Neural Embedded Mixed-Integer Optimization for Location-Routing Problems

Waquar Kaleem[1], **Doyoung Lee**[2,3], **Changhyun Kwon**[2,3], **Anirudh Subramanyam**[1]
[1]**Department of Industrial and Manufacturing Engineering, Pennsylvania State University, USA**
[2]**Department of Industrial and Systems Engineering, KAIST, Daejeon 34141, Republic of Korea**
[3]**Omelet, Inc., Daejeon 34051, Republic of Korea**

Abstract

We propose Neural-Embedded Optimization for the Location-Routing Problem (NEO-LRP), which jointly minimizes facility opening and vehicle routing costs. Deep set and graph neural networks are used to predict vehicle routing costs for arbitrary customer subsets and then used as surrogates within a mixed-integer program. This reformulation significantly reduces model complexity and enables an efficient solution. The modular design supports generalization to various vehicle routing variants and constraints. Computational results on benchmark instances show that the proposed method consistently achieves near-optimal location-allocation solutions with significantly lower runtimes compared to state-of-the-art heuristics, making it a practical approach for large-scale location-routing problems.

1 Introduction

The Capacitated Location-Routing Problem (CLRP) integrates two fundamental logistics problems, namely facility location and vehicle routing, into a unified framework. It aims to determine which depots to open and how to route vehicles from these depots to serve customer demands while minimizing both vehicle routing and facility opening costs under capacity constraints. The CLRP naturally arises in applications such as micro-delivery or micro-fulfillment hubs and urban consolidation centers (UCCs) for last-mile distribution, where facility opening and vehicle routes must be jointly planned (New York City Department of Transportation, 2025). Because it is NP-hard and generalizes both the Facility Location Problem (FLP) and the Capacitated Vehicle Routing Problem (CVRP), solving the CLRP has remained a computational challenge in both theory and practice.

Combining FLP and CVRP, the CLRP considers capacity constraints on depots (i.e., facilities) and fixed costs for depot openings and vehicle usage. Figure 1 illustrates a feasible CLRP solution, depicting depots (squares) and customers (circles). Depot location decisions directly influence the routing decisions, highlighting the integrated nature of the problem. Specifically, the CLRP involves both strategic decisions, such as determining optimal depot locations and the number of vehicles to operate, as well as tactical decisions, such as vehicle routing plans for each opened depot. The primary objective is to minimize the sum of depot opening costs, fixed vehicle costs, and routing costs across the network.

Prior work on the CLRP falls into three main categories. *Exact methods* such as branch(-price)-and-cut offer optimality guarantees, but can scale poorly due to the combinatorial challenge of jointly modeling location and routing decisions (Baldacci et al., 2011; Contardo et al., 2014, 2013). *Heuristic methods*, including tabu search (Prins et al., 2007), genetic algorithms (Duhamel et al., 2010), and randomized variable neighborhood search (Löffler et al., 2023), scale to large instances but require heavy parameter tuning and problem-specific design (Schneider and Löffler, 2019). More recently, *machine learning approaches* use surrogate models to approximate routing costs or other decisions,

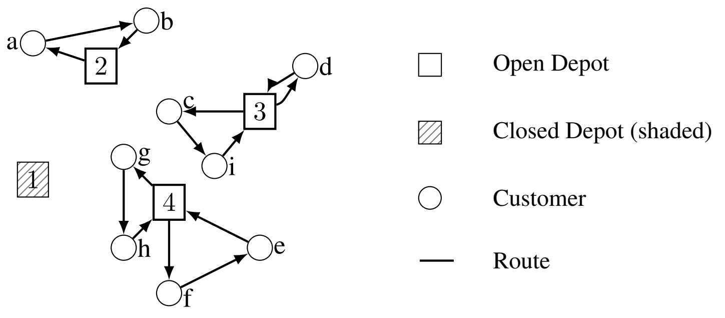

Figure 1: An Example of a CLRP Solution

which are then incorporated into metaheuristics such as genetic algorithms (Varol et al., 2024; Sobhanan et al., 2025). While these methods demonstrate promising scalability, they suffer from a lack of modularity, limited support for side constraints, and heavy reliance on parameter tuning.

The main contributions of our work are threefold. *First*, we propose Neural Embedded Optimization for Location-Routing Problems (NEO-LRP), a novel solution framework that approximates the vehicle routing cost at each candidate depot using either a permutation-invariant or graph neural network, which is then embedded within an easy-to-solve MIP model to determine location-allocation decisions. *Second*, we demonstrate that the same neural-embedded MIP formulation can flexibly generalize across multiple problem variants, such as the multi-depot VRP (MDVRP) and Two-Echelon CVRP, and accommodate structural differences in constraint specifications—e.g., the presence or absence of depot or vehicle capacity limits—via minimal model modifications. *Third*, we show through extensive experiments that, when compared to state-of-the-art heuristics, NEO-LRP achieves comparable or superior solution quality while significantly reducing computation time, offering a promising alternative for scalable and joint location and routing optimization.

2 Neural Embedded Optimization for Location-Routing Problems

We have provided an exact mathematical optimization formulation of the CLRP in Appendix A. The exact CLRP formulation faces severe scalability issues due to the arc-flow variables, whose number grows quadratically with the number of customers. As a result, even moderate-sized instances with fewer than one hundred customers and multiple depots can become intractable for standard mixed-integer programming solvers. This stems from the tightly coupled, combinatorial nature of the depot location and routing decisions.

To address this, we propose a neural-embedded reformulation that decomposes the CLRP into two hierarchical components: a high-level facility location problem (FLP) for depot openings and customer assignments, and a low-level capacitated vehicle routing problem (CVRP) for constructing feasible routes. This separation preserves the core CLRP structure while isolating routing cost evaluation.

We introduce the notation necessary for the introduction of the neural-embedded reformulation of the CLRP. Consider a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where the node set $\mathcal{V} = \mathcal{V}_D \cup \mathcal{V}_C$ consists of potential depot nodes $\mathcal{V}_D$ and customer nodes $\mathcal{V}_C$. The arc set $\mathcal{A} \subseteq \{(i,j) \in \mathcal{V} \times \mathcal{V} \mid i \neq j,\ (i,j) \notin \mathcal{V}_D \times \mathcal{V}_D\}$ represents feasible travel paths between nodes. Let F denote the fixed cost per vehicle, and let opening a depot $d \in \mathcal{V}_D$ incur a fixed cost O_d. Each arc $(i,j) \in \mathcal{A}$ is associated with a travel cost c_{ij}. For each node $i \in \mathcal{V}$, the set of leaving arcs is denoted by $\delta^+(i) = \{(i,j) \mid (i,j) \in \mathcal{A}\}$. The binary variables $x_{ijd} \in \{0,1\}$ represent the routing decisions, indicating whether a vehicle originating from depot $d \in \mathcal{V}_D$ traverses arc $(i,j) \in \mathcal{A}$. The binary variables $y_d \in \{0,1\}$ indicate depot opening decisions.

The original CLRP objective includes depot opening, routing, and vehicle usage costs. The latter two components involve the arc-level routing variables x_{ijd}:

$$\min \sum_{d \in \mathcal{V}_D} O_d y_d + \sum_{d \in \mathcal{V}_D} \sum_{(i,j) \in \mathcal{A}} c_{ij} x_{ijd} + F \sum_{d \in \mathcal{V}_D} \sum_{(d,j) \in \delta^+(d)} x_{djd}.$$

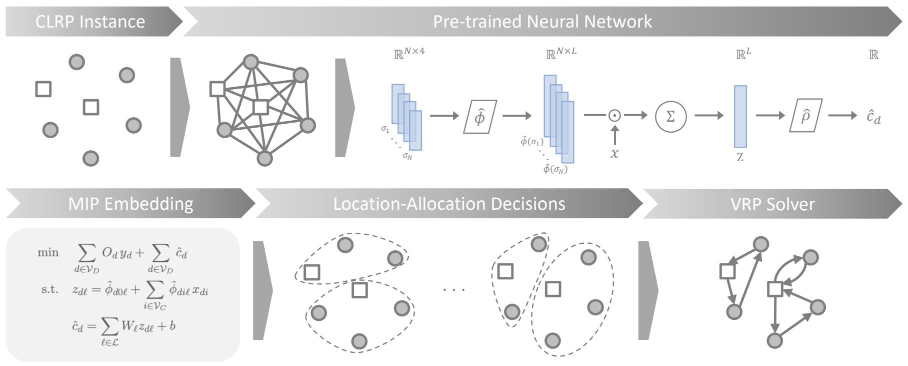
CLRP Instance
Pre-trained Neural Network
$\mathbb{R}^{N\times 4}$
$\mathbb{R}^{N\times L}$
$\mathbb{R}^{L}$
$\mathbb{R}$
σ_1
σ_N
$\hat{\phi}$
$\hat{\phi}(\sigma_1)$
$\hat{\phi}(\sigma_N)$
x
Σ
Z
$\hat{\rho}$
$\hat{c}_d$
MIP Embedding
Location-Allocation Decisions
VRP Solver
$\min \sum_{d\in\mathcal{V}_D} O_d y_d + \sum_{d\in\mathcal{V}_D} \hat{c}_d$
s.t. $z_{d\ell} = \hat{\phi}_{d0\ell} + \sum_{i\in\mathcal{V}_C} \hat{\phi}_{di\ell} x_{di}$
$\hat{c}_d = \sum_{\ell\in\mathcal{L}} W_\ell z_{d\ell} + b$

Figure 2: The NEO-LRP framework for solving the CLRP

Since the number of arc-flow variables grows quadratically with the number of customers, we approximate the latter two cost components (routing and vehicle costs) using a depot-level cost function, defined as

$$c_d(x) \approx \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ijd} + F \sum_{(d,j)\in\delta^+(d)} x_{djd},$$

which yields a simplified objective

$$\min \sum_{d\in\mathcal{V}_D} O_d y_d + \sum_{d\in\mathcal{V}_D} c_d(x),$$

where $x = (x_{ijd} : (i,j) \in \mathcal{A}, d \in \mathcal{V}_D)$. Note that $c_d(x)$ can be further simplified, with a slight abuse of the notation, as $c_d(S_d)$ where $S_d \subseteq \mathcal{V}_C$ is the set of customers assigned to depot d, since the routing variables x are unnecessary for the location decisions and they can be determined implicitly via solving a CVRP for each depot d.

We will approximate $c_d(\cdot)$ by a neural surrogate $\hat{c}_d(\cdot)$, which enables the cost prediction without solving CVRPs. Our surrogate-based reformulation will replace the arc-flow variables with a few scalar cost terms, reducing the model size. However, the challenge now is that the route evaluation burden is offloaded to a decision-dependent neural predictor. We show how to alleviate this decision dependence without sacrificing computational scalability in the subsequent sections.

2.1 Overall Framework

Figure 2 illustrates our Neural-Embedded Optimization framework for the CLRP, which we call NEO-LRP. The framework integrates neural cost predictions into an MIP model, enabling efficient and scalable end-to-end optimization.

The neural cost predictor $f_\theta(\mathcal{G})$, with $\mathcal{G}$ representing the depot-customer input graph and θ being the model parameters, is a permutation-invariant set function, allowing node embeddings to be aggregated based on location–allocation decisions. This property enables its embedding as MIP constraints. It is important to highlight that the neural encoder that generates node embeddings is trained offline, allowing for deep embedding networks without affecting the final MIP performance. While precomputing embeddings and aggregating them during optimization may omit structural information—such as the exact customer subset each depot will serve—it significantly improves computational efficiency by avoiding repeated neural evaluations during optimization.

The modular design enables adaptation to different problem variants without major changes to the overall framework. Moreover, by decomposing the CLRP into a facility location phase and a routing phase, our framework first solves a neural-embedded FLP to generate high-quality location-allocation decisions, and then completes routing via a CVRP solver. When the embeddings preserve relative cost quality, this two-stage approach yields fast and accurate solutions.

2.2 Neural Surrogate Modeling and Cost Predictor

We begin by presenting an exact representation result, which establishes that each routing cost function $c_d(S)$, for any subset $S \subseteq \mathcal{V}_C$, can be represented using a common deep sets architecture

(Zaheer et al., 2017) that is independent of the depot node $d \in \mathcal{V}_D$. The proof of the following theorem is omitted for the sake of brevity.

Theorem 1 (Depot-independent sum-decomposition of routing costs)**.** *Let $P > 0$ be any fixed constant. Define the normalized feature vector $\sigma_{di} := (P^{-1}(p_i^{\mathrm{x}} - p_d^{\mathrm{x}}), P^{-1}(p_i^{\mathrm{y}} - p_d^{\mathrm{y}}), C^{-1}d_i) \in \mathbb{R}^3$ for all depots $d \in \mathcal{V}_D$ and customers $i \in \mathcal{V}_C$, where $(p_j^{\mathrm{x}}, p_j^{\mathrm{y}})$ denotes the coordinates of node j. Then, there exist $L \in \mathbb{N}$ and functions $\phi : \mathbb{R}^3 \to \mathbb{R}^L$ and $\rho : \mathbb{R}^L \to \mathbb{R}$ such that*

$$c_d(S) = \rho\left(\sum_{i \in S} \phi(\sigma_{di})\right) \tag{1}$$

for all $S \subseteq \mathcal{V}_C$ and $d \in \mathcal{V}_D$.

Theorem 1 suggests that common ϕ and ρ can be used to decompose functions c_d for all $d \in \mathcal{V}_D$, and can approximate the optimal cost of *any* CVRP instance with appropriately normalized features, specifically, depot-centered coordinates and capacity-normalized demands. However, although the theorem ensures the existence of such functions, it does not provide an explicit construction method. In practice, these functions can be approximated using various machine learning approaches including feedforward neural networks (Zaheer et al., 2017), kernel methods (Schölkopf and Smola, 2002), or more expressive architectures like graph neural networks (Khalil et al., 2017; Joshi et al., 2019) and graph transformers (Kool et al., 2019; Kwon et al., 2020). While the latter employs attention mechanisms beyond the node-independent processing suggested by the theorem, they coincide with the former whenever the underlying graphs are complete and preserve permutation invariance even when they may not be complete.

We approximate ϕ and ρ with neural networks $\hat{\phi}$ and $\hat{\rho}$, respectively. We implement $\hat{\phi}$ using two architectures: fully-connected feedforward neural networks (FFNN) and Graph Transformers (GT). In contrast, $\hat{\rho}$ is implemented as an FFNN for regression and embedding within the final MIP model. In case of GT, we augment the three-dimensional features from Theorem 1 with a "node type" indicator, yielding $\hat{\sigma}_{di} = (P^{-1}(p_i^{\mathrm{x}} - p_d^{\mathrm{x}}), P^{-1}(p_i^{\mathrm{y}} - p_d^{\mathrm{y}}), C^{-1}d_i, 0) \in \mathbb{R}^4$ for each customer node i and $(0, 0, 0, 1) \in \mathbb{R}^4$ for depots. This representation enables $\hat{\phi}$ to extract features into an L-dimensional latent space, which $\hat{\rho}$ then maps to cost predictions. Implementation details are provided in Appendix B.

2.3 Neural Embedded MIP Formulation for the CLRP

The neural-embedded formulation for the location-allocation component of the CLRP integrates latent node embeddings, generated using either FFNN or GT, into the optimization model. Most decision variables are identical to those defined in Section 2, with the following additions: binary variables $a_{di} \in \{0, 1\}$ indicate customer assignments to depots, continuous variables $z_{d\ell} \in \mathbb{R}$ represent latent embedding aggregations for each depot $d \in \mathcal{V}_D$ and latent dimension $\ell \in \mathcal{L}$, and continuous variables $\tilde{c}_d \in \mathbb{R}_+$ represent the predicted route and fixed vehicle costs associated with depot $d \in \mathcal{V}_D$. The weights W_ℓ and biases b are obtained from the trained neural network.

The neural embedded MIP formulation is as follows:

$$\min \quad \sum_{d \in \mathcal{V}_D} O_d\, y_d + \sum_{d \in \mathcal{V}_D} \tilde{c}_d \tag{2}$$

$$\text{s.t.} \quad \sum_{d \in \mathcal{V}_D} a_{di} = 1 \qquad \forall i \in \mathcal{V}_C \tag{3}$$

$$a_{di} \le y_d \qquad \forall d \in \mathcal{V}_D,\ i \in \mathcal{V}_C \tag{4}$$

$$\sum_{i \in \mathcal{V}_C} d_i\, a_{di} \le Q_d\, y_d \qquad \forall d \in \mathcal{V}_D \tag{5}$$

$$z_{d\ell} = \phi_{d0\ell} + \sum_{i \in \mathcal{V}_C} \phi_{di\ell}\, a_{di} \qquad \forall d \in \mathcal{V}_D,\ \ell \in \mathcal{L} \tag{6}$$

$$\tilde{c}_d \le M\, y_d \qquad \forall d \in \mathcal{V}_D \tag{7}$$

$$\tilde{c}_d \ge \sum_{\ell \in \mathcal{L}} W_\ell z_{d\ell} + b - M(1 - y_d) \qquad \forall d \in \mathcal{V}_D \tag{8}$$

$$\tilde{c}_d \leq \sum_{\ell \in \mathcal{L}} W_\ell z_{d\ell} + b + M(1 - y_d) \qquad \forall d \in \mathcal{V}_D \tag{9}$$

$$a_{di} \in \{0,1\} \qquad \forall d \in \mathcal{V}_D,\ i \in \mathcal{V}_C \tag{10}$$

$$y_d \in \{0,1\} \qquad \forall d \in \mathcal{V}_D \tag{11}$$

$$z_{d\ell} \in \mathbb{R} \qquad \forall d \in \mathcal{V}_D,\ \ell \in \mathcal{L} \tag{12}$$

$$\tilde{c}_d \geq 0 \qquad \forall d \in \mathcal{V}_D \tag{13}$$

The objective function (2) minimizes the sum of depot opening costs and the predicted costs ($\tilde{c}_d$). Constraints (3)–(4) ensure that each customer is assigned to exactly one open depot. Constraints (5) impose capacity restrictions for depots. Constraints (6) aggregate the latent node embeddings $\phi_\bullet$ into depot-specific embedding vectors by summing the depot embedding and customer embeddings according to the depot-customer assignments defined by a_{di}. Constraints (7)–(9) enforce logical conditions via big-M constraints, consistent with prior approaches for embedding neural predictions into MIPs (Fischetti and Jo, 2018). Constraints (10)–(13) specify the domain of the decision variables.

2.3.1 Extension to Variant and Constrained LRP Settings

Extension to CLRP Variants The modular structure of our neural-embedded framework extends naturally to hierarchical CVRP variants such as the Multi-Depot VRP (MDVRP) and Two-Echelon CVRP (2E-CVRP). While these problems inherently involve routing at the lower level, our approach can approximate the corresponding costs via neural surrogates. For instance, in the 2E-CVRP, the second-echelon delivery cost is captured by a depot-specific surrogate variable $\tilde{c}_s$, integrated into a neural-augmented FLP-MIP:

$$\min \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} c_{ij}\, x_{ijk} + \sum_{s \in \mathcal{V}_S} \tilde{c}_s.$$

with latent embedding constraints:

$$z_{s\ell} = \phi_{s0\ell} + \sum_{i \in \mathcal{V}_C} \phi_{si\ell}\, a_{si}, \quad \tilde{c}_s = \sum_{\ell \in \mathcal{L}} W_\ell z_{s\ell} + b.$$

This maintains MIP compatibility while enabling adaptation to diverse two-tier delivery settings. Full details are in Appendix D.

Unified Handling of Heterogeneous Constraints Existing CLRP benchmarks vary widely in constraint specifications; for example, some include both vehicle and depot capacities (Prins et al., 2004; Barreto, 2004), while others omit depot limits (Tuzun and Burke, 1999). Heuristics often require structural redesign to adapt to such variations. In contrast, our proposed neural-embedded MIP formulation accommodates these differences via simple constraint toggles (e.g., enabling or removing depot capacity constraints like $\sum_{i \in \mathcal{V}_C} d_i\, a_{di} \leq Q_d\, y_d$). This flexibility allows for portability of our pretrained networks and framework across diverse benchmark sets.

2.3.2 Reduction in Variable Complexity

We compare the number of decision variables in the exact CLRP formulation and the neural-embedded MIP to assess model complexity. The exact CLRP includes arc-level routing and flow variables, which scale quadratically with the number of customers. In contrast, the neural-embedded formulation eliminates these arc-dependent variables and introduces a small set of latent variables $z_{d\ell}$, whose size depends only on the number of depots and the embedding dimension L. By adjusting L, we can directly control the number of continuous variables, offering a tunable trade-off between model expressiveness and computational cost.

As shown in Appendix E (Tables 3–4), binary variables are reduced by factors of $28\times$ to $200\times$, and continuous variables by over $600\times$ in large instances. This reduction makes the CLRP tractable for problem sizes that are otherwise infeasible under the exact formulation.

2.4 Final Solution Construction

The neural embedded MIP formulation (Section 2.3) focuses on the facility location and customer allocation aspects of the CLRP, using a neural network to approximate routing costs. As it does

Table 1: Comparison with SOTA Heuristics by Instance Size (averaged)

Size	HCC-500K		$\text{TSBA}_{\text{speed}}$		GRASP/VNS		NEO-FFNN (ours)		NEO-GT (ours)	
	$E_{\text{BKS}}^{\text{gap}}$ (%)	T_{total} (s)	$E_{\text{BKS}}^{\text{gap}}$ (%)	T_{total} (s)	$E_{\text{BKS}}^{\text{gap}}$ (%)	T_{total} (s)	$E_{\text{BKS}}^{\text{gap}}$ (%)	T_{total} (s)	$E_{\text{BKS}}^{\text{gap}}$ (%)	T_{total} (s)
20	0.00	49.50	0.00	0.65	0.02	0.71	6.05	0.06	5.70	0.06
50	0.09	113.50	0.06	2.88	0.14	8.31	3.44	0.10	3.20	0.09
100	1.20	449.25	0.27	15.08	1.40	72.34	2.14	0.38	1.97	0.36
200	1.47	1172.67	0.36	125.96	1.24	715.53	1.36	1.36	0.97	1.25

not generate explicit vehicle routes, we solve the CVRP in a post-processing step. Specifically, given the facility openings y_d and customer assignments a_{di} from the optimization model, we use an existing solver, namely VROOM (Coupey et al., 2024), to compute detailed CVRP solutions and exact routing costs along with the selected location-allocation decisions.

3 Experimental Results

We evaluate the proposed methodology against baseline heuristics. Experiments were run on a Linux machine equipped with an AMD Ryzen 9 5900X, 64 GB RAM, RTX 4070 GPU, running CUDA 12.6, Ubuntu 22.04. We used Gurobi 10.0.2 with the `gurobi-machinelearning` 1.3.2 package (Gurobi Optimization, LLC, 2021; Gurobi Optimization, 2024) and implemented all neural network components in PyTorch 2.0.1 (Paszke et al., 2019). Ground-truth routes were generated using VROOM (Coupey et al., 2024).

In Table 1, we provide a detailed performance comparison of the proposed NEO-LRP framework against three prominent heuristic algorithms for the CLRP: HCC-500K (Hemmelmayr et al., 2012), $\text{TSBA}_{\text{speed}}$ (Schneider and Löffler, 2019), and GRASP/VNS (Löffler et al., 2023). Experiments were conducted on the standard Prodhon benchmark set (Prins et al., 2004). See Appendix B.3 for details. For the baseline heuristics (HCC-500K, $\text{TSBA}_{\text{speed}}$, and GRASP/VNS), we report the performance values as published in the original studies (Hemmelmayr et al., 2012; Schneider and Löffler, 2019; Löffler et al., 2023).

We report results for two variants of our neural-embedded optimization framework, NEO-FFNN and NEO-GT, where the cost prediction model is implemented using a fully-connected feedforward neural network and a graph transformer, respectively. Results are averaged over five runs per instance and include both solution quality, expressed as the percentage gap to the best-known solution ($E_{\text{BKS}}^{\text{gap}}$), and total computation time (T_{total}) in seconds. Both NEO variants demonstrate strong performance, with NEO-FFNN achieving reasonable solution quality within sub-second runtimes. The NEO-GT model, however, consistently outperforms HCC-500K and GRASP/VNS in both accuracy and especially speed for large-scale instances. Although it does not always match the optimality gap of $\text{TSBA}_{\text{speed}}$, NEO-GT delivers near-optimal solutions with significantly reduced computation time. This efficiency stems from the hybrid design that embeds graph transformer-based cost predictions into a location-allocation MIP formulation, followed by fast post-hoc routing via VROOM. These results demonstrate the scalability and practicality of NEO-LRP for large-scale CLRP settings. Full per-instance results and ablation studies are presented in Appendices C and F, respectively.

4 Conclusion

This work introduced NEO-LRP, a neural-embedded optimization framework that integrates machine learning with mixed-integer programming to address the Capacitated Location-Routing Problem. By replacing arc-level routing variables with depot-level surrogate costs predicted by neural networks, the framework yields a compact MIP formulation that substantially reduces model size and makes large-scale instances tractable. Our experiments on benchmark instances demonstrate that NEO-LRP consistently achieves high-quality solutions with runtimes far shorter than state-of-the-art heuristics. These results highlight the potential of neural-embedded optimization as a scalable and practical solution method for location-routing problems. Looking ahead, the methodology can be extended to other CLRP variants, such as the MDVRP and 2E-CVRP, and adapted to incorporate additional side constraints through modular constraint handling.

References

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS) Deep Learning Symposium*. arXiv preprint arXiv:1607.06450.

Baldacci, R., Mingozzi, A., and Wolfler Calvo, R. (2011). An exact method for the capacitated location-routing problem. *Operations Research*, 59(5):1284–1296.

Barreto, S. (2004). *Análise e Modelização de Problemas de localização-distribuição*. PhD thesis, Universidade de Aveiro (Portugal).

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (elus). In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*.

Contardo, C., Cordeau, J.-F., and Gendron, B. (2013). A computational comparison of flow formulations for the capacitated location-routing problem. *Discrete Optimization*, 10(4):263–295.

Contardo, C., Cordeau, J.-F., and Gendron, B. (2014). An exact algorithm based on cut-and-column generation for the capacitated location-routing problem. *INFORMS Journal on Computing*, 26(1):88–102.

Coupey, J., Nicod, J.-M., and Varnier, C. (2024). *VROOM v1.14, Vehicle Routing Open-source Optimization Machine*. Verso (`https://verso-optim.com/`), Besançon, France. `http://vroom-project.org/`.

Duhamel, C., Lacomme, P., Prins, C., and Prodhon, C. (2010). A grasp$\times$ els approach for the capacitated location-routing problem. *Computers & Operations Research*, 37(11):1912–1923.

Errami, N., Queiroga, E., Sadykov, R., and Uchoa, E. (2024). Vrpsolvereasy: a python library for the exact solution of a rich vehicle routing problem. *INFORMS Journal on Computing*, 36(4):956–965.

Fischetti, M. and Jo, J. (2018). Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3):296–309.

Furnon, V. and Perron, L. (2024). OR-Tools routing library. `https://developers.google.com/optimization/routing/`.

Gurobi Optimization, L. (2024). gurobi-machinelearning: A python package for mixed-integer programming formulations of trained machine learning models. Python package for embedding trained ML models in MIP optimization.

Gurobi Optimization, LLC (2021). Gurobi optimizer reference manual. Online.

Hemmelmayr, V. C., Cordeau, J.-F., and Crainic, T. G. (2012). An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research*, 39(12):3215–3228.

Joshi, C. K., Laurent, T., and Bresson, X. (2019). An efficient graph convolutional network technique for the travelling salesman problem. In *INFORMS Annual Meeting*. arXiv preprint arXiv:1906.01227.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., and Song, L. (2017). Learning combinatorial optimization algorithms over graphs. *Advances in Neural Information Processing Systems*, 30.

Kool, W., van Hoof, H., and Welling, M. (2019). Attention, learn to solve routing problems! In *International Conference on Learning Representations*.

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

Löffler, M., Bartolini, E., and Schneider, M. (2023). A conceptually simple algorithm for the capacitated location-routing problem. *EURO Journal on Computational Optimization*, 11:100063.

New York City Department of Transportation (2025). Nyc dot authorizes on-street 'microhub zones' to combat negative environmental and safety effects of truck deliveries. Press release.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32.

Prins, C., Prodhon, C., and Calvo, R. W. (2004). Nouveaux algorithmes pour le problème de localisation et routage avec contraintes de capacité. In *MOSIM'04 (4ème Conf. Francophone de Modélisation et Simulation)*.

Prins, C., Prodhon, C., Ruiz, A., Soriano, P., and Wolfler Calvo, R. (2007). Solving the capacitated location-routing problem by a cooperative lagrangean relaxation-granular tabu search heuristic. *Transportation Science*, 41(4):470–483.

Schneider, M. and Löffler, M. (2019). Large composite neighborhoods for the capacitated location-routing problem. *Transportation Science*, 53(1):301–318.

Schölkopf, B. and Smola, A. J. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., and Sun, Y. (2021). Masked label prediction: Unified message passing model for semi-supervised classification. In *Proceedings of IJCAI*, pages 1548–1554.

Sobhanan, A., Park, J., Park, J., and Kwon, C. (2025). Genetic algorithms with neural cost predictor for solving hierarchical vehicle routing problems. *Transportation Science*, 59(2):322–339.

Tuzun, D. and Burke, L. I. (1999). A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research*, 116(1):87–99.

Varol, T., Özener, O. Ö., and Albey, E. (2024). Neural network estimators for optimal tour lengths of traveling salesperson problem instances with arbitrary node distributions. *Transportation Science*, 58(1):45–66.

Zaheer, M., Kottur, S., Ravanbhakhsh, S., Póczos, B., Salakhutdinov, R., and Smola, A. J. (2017). Deep sets. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3394–3404.

Appendices

A Mathematical Formulation for CLRP

Formally, the CLRP can be defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where the node set $\mathcal{V} = \mathcal{V}_D \cup \mathcal{V}_C$ consists of potential depot nodes $\mathcal{V}_D$ and customer nodes $\mathcal{V}_C$. The arcs $\mathcal{A} \subseteq \{(i,j) \in \mathcal{V} \times \mathcal{V} \mid i \neq j,\ (i,j) \notin \mathcal{V}_D \times \mathcal{V}_D\}$ represents possible direct travel paths between nodes. Let C denote the (homogeneous) vehicle capacity, Q_d the capacity of depot $d \in \mathcal{V}_D$, and F the per-vehicle fixed cost. Each customer $c \in \mathcal{V}_C$ has an associated demand d_c, and opening depot $d \in \mathcal{V}_D$ incurs a fixed opening cost O_d. Additionally, c_{ij} represents the travel cost along arc $(i,j) \in \mathcal{A}$. For each node $i \in \mathcal{V}$, the sets of arcs leaving and entering node i are denoted by $\delta^+(i) = \{(i,j) \mid (i,j) \in \mathcal{A}\}$ and $\delta^-(i) = \{(j,i) \mid (j,i) \in \mathcal{A}\}$, respectively.

The binary variables $x_{ijd} \in \{0,1\}$ represent the routing decisions, indicating whether a vehicle originating from depot $d \in \mathcal{V}_D$ traverses arc $(i,j) \in \mathcal{A}$. The continuous variables $v_{ijd} \geq 0$ represent the vehicle load on arc $(i,j) \in \mathcal{A}$ for vehicles originating from depot $d \in \mathcal{V}_D$. The binary variables $y_d \in \{0,1\}$ indicate depot opening decisions.

The three-index mathematical formulation of the CLRP is presented as follows:

$$\min \quad \sum_{d \in \mathcal{V}_D} O_d\, y_d + \sum_{d \in \mathcal{V}_D} \sum_{(i,j) \in \mathcal{A}} c_{ij}\, x_{ijd} + F \sum_{d \in \mathcal{V}_D} \sum_{(d,j) \in \delta^+(d)} x_{djd} \tag{14}$$

$$\text{s.t.} \quad x_{ijd} \leq y_d \qquad \forall d \in \mathcal{V}_D,\ (i,j) \in \mathcal{A} \tag{15}$$

$$\sum_{d \in \mathcal{V}_D} \sum_{(c,j) \in \delta^+(c)} x_{cjd} = 1 \qquad \forall c \in \mathcal{V}_C \tag{16}$$

$$\sum_{(i,c) \in \delta^-(c)} x_{icd} = \sum_{(c,j) \in \delta^+(c)} x_{cjd} \qquad \forall c \in \mathcal{V}_C,\ d \in \mathcal{V}_D \tag{17}$$

$$\sum_{d' \in \mathcal{V}_D \setminus \{d\}} \Big(\sum_{(d,j) \in \delta^+(d)} x_{djd'} + \sum_{(i,d) \in \delta^-(d)} x_{idd'} \Big) = 0 \qquad \forall d \in \mathcal{V}_D \tag{18}$$

$$\sum_{d \in \mathcal{V}_D} \sum_{(c,j) \in \delta^+(c)} v_{cjd} = \sum_{d \in \mathcal{V}_D} \sum_{(i,c) \in \delta^-(c)} v_{icd} + d_c \qquad \forall c \in \mathcal{V}_C \tag{19}$$

$$\sum_{(i,d) \in \delta^-(d)} v_{idd} \leq Q_d\, y_d \qquad \forall d \in \mathcal{V}_D \tag{20}$$

$$v_{ijd} \leq C\, x_{ijd} \qquad \forall d \in \mathcal{V}_D,\ (i,j) \in \mathcal{A} \tag{21}$$

$$v_{ijd} \geq 0 \qquad \forall d \in \mathcal{V}_D,\ (i,j) \in \mathcal{A} \tag{22}$$

$$x_{ijd} \in \{0,1\} \qquad \forall d \in \mathcal{V}_D,\ (i,j) \in \mathcal{A} \tag{23}$$

$$y_d \in \{0,1\} \qquad \forall d \in \mathcal{V}_D \tag{24}$$

The objective function (14) minimizes the sum of depot opening costs, travel costs, and fixed vehicle costs. Constraints (15) ensure that no routes are assigned to depots unless they are open. Constraint (16) ensures that each customer is assigned to exactly one depot route. Constraints (17) enforce flow conservation at each customer node separately for each depot, ensuring continuity of vehicle routes originating from the same depot. Constraints (18) eliminate direct vehicle movements between distinct depots. Constraints (19) enforce vehicle load conservation at customer nodes. Constraints (20) and (21) impose capacity restrictions for depots and vehicles, respectively. Finally, constraints (22)–(24) specify the domain of the decision variables.

B Neural Network Architecture and Implementation Details

B.1 Feature Extraction $\hat{\phi}$

We explore two neural network architectures to approximate the feature extraction function $\hat{\phi}$: fully-connected feedforward networks and Graph Transformers.

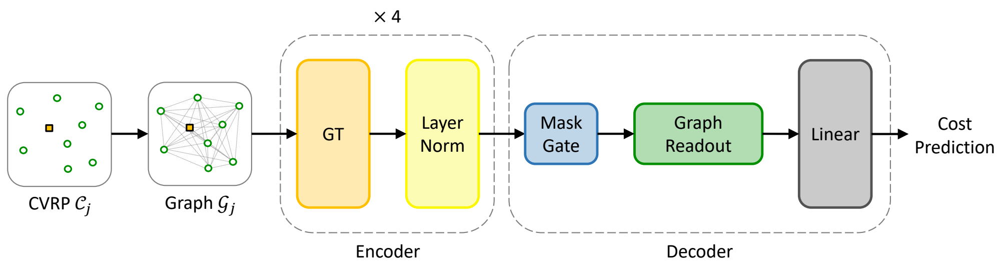

Figure 3: Neural Network Architecture

Fully-Connected Feedforward Network The first approach for $\hat{\phi}$ is based on a fully-connected feedforward architecture. In the training phase, we embed each individual customer-level feature vector, σ_{di}, into an L-dimensional latent space. This is done by passing each feature vector independently through the feature extractor network $\hat{\phi}$, and then aggregating the resulting $|S|$ latent space features to produce a single aggregated latent vector $\mathbf{z}_d$. This final embedding is then passed through a ReLU feedforward network $\hat{\rho}$ to predict the normalized output.

In our implementation, $\hat{\phi}$ consists of ReLU-activated layers except for the linear output layer, while $\hat{\rho}$ is entirely composed of ReLU-activated layers, including the final output layer. Specifically, the fully-connected architecture can be expressed as:

$$\hat{\phi}_{\text{FC}}(\sigma_{di}) = \mathbf{W}_L \cdot \text{ReLU}(\mathbf{W}_{L-1} \cdot \text{ReLU}(\cdots \text{ReLU}(\mathbf{W}_1 \sigma_{di} + \mathbf{b}_1) \cdots) + \mathbf{b}_{L-1}) + \mathbf{b}_L.$$

The detailed hyperparameter search space for the FFNN architecture is provided in Table 6 in Appendix G.

Graph Transformer Network The second approach for $\hat{\phi}$ leverages a Graph Transformer architecture to capture richer spatial relationships and inter-node dependencies. Figure 3 illustrates the neural network architecture employed for cost prediction, utilizing a Graph Transformer (GT) to effectively capture spatial relationships and inter-node dependencies inherent in CVRP instances. Given a CVRP instance $\mathcal{C}$, the neural network constructs its corresponding complete graph representation $\mathcal{G}$ and predicts the cost through the following encoder-decoder framework:

$$\mathcal{G}_d \xleftarrow[\text{construction}]{\text{graph}} (d, S) \qquad \hat{c}_d(S) = f_\theta(\mathcal{G}_d) := \texttt{Decoder}(\texttt{Encoder}(\mathcal{G}_d))$$

A given depot-customer pair (d, S), where $d \in \mathcal{V}_D$ and $S \subseteq \mathcal{V}_C$, is transformed into a complete graph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$. The node set $\mathcal{V}$ comprises the depot node d and $|S|$ assigned customer nodes, and $\mathcal{E}$ is the set of edges connecting all node pairs. Each node $v \in \mathcal{V}$ is characterized by a feature vector $\mathbf{X}_v$ that includes spatial coordinates (x, y), a depot indicator, and demand. Edges $e_{uv} \in \mathcal{E}$ have corresponding features $\mathbf{X}_{e_{uv}}$, defined as the normalized Euclidean distance between nodes u and v. These node and edge feature vectors precisely capture positional and relational information necessary for effective message passing in subsequent layers.

The `Encoder` transforms initial node features into latent embeddings through a sequential stack of four identical layers, each consisting of a Graph Transformer (GT) layer, Layer Normalization (LayerNorm) (Ba et al., 2016), and an Exponential Linear Unit (ELU) (Clevert et al., 2016) activation. Each GT layer, inspired by Shi et al. (2021), leverages multi-head self-attention to aggregate node and edge information, updating node embeddings $\mathbf{h}'_v$ as follows:

$$\mathbf{h}'_v = \mathbf{W}^r \mathbf{h}_v + \sum_{u \in \mathcal{V} \setminus \{v\}} a_{uv} \left(\mathbf{W}^v \mathbf{h}_u + \mathbf{W}^e \mathbf{X}_{e_{uv}} \right),$$

where $\mathbf{W}^r$, $\mathbf{W}^v$, and $\mathbf{W}^e$ are learnable weight matrices corresponding to residual connections, node features, and edge features, respectively. Attention coefficients a_{uv} quantify node interactions:

$$a_{uv} = \text{softmax} \left(\frac{(\mathbf{W}^q \mathbf{h}_v)^\top (\mathbf{W}^k \mathbf{h}_u + \mathbf{W}^e \mathbf{X}_{e_{uv}})}{\sqrt{d}} \right),$$

with query and key matrices $\mathbf{W}^q, \mathbf{W}^k$ and embedding dimensionality d. LayerNorm stabilizes training by normalizing features across nodes, preserving structural information through adaptive scaling and shifting. The final updated node embedding per layer is:

$$\mathbf{h}'_v = \text{ELU}(\text{LayerNorm}(\text{GT}(\mathbf{h}_v))).$$

A key advantage of both architectures is that they can process customer sets S of arbitrary size beyond those that may have been seen during training. Moreover, the feature extractor network $\hat{\phi}$ can be quite complex and does not even have to be ReLU-activated, as only the regressor network $\hat{\rho}$ is embedded in the MIP model. This modularity allows for flexibility in the feature extraction stage while maintaining the tractability of the MIP embedding. The detailed hyperparameter search spaces for both the FFNN and GT architectures are provided in Appendix G (Tables 6 and 7, respectively).

B.2 Regressor $\hat{\rho}$

The final prediction in our model is obtained by aggregating node-level representations through a global additive pooling operation, followed by a linear transformation. This architecture aligns closely with the Deep Sets framework (Zaheer et al., 2017), which provides a principled way to model functions over sets that are invariant to permutation. In the context of our surrogate cost model, the function mapping a customer set $S \subseteq \mathcal{V}_C$ assigned to depot $d \in \mathcal{V}_D$ can be expressed as:

$$\hat{c}_d(S) = \hat{\rho}\left(\sum_{c \in S} \hat{\phi}(\sigma_{di})\right),$$

where $\sigma_{di} \in \mathbb{R}^4$ denotes the normalized feature vector. The transformation $\hat{\phi}$ projects each customer feature into a latent space, and the summation operator aggregates this information over the entire set. The output is then passed through the regression function $\hat{\rho}$, which produces the final cost estimate.

In our network implementation, the encoder serves as the learnable embedding function ϕ, while the global additive pooling layer (Graph Readout in Figure 3) performs the permutation-invariant aggregation. The linear decoder layer approximates ρ, translating the aggregated embedding into a scalar prediction. This structure naturally captures the set-based nature of depot-customer assignments, enabling the network to generalize across varying customer configurations and preserve consistency with the routing cost structure. Moreover, it ensures that the learned surrogate can be seamlessly embedded into the upper-level CLRP formulation without compromising tractability or representational fidelity.

The `Decoder` predicts travel costs and fixed vehicle costs by aggregating relevant node embeddings. Initially, a *Mask Gate* selectively retains embeddings from active nodes (depot and assigned customers), denoted by the mask vector mask_v:

$$\mathbf{h}_v^{\text{masked}} = \mathbf{h}'_v \odot \text{mask}_v.$$

Then, the masked node embeddings are aggregated through a global additive pooling operation, yielding a single graph-level representation:

$$\mathbf{h}_{\mathcal{G}_d} = \sum_{v \in \mathcal{V}_d} \mathbf{h}_v^{\text{masked}}.$$

This global additive pooling effectively summarizes node features into a unified embedding, reflecting overall graph structure and node relationships. Finally, a linear transformation maps this aggregated representation to the scalar predicted cost:

$$\hat{c}_d(S) = \mathbf{W}\mathbf{h}_{\mathcal{G}_d} + b,$$

where $\mathbf{W}$ and b are learnable parameters. This architecture ensures that the predicted travel and fixed vehicle costs accurately reflect both local and global structural properties of the depot-customer graph $\mathcal{G}_d$, which is essential for cost estimation across varying customer configurations.

B.3 Benchmark Instances

We use the CLRP benchmark set from (Prins et al., 2004), known as the Prodhon instances. It includes 30 instances with 20–200 customers and 5–10 depots, featuring vehicle capacities of 70 or 150 and constrained depot capacities. The set contains both clustered and randomly distributed customers.

B.4 Data Generation

To train our neural embedded models, we generated 128,000 synthetic CVRP instances via random sampling, following the structure of the Prodhon benchmark (Prins et al., 2004). Depots and customers were randomly placed in a 2D space, with demands uniformly sampled from [11, 20] and vehicle capacities set to 70 or 150. Feasible subsets were generated by ensuring depot and vehicle capacity constraints.

Each instance was solved using VROOM (Coupey et al., 2024) to obtain cost labels, incorporating both routing distances and fixed vehicle costs. Coordinates were normalized by shifting the depot to the origin and scaling by the spatial range. The final input features included normalized coordinates, depot indicators, and scaled demands. A smooth L1 loss was used during training to ensure stable convergence.

C Detailed Comparison with State-of-the-Art Algorithms

Table 2: Comparison with State-of-the-Art Heuristics on CLRP Benchmark Sets

		HCC-500K		TSBA$_{\text{speed}}$		GRASP/VNS		NEO-FFNN (ours)		NEO-GT (ours)	
Instance	BKS	$E_{\text{BKS}}^{\text{gap}}$ (%)	T_{total} (s)	$E_{\text{BKS}}^{\text{gap}}$ (%)	T_{total} (s)	$E_{\text{BKS}}^{\text{gap}}$ (%)	T_{total} (s)	$E_{\text{BKS}}^{\text{gap}}$ (%)	T_{total} (s)	$E_{\text{BKS}}^{\text{gap}}$ (%)	T_{total} (s)
20-5-1a	54,793	0.00	39	0.00	0.80	0.08	0.78	8.54	0.09	3.24	0.10
20-5-1b	39,104	0.00	54	0.00	0.53	0.00	0.67	6.58	0.04	6.30	0.05
20-5-2a	48,908	0.00	38	0.00	0.74	0.00	0.76	5.04	0.05	5.04	0.04
20-5-2b	37,542	0.00	67	0.00	0.51	0.00	0.65	4.04	0.05	8.25	0.05
Average		0.00	49.50	0.00	0.65	0.02	0.71	6.05	0.06	5.70	0.06
50-5-1a	90,111	0.00	101	0.00	2.48	0.00	7.95	4.12	0.10	5.37	0.10
50-5-1b	63,242	0.00	65	0.00	2.35	0.00	8.59	2.91	0.10	1.18	0.10
50-5-2a	88,293	0.32	99	0.06	3.32	0.35	8.52	2.81	0.12	3.03	0.08
50-5-2b	67,308	0.21	200	0.14	3.07	0.54	9.18	7.10	0.09	6.65	0.08
50-5-2bBIS	51,822	0.03	98	0.08	2.70	0.02	8.98	5.80	0.08	5.29	0.06
50-5-2BIS	84,055	0.08	107	0.00	3.40	0.00	7.90	1.62	0.10	1.68	0.10
50-5-3a	86,203	0.07	101	0.19	3.34	0.19	7.78	1.86	0.11	0.73	0.11
50-5-3b	61,830	0.00	137	0.01	2.35	0.00	7.59	1.26	0.10	1.68	0.10
Average		0.09	113.50	0.06	2.88	0.14	8.31	3.44	0.10	3.20	0.09
100-5-1a	274,814	0.56	520	0.37	15.14	0.44	70.15	2.16	0.37	1.28	0.32
100-5-1b	213,568	0.69	1190	0.50	11.68	0.38	70.81	2.26	0.31	1.25	0.32
100-5-2a	193,671	0.12	463	0.07	11.86	0.23	82.00	2.83	0.43	0.39	0.47
100-5-2b	157,095	0.04	859	0.05	8.11	0.07	61.93	0.79	0.44	0.59	0.46
100-5-3a	200,079	0.21	454	0.21	14.05	0.24	64.37	1.49	0.37	1.34	0.46
100-5-3b	152,441	0.30	684	0.03	8.39	1.03	57.29	1.57	0.36	2.27	0.34
100-10-1a	287,661	4.28	210	0.24	25.54	0.61	78.81	2.24	0.34	2.80	0.34
100-10-1b	230,989	4.03	188	0.47	16.57	1.19	87.95	4.29	0.31	3.61	0.25
100-10-2a	243,590	0.80	136	0.05	21.16	2.06	75.65	0.91	0.43	1.09	0.33
100-10-2b	203,988	0.25	261	0.00	10.93	1.23	67.50	1.82	0.39	3.58	0.30
100-10-3a	250,882	1.59	202	0.93	22.60	3.83	71.87	1.92	0.43	2.10	0.37
100-10-3b	203,114	1.51	224	0.29	14.88	5.53	79.76	3.37	0.44	3.35	0.36
Average		1.20	449.25	0.27	15.08	1.40	72.34	2.14	0.38	1.97	0.36
200-10-1a	474,850	1.76	752	0.62	179.62	3.16	752.03	0.97	1.47	0.87	1.29
200-10-1b	375,177	1.43	1346	0.42	115.72	2.74	735.75	1.25	1.36	1.23	1.25
200-10-2a	448,077	0.82	1201	0.35	147.04	0.38	642.16	0.40	1.23	0.47	1.13
200-10-2b	373,696	0.65	1349	0.14	69.52	0.23	683.19	0.44	1.33	0.54	1.20
200-10-3a	469,433	2.12	1251	0.49	176.25	0.48	661.82	1.36	1.45	1.20	1.32
200-10-3b	362,320	2.01	1137	0.14	67.58	0.45	818.25	3.73	1.34	1.52	1.32
Average		1.47	1172.67	0.36	125.96	1.24	715.53	1.36	1.36	0.97	1.25
Processor			Opteron 275		Xeon E5-2670		Xeon E5-2430v2		Ryzen 9 5900X		Ryzen 9 5900X
GHz			2.2		2.6		2.5		3.7		3.7
Passmark score			1159		1652		1439		3470		3470

D Neural Embedded FLP-MIP Formulation for 2E-CVRP

This presents the complete mixed-integer programming (MIP) formulation of the Two-Echelon Capacitated Vehicle Routing Problem (2E-CVRP) augmented with a neural-embedded surrogate model for second-echelon routing cost prediction. The formulation integrates neural predictions into a first-stage facility location model with satellite opening and customer assignment decisions.

Problem Description

The Two-Echelon Capacitated Vehicle Routing Problem (2E-CVRP) is an NP-hard variant of the Capacitated Vehicle Routing Problem (CVRP), characterized by a hierarchical distribution structure involving a single depot, multiple satellites, and customers. Vehicles operate over two distinct echelons: the first echelon connects the depot to satellites, and the second echelon links satellites to customers. Vehicles in the first echelon, commonly referred to as trucks or primary vehicles, typically have larger capacities compared to second-echelon vehicles, known as city freighters or secondary vehicles. Each vehicle type has its own fixed capacity and fleet size.

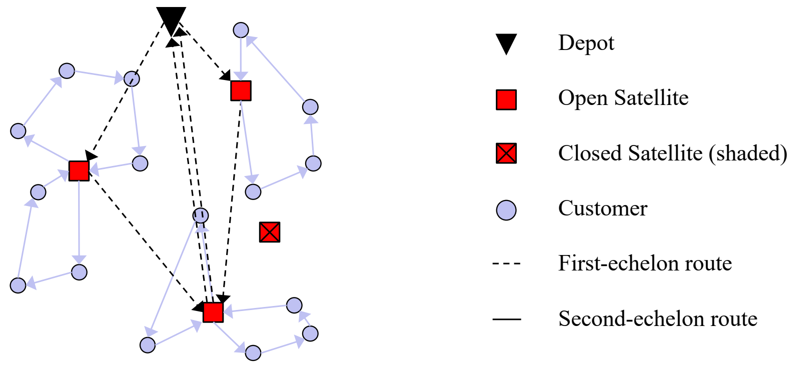

Figure 4: An Example of a 2E-CVRP Solution

Figure 4 illustrates a feasible 2E-CVRP solution, depicting the depot (triangle), satellites (squares), and customers (circles). In the first echelon, split deliveries to satellites are permitted, resembling the structure of the Split Delivery Vehicle Routing Problem (SDVRP). Conversely, split deliveries are disallowed in the second echelon, reflecting a Capacitated Location-Routing Problem (CLRP) structure. Moreover, freight transfers are restricted to echelon-specific movements, prohibiting direct delivery from the depot to customers.

Due to the hierarchical nature of 2E-CVRP, second-echelon decisions directly influence the first echelon. Thus, solving the 2E-CVRP involves strategic decisions, such as selecting operational satellites, and tactical decisions, such as vehicle routing in each echelon. The primary objective is to minimize the total transportation cost across both echelons.

Model Parameters

- $\mathcal{V}_0$: Set of depot nodes (typically a singleton set $\{0\}$).
- $\mathcal{V}_S$: Set of satellite nodes.
- $\mathcal{V}_C$: Set of customer nodes.
- $\mathcal{K}$: Set of first-echelon vehicle indices, with $|\mathcal{K}| = K$.
- $\mathcal{A}$: Set of arcs between depot and satellites (first echelon).
- C: Capacity of first-echelon vehicles.
- C': Capacity of second-echelon vehicles (city freighters).
- K': Maximum number of second-echelon vehicles available.
- d_i: Demand of customer $i \in \mathcal{V}_C$.
- c_{ij}: Travel cost on arc $(i, j) \in \mathcal{A}$.
- M: A sufficiently large constant used in big-M constraints.
- L: Set of latent dimensions for neural embeddings.
- $\phi_{si\ell}$: Neural network weight associated with customer i and satellite s in dimension ℓ.
- $\phi_{s0\ell}$: Neural bias term associated with satellite s in dimension ℓ.
- W_ℓ: Neural readout weight for latent dimension ℓ.
- b: Neural readout bias term.

Decision Variables

- $x_{ijk} \in \{0,1\}$: Binary variable indicating whether vehicle $k \in \mathcal{K}$ traverses arc $(i,j) \in \mathcal{A}$ in the first echelon.
- $w_{sk} \in \mathbb{R}_+$: Freight quantity delivered to satellite $s \in \mathcal{V}_S$ by vehicle $k \in \mathcal{K}$.
- $u_{ik} \in \mathbb{R}_+$: Sequence position of node $i \in \mathcal{V}_S$ visited by vehicle $k \in \mathcal{K}$ (for subtour elimination).
- $a_{si} \in \{0,1\}$: Binary variable indicating if customer $i \in \mathcal{V}_C$ is assigned to satellite $s \in \mathcal{V}_S$.
- $y_s \in \{0,1\}$: Binary variable indicating whether satellite $s \in \mathcal{V}_S$ is opened.
- $k_s \in \mathbb{Z}_+$: Number of city freighters assigned to satellite s.
- $z_{s\ell} \in \mathbb{R}$: Aggregated latent embedding of satellite s in dimension $\ell \in L$.
- $\tilde{c}_s \in \mathbb{R}_+$: Surrogate cost prediction for second-echelon routing and vehicle cost from satellite s.
- $t_s \in \mathbb{R}_+$: Total demand served by satellite s.

Model Formulation

$$
\begin{aligned}
\min \quad & \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} c_{ij}\, x_{ijk} + \sum_{s\in\mathcal{V}_S} \tilde{c}_s && && (25)\\
\text{s.t.} \quad & \sum_{(i,j)\in\delta^+(s)} x_{ijk} = \sum_{(i,j)\in\delta^-(s)} x_{ijk} && \forall s\in\mathcal{V}_S,\ k\in\mathcal{K} && (26)\\
& \sum_{(i,j)\in\delta^+(s)} x_{ijk} \le 1 && \forall s\in\mathcal{V}_0\cup\mathcal{V}_S,\ k\in\mathcal{K} && (27)\\
& u_{ik} + 1 \le u_{jk} + M\big(1 - x_{ijk}\big) && \forall (i,j)\in\mathcal{A}(\mathcal{V}_S),\ k\in\mathcal{K} && (28)\\
& w_{sk} \le C \sum_{(i,j)\in\delta^+(s)} x_{ijk} && \forall s\in\mathcal{V}_S,\ k\in\mathcal{K} && (29)\\
& \sum_{s\in\mathcal{V}_S} w_{sk} \le C && \forall k\in\mathcal{K} && (30)\\
& \sum_{k\in\mathcal{K}} w_{sk} = t_s && \forall s\in\mathcal{V}_S && (31)\\
& \sum_{s\in\mathcal{V}_S} a_{si} = 1 && \forall i\in\mathcal{V}_C && (32)\\
& a_{si} \le y_s && \forall s\in\mathcal{V}_S,\ i\in\mathcal{V}_C && (33)\\
& \sum_{i\in\mathcal{V}_C} a_{si} \ge y_s && \forall s\in\mathcal{V}_S && (34)\\
& t_s = \sum_{i\in\mathcal{V}_C} d_i\, a_{si} && \forall s\in\mathcal{V}_S && (35)\\
& k_s\, C' \ge t_s && \forall s\in\mathcal{V}_S && (36)\\
& t_s \ge C'\big(k_s - 1\big) + 1 && \forall s\in\mathcal{V}_S && (37)\\
& k_s \le K'\, y_s && \forall s\in\mathcal{V}_S && (38)\\
& \sum_{s\in\mathcal{V}_S} k_s \le K' && && (39)\\
& z_{s\ell} = \phi_{s0\ell} + \sum_{i\in\mathcal{V}_C} \phi_{si\ell}\, a_{si} && \forall s\in\mathcal{V}_S,\ \ell\in\mathcal{L} && (40)\\
& \tilde{c}_s \le M\, y_s && \forall s\in\mathcal{V}_S && (41)\\
& \tilde{c}_s \ge \sum_{\ell\in\mathcal{L}} W_\ell z_{s\ell} + b - M\big(1 - y_s\big) && \forall s\in\mathcal{V}_S && (42)
\end{aligned}
$$

$$\tilde{c}_s \leq \sum_{\ell \in \mathcal{L}} W_\ell z_{s\ell} + b + M\big(1 - y_s\big) \qquad \forall s \in \mathcal{V}_S \tag{43}$$

$$x_{ijk},\, a_{si},\, y_s \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A},\ s \in \mathcal{V}_S,\ k \in \mathcal{K} \tag{44}$$

$$k_s \in \mathbb{Z}_+ \qquad \forall s \in \mathcal{V}_S \tag{45}$$

$$w_{sk},\, u_{ik},\, t_s,\, z_{s\ell},\, \tilde{c}_s \geq 0 \qquad \forall s \in \mathcal{V}_S,\ i \in \mathcal{V}_S,\ k \in \mathcal{K},\ \ell \in \mathcal{L} \tag{46}$$

This formulation maintains tractability while integrating neural cost surrogates directly into the MIP model. The second-echelon cost term $\tilde{c}_s$ replaces the need for arc-based routing variables, enabling efficient evaluation of location-routing trade-offs.

E Variable Count Tables

Table 3: Exact CLRP vs. Neural-Embedded FLP – Binary Variables

Variable	Description	(20, 5)		(50, 5)		(100, 5)		(100, 10)		(200, 10)	
		Exact	Neural	Exact	Neural	Exact	Neural	Exact	Neural	Exact	Neural
y_d	Depot open	5	5	5	5	5	5	10	10	10	10
x_{ijd}	Arc-level Routing	3,000	–	14,850	–	54,600	–	119,900	–	438,900	–
x_{di}	Customer assignment	–	100	–	250	–	500	–	1,000	–	2,000
Total binary vars.		3,005	105	14,855	255	54,605	505	119,910	1,010	438,910	2,010
Reduction ($\times$)		28.6		58.3		108.1		118.7		218.4	

Table 4: Exact CLRP vs. Neural-Embedded FLP – Continuous Variables

Variable	Description	(20, 5)		(50, 5)		(100, 5)		(100, 10)		(200, 10)	
		Exact	Neural	Exact	Neural	Exact	Neural	Exact	Neural	Exact	Neural
f_{ijd}	Arc load	3,000	–	14,850	–	54,600	–	119,900	–	438,900	–
$z_{d\ell}$	Latent aggregation	–	320	–	320	–	320	–	640	–	640
$\tilde{c}_d$	NN-predicted cost	–	5	–	5	–	5	–	10	–	10
Total continuous vars.		3,000	325	14,850	325	54,600	325	119,900	650	438,900	650
Reduction ($\times$)		9.2		45.7		168.0		184.5		675.2	

Counting rules. C: #customers, D: #depots, $N = C + D$. Exact: binary $D + D\,N(N{-}1)$, continuous $D\,N(N{-}1)$; Neural: binary $D + D\,C$, continuous $D(L{+}1)$ with $L = 64$.

F Ablation Studies

We perform detailed experimental analyses to understand the impact of various components of our framework, including the choice of sampling method, sample requirements for training, choice of vehicle routing solvers for generating labeled training data, as well as the use of single pre-trained versus instance-specific neural surrogates. All results are averaged over five runs.

F.1 Effect of Problem Size

To analyze the impact of instance complexity, we assess how the number of customers affects the performance of our model. Specifically, we evaluate NEO-LRP on CLRP instances with 20, 50, 100, and 200 customers, using the same training and evaluation setup. For each problem size, performance metrics are averaged over five independent runs.

Table 1(b) shows that the average optimality gap $E^{\text{gap}}_{\text{BKS}}$ tends to decrease as the problem size increases. This trend can be attributed to two main factors. First, the relative impact of individual assignment errors becomes less significant in larger instances, as routing costs are distributed over a greater number of customers. Second, the model's capacity to generalize improves with scale, especially when structural patterns—such as regional cost heterogeneity or spatial dependencies—become more pronounced in larger graphs.

These results indicate that NEO-LRP scales well with instance size and is capable of exploiting structural regularities in large-scale location-routing problems. Overall, the observed improvement in performance highlights a desirable property of neural-embedded optimization frameworks: when trained on a sufficiently diverse dataset, a single model can generalize effectively to larger and more complex instances, without needing instance-specific retraining.

F.2 Effect of Routing Solver

To generate high-quality ground-truth labels within reasonable computation times, we evaluated three representative CVRP solvers: VRPSolverEasy (Errami et al., 2024), OR-Tools (Furnon and Perron, 2024), and VROOM. Table 5 presents the computational results obtained across instances with varying customer sizes (n) and vehicle capacities (c). VRPSolverEasy, which relies on exact optimization, consistently returned optimal solutions (BKS) but required excessive computation time for larger instances (e.g., $n = 200$), making it unsuitable for large-scale data generation. OR-Tools was configured with the `AUTOMATIC` setting for initial solution construction and the `GUIDED_LOCAL_SEARCH` (GLS) metaheuristic for local search. GLS was chosen as it is widely regarded as the most efficient general-purpose improvement method for vehicle routing problems. A fixed time limit of 5 seconds was imposed on each run. While OR-Tools produced solutions with acceptable runtimes, the resulting cost gaps were relatively higher compared to other methods. VROOM, in contrast, achieved a better balance between solution quality and speed. It consistently produced near-optimal solutions, typically with gaps under 2%, and completed all instances within practical time limits. Given this trade-off, VROOM was selected as the default solver for generating routing labels during neural model training.

Table 5: Comparison of CVRP Routing Solvers

Instance	VRPSolverEasy		OR-Tools		VROOM	
	BKS	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
n20-c70	34 961	0.07	0.00	5.00	0.00	0.02
n20-c150	25 285	0.14	0.00	5.00	0.00	0.02
n50-c70	71 195	4.54	1.06	5.00	0.72	0.13
n50-c150	42 831	4.15	1.93	5.00	0.00	0.14
n100-c70	135 190	290.31	5.21	5.00	0.93	0.68
n100-c150	76 201	131.10	7.53	5.00	0.00	0.63
n200-c70	247 511	37 173.32	4.34	5.00	1.81	3.86
n200-c150	132 874	25 017.05	8.19	5.00	1.43	3.92

G Hyperparameters

Table 6: FFNN Hyperparameters and Their Search Ranges

Hyperparameter	Range
Batch size	{8, 16, 32}
Initial learning rate	{0.1, 0.01, 0.001, 0.0001}
Optimizer	AdamW
Loss function	{MSE, Huber, Smooth L1}
Hidden dimension in $\hat{\phi}$	{6, 8, 32, 64, 128, 256, 512, 1024}
Latent dimension	{6, 8, 16, 32}
Number of hidden layers in $\hat{\phi}$	{2, 3, 4, 5, 6}
Number of hidden layers in $\hat{\rho}$	{1, 2, 3, 4}

Table 7: GT Hyperparameters and Their Search Ranges

Hyperparameter	Range
Initial learning rate	$\{0.1, 0.01, 0.001, 0.0001\}$
Batch size	$\{8, 16, 32, 64, 128\}$
Optimizer	AdamW
Loss function	{MSE, Huber, Smooth L1}
Encoding dimension	$\{8, 16, 32, 64, 128\}$
Number of attention heads	$\{4, 8\}$
Number of GT layers	$\{2, 3, 4, 5\}$
Dropout rate	$\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$
Normalization	{Graph Norm, Batch Norm, Layer Norm}
Activation function	{ELU, ReLU, Leaky ReLU}
Beta	{True, False}