\_sym = \begin{cases} 0, if\ depot \\ 2\left|\frac{k-1}{L-1} - 0.5\right|, otherwise \end{cases}$$

where L is the length of the route (excluding depot) and k the position of the node in the route. By taking the absolute value, we include the undirected aspect of the problem

- Edge Features:

1. Normalized distance between nodes i and j

$$dist = \frac{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{diameter}$$

2. The flag *route_flag* which is 1 if the edge belongs to the initial solution and 0 otherwise

- Global Feature:

1. The normalized weighted mean centroid to depot distance of the routes r of the initial solution or WMCD

$$WMCD = \sum_{r=1}^{R} w_r \frac{\sqrt{\left(C_{r,x} - x_d\right)^2 + \left(C_{r,y} - y_d\right)^2}}{diameter}$$

where R is the set of routes, $w_r = \frac{\sum_{i \in V_r} d_i}{\sum_j d_j}$, and the coordinates of centroid to depot of a route $C_{r,x} = \frac{1}{|V_r|}\sum_{i \in V_r} x_i$, $C_{r,y} = \frac{1}{|V_r|}\sum_{i \in V_r} y_i$

The selection of the features is based on the idea of providing information about each instance (node features 1,2,3,4,5 and edge features) and each initial solution (node feature 6 and global feature).

A full connected graph would produce many edges that would be computationally costly for a single message passing layer, since the neighborhood of a node would be the whole graph. For that reason, a route k-NN edge strategy is selected, which builds the edge set as a union of two types of edges:

- k-NN edges (directed): Each node is connected to its k closest neighbors using the x,y coordinates. These edges are directed, so if node A is among B's neighbors, it means B→A but it's not guaranteed that A→B.
- route edges (bidirectional): Taken from the initial solution. Since the problem is undirected, they are bidirectional. If a neighbor B is connected to node A in the initial solution, then it denotes A↔B

However, the message passed comes from the neighbor nodes and their corresponding directed edges. The route edges are applied to introduce neighbors that are outside the k closest neighbors but are connected in an initial solution. In that way more information is propagating through the graph.

3. Computational Results

The simple metaheuristic GRASP-VND and the enhanced version GINE-GRASP-VND are coded in Python along with the necessary utilities. The parameter alpha that is used for GRASP is 0.3 Some well-known and publicly available instances of CVRP are used to evaluate the proposed methodology. From the CVRPLIB repository [9] Set A (Augerat,1995) are used to verify the effectiveness of the ML approach. The following table gives information about each instance:

Instances 1,6,7,11,12,13,16,18,19,23,26 and 27 from the table are selected for labeling. For each instance the GRASP-VND is run, to generate labeled data. 25% of the solutions with the lowest improved cost are labeled as promising solutions and 25% with the highest

improved cost as unpromising. The remaining 50% are discarded. The following table presents the number of labeled data per instance:

Instance	# of labeled samples
A-n32-k5	81
A-n37-k5	100
A-n37-k6	101
A-n44-k6	102
A-n45-k6	100
A-n45-k7	100
A-n53-k7	128
A-n55-k9	127
A-n60-k9	150
A-n63-k10	152
A-n69-k9	177
A-n80-k10	204
Total	**1522**

The total of 1522 labeled samples is fed to GINE model. 70% is used as the training set, 20% as the validation set and 10% as the test set. The route k-NN edge structure is used with k = 14. The model consists of 4 layers of GINEconv with 192 hidden dimensions. A softmax function is used for classification, Cross-Entropy is the loss function and Adam is the optimizer.

The neural network trained with labeled CVRP instances of node sizes 30 to 80 managed to classify correctly only 60.12% of the training samples. The method was first tested to assume its generality upon different sizes of nodes. It was found that instances with up to 100 customers can provide the potential effectiveness of the method.

After 68 epochs of total running time 15 min 52 s, the metrics are:

- Cross-Entropy Loss = 0.6692
- Classification accuracy = 0.6012, i.e. the model classifies correctly the 60.12% of the labeled samples
- True Negatives (TN)= 632, i.e. the number of samples that are truly labeled “0” (unpromising) and are classified by the model as “0”
- False Positives (FP) = 127, i.e. the number of samples that are truly labeled “0” but are classified by the model as “1” (promising)
- False Negatives (FN) = 480, i.e. the number of samples that are truly labeled “1” but are classified by the model as “0”
- True Positives (TP) = 283, i.e. the samples that are truly labeled “1” and are classified by the model as “1”

Since the training samples have perfectly balanced classes – i.e. 50% are "1" and 50% are "0" – precision, recall and f1 score for the class "1" are calculated as follows:

- $Precision = \frac{TP}{TP+FP} = 69.0\%$, i.e. 69% of the samples the model said were promising, are really promising.
- $Recall = \frac{TP}{TP+FN} = 37.1\%$, i.e the model correctly identified 37.1% of all truly promising solutions
- $F1 = 2\frac{Precision \times Recall}{Precision+Recall} = 48.3\%$, i.e. the balance between precision and recall

It seems that the trained model is better than a random guessing (50%) of running VND or not, since it identifies 60.12% of the samples. According to its precision metric, 31% of VND runs are wasted on solutions that won't lead to better improved solutions. On the other hand, the recall metric point that it loses 62.9% of promising solutions meaning they never go through VND.

Then the model was incorporated with the GRASP-VND and was compared with the simple metaheuristic with several instances and for the same run time. When applying the enhanced method to a large size instance it provided a significant improved solution than the simple metaheuristic, suggesting that it exists room for improvement.

instance	GRASP-VND	GINE-GRASP-VND					
	Mean GAP %	Mean GAP %	Accuracy	TN	TP	FN	FP
A-n32-k5	1,218	**0,746**	0.63	29	22	19	11
A-n37-k5	4,118	2,997	0.51	49	2	48	1
A-n37-k6	3,172	3,904	0.584	45	14	37	5
A-n54-k7	5,047	4,640	0603	50	32	36	7
A-n80-k10	10,990	11,058	0.623	58	69	32	45
M-n101-k10	20,213	15,890	0.669	65	83	49	56

4. Conclusions

This paper investigated a hybrid machine learning approach for enhancing a classical GRASP–BVND metaheuristic for the CVRP. The proposed method integrates an edge-aware GINE as an ISC that predicts whether a GRASP-constructed solution is promising enough to justify the application of a Variable Neighborhood Descent local search. The goal is to reduce unnecessary local search calls while preserving, or ideally improving, solution quality.

The GINE classifier was trained on labeled initial solutions generated by GRASP–BVND on CVRP instances with 30 to 80 customers from the Augerat Set A. Using a balanced labeling scheme, the model achieved a classification accuracy of around 60% on held-out data, which is only moderately better than random guessing on balanced classes. The precision of the "promising" class shows that a non-negligible fraction of local search runs is still wasted on unpromising solutions, while the low recall indicates that many promising solutions are incorrectly filtered out and never improved by VND.

When the trained classifier was embedded into the GRASP–BVND framework and tested under equal time limits, the results were mixed. On a larger instance, the enhanced method yielded a noticeably better solution quality, suggesting that selective local search guided by a learned model may become more beneficial as instance size grows and the cost of local search increases.

Overall, the study provides an initial proof-of-concept that GNN-based classifiers can be integrated into metaheuristics for VRPs, but also highlights several limitations. The current feature set and labeling strategy appear insufficient to obtain a high-quality classifier, and the resulting accuracy is not yet strong enough to consistently improve the metaheuristic on standard benchmark instances. Future work should therefore focus on designing richer graph and route features, exploring alternative labeling schemes (e.g., using continuous improvement scores or cost-sensitive thresholds), and experimenting with more advanced architectures or training procedures, such as cost-aware losses or ensemble methods. In addition, extending the approach to larger and more diverse VRP variants, and integrating it with other learning components (e.g., learned neighborhood selection or adaptive parameter control), may further unlock the potential of machine learning–driven metaheuristics for real-world routing applications.

References

[1] Bengio, Y., Lodi, A., Prouvost, A. *Machine learning for combinatorial optimization: A methodological tour d'horizon* - European Journal of Operational Research ISSN 0377-2217. [pp. 405-421]. [February-15] 2021

[2] LeCun Y., Bengio Y., Hinton G. - Deep learning - Nature, ISSN 0028-0836. [pp. 436-444]. [May] 2015

[3] Mazzieri, D. *Machine Learning for combinatorial optimization: the case of Vehicle Routing* - University of Bolognia digital library. [December- 3] 2021

[4] Gunawan A., Kendall G., McCollum B., Seow H.-V., Lee L. S. - *Vehicle routing: Review of benchmark datasets* - Journal of The Operational Research Society, ISSN 0160-5682. [pages not specified] 2021

[5] Blum, C., Roli, A. - *Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison* - ACM Computing Surveys ISSN 0360-0300. [pp. 268–308]. [September-01] 2003

[6] Bogyrbayeva A., Meraliyev M., Mustakhov T., Dauletbayev B. - *Machine Learning to Solve Vehicle Routing Problems: A Survey* - IEEE Transactions on Intelligent Transportation Systems, ISSN 1524-9050. Vol. 25, No. 6, 2024

[7] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M. - *Graph neural networks: A review of methods and applications* - AI Open ISSN 2666-6510. [pp. 57–81]. [January-01] 2020

[8] Mesa, J. P., Montoya, A., Ramos-Pollan, R., Toro, M. - *Machine-learning component for multi-start metaheuristics to solve the capacitated vehicle routing problem* - Applied Soft Computing ISSN 1568-4946. [p. 112916]. [April-01] 2025

[9] Lima, I. - *CVRPLIB* - Online resource. [accessed at https://vrp.galgos.inf.puc-rio.br/index.php/en/]. [January-01] 2014.

Bibliography

Bengio, Y., Lodi, A., Prouvost, A. - *Machine learning for combinatorial optimization: A methodological tour d'horizon* - European Journal of Operational Research ISSN 0377-2217. [pp. 405-421]. [February-15] 2021

Blum, C., Roli, A. - *Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison* - ACM Computing Surveys ISSN 0360-0300. [pp. 268–308]. [September-01] 2003

Bogyrbayeva A., Meraliyev M., Mustakhov T., Dauletbayev B. - *Machine Learning to Solve Vehicle Routing Problems: A Survey* - IEEE Transactions on Intelligent Transportation Systems, ISSN 1524-9050. Vol. 25, No. 6, 2024

Gunawan A., Kendall G., McCollum B., Seow H.-V., Lee L. S. - *Vehicle routing: Review of benchmark datasets* - Journal of The Operational Research Society, ISSN 0160-5682. [pages not specified] 2021

LeCun Y., Bengio Y., Hinton G. - *Deep learning* - Nature, ISSN 0028-0836. [pp. 436-444]. [May] 2015

Lima, I. - *CVRPLIB* - Online resource. [accessed at https://vrp.galgos.inf.puc-rio.br/index.php/en/]. [January-01] 2014.

Mazzieri, D. - *Machine Learning for combinatorial optimization: the case of Vehicle Routing* - University of Bologna digital library. [December- 3] 2021

Mesa, J. P., Montoya, A., Ramos-Pollan, R., Toro, M. - *Machine-learning component for multi-start metaheuristics to solve the capacitated vehicle routing problem* - Applied Soft Computing ISSN 1568-4946. [p. 112916]. [April-01] 2025

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M. - *Graph neural networks: A review of methods and applications* - AI Open ISSN 2666-6510. [pp. 57–81]. [January-01] 2020