RRNCO: Towards Real-World Routing with Neural Combinatorial Optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

Vehicle Routing Problems (VRPs) are a class of NP-hard problems ubiquitous in several real-world logistics scenarios that pose significant challenges for optimization. Neural Combinatorial Optimization (NCO) has emerged as a promising alternative to classical approaches, as it can learn fast heuristics to solve VRPs. However, existing research works in NCO for VRPs learn from simplified, symmetric Euclidean settings, failing to handle the asymmetric distances and travel durations inherent to real-world road networks. This critical sim-to-real gap severely hinders their practical deployment. To address this fundamental limitation, we introduce RRNCO, a novel NCO architecture with two key innovations for handling real-world routing complexity. First, we propose an Adaptive Node Embedding (ANE) approach that fuses coordinate information with distance features through learned contextual gating. Unlike existing methods relying solely on spatial coordinates or requiring full distance matrix processing, our approach efficiently captures both local geometric structure and global routing constraints through probability-weighted distance sampling that prioritizes nearby nodes while preserving asymmetric relationships. Second, we introduce Neural Adaptive Bias (NAB), the first mechanism to jointly model asymmetric distance and duration matrices within a deep neural routing framework. NAB's gating-based architecture learns to dynamically integrate distance, duration, and directional angles into a unified contextual bias that guides the Adaptation Attention Free Module (AAFM). Together, these innovations enable RRNCO to explicitly capture real-world routing asymmetries where costs from location A to B differ from B to A due to traffic patterns, road directionality, and temporal dynamics. We validate our method on a newly constructed dataset featuring real-world asymmetric distance and duration matrices from 100 diverse cities. Experiments demonstrate that RRNCO achieves state-of-the-art performance among NCO methods on realistic VRPs. We release our dataset and code to advance research in practical neural combinatorial optimization.

1 Introduction

Vehicle Routing Problems (VRPs) are foundational NP-hard challenges in logistics, where routing efficiency improvements can yield substantial cost savings [1]. While traditional solvers exist [2, 3, 4, 5, 6, 7], their computational complexity and need for expert tuning limit their use in large-scale, real-time applications. Neural Combinatorial Optimization (NCO) has emerged as a promising data-driven paradigm, using Reinforcement Learning (RL) to learn fast and scalable heuristics for VRPs [8, 9, 10]. Despite impressive results on synthetic benchmarks [11, 12, 13, 14, 15, 16], a critical sim-to-real gap persists. Most NCO research relies on simplified Euclidean datasets, failing to capture the asymmetric travel times and distances ($d_{ij} \neq d_{ji}$) inherent to real-world road networks

[17, 18]. Furthermore, dominant NCO architectures, often based on node-centric Transformers [19], struggle to efficiently embed the rich, asymmetric edge features (e.g., distance and duration matrices) crucial for realistic routing problems [20]. A detailed discussion of related work is provided in the Appendix C

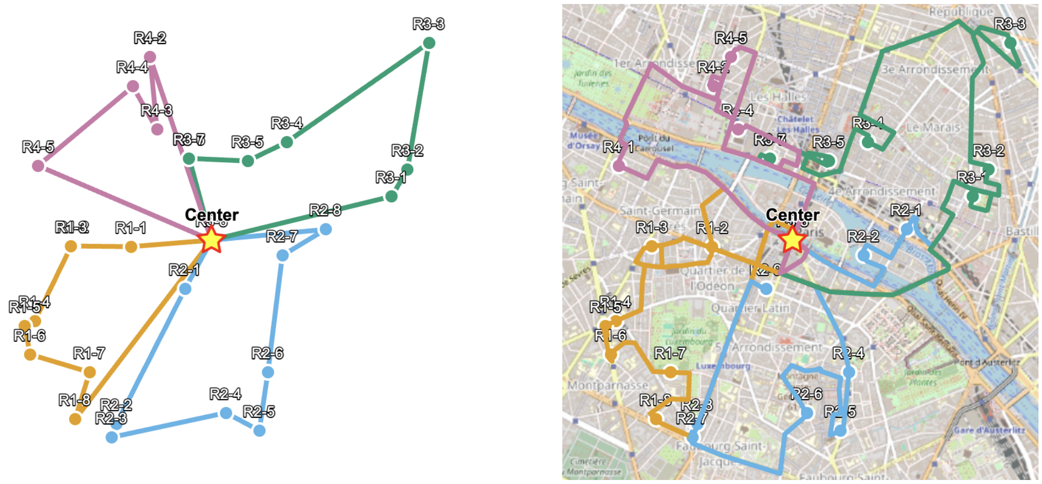

Figure 1: [Left] Most NCO works consider simplified Euclidean settings. [Right] Our work models real-world instances where durations and travel times can be asymmetric.

Our Real Routing NCO (RRNCO) bridges this gap—as illustrated in Fig. 1—through innovations in both modeling and data. We introduce a novel neural architecture with two key technical contributions: (i) an **Adaptive Node Embedding (ANE)** that dynamically fuses coordinates and distance information via learned contextual gating and probability-weighted sampling; and (ii) a **Neural Adaptive Bias (NAB)**, the first mechanism to jointly model asymmetric distance and duration matrices within a deep routing framework, guiding our Adaptation Attention Free Module (AAFM). To validate our approach, we construct a comprehensive benchmark dataset from 100 diverse cities, featuring real-world asymmetric distance and duration matrices from OpenStreetMap [21].

Our contributions are: (1) A novel NCO architecture (RRNCO) with ANE and NAB to natively handle real-world routing asymmetries. (2) An extensive, open-source VRP dataset from 100 cities with asymmetric matrices. (3) State-of-the-art empirical results on realistic VRP instances. (4) Open-source code and data to foster reproducible research.

2 Preliminaries: Solving VRPs with NCO

A VRP is defined on a graph $G = (V, E)$, where the goal is to find optimal routes. In real-world settings, the cost between nodes is asymmetric and multi-modal, represented by distance and duration matrices $D, T \in \mathbb{R}^{n \times n}$. We frame the VRP as a sequential decision process solved by a deep generative model using an autoregressive encoder-decoder framework [9]. The model constructs a solution $\boldsymbol{a}$ (a sequence of visited locations) for a given problem instance $\boldsymbol{x}$.

The policy is trained using reinforcement learning (RL) to discover effective heuristics without labeled data. Specifically, we optimize the policy parameters θ to maximize the expected reward $R(\boldsymbol{a}, \boldsymbol{x})$, which is the negative route cost:

$$\max_{\theta} J(\theta) = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}} \mathbb{E}_{\boldsymbol{a} \sim \pi_\theta(\cdot|\boldsymbol{x})}[R(\boldsymbol{a}, \boldsymbol{x})]. \tag{1}$$

We use the REINFORCE algorithm with the variance-reducing POMO baseline [11], a standard and effective training method for NCO routing solvers. This approach requires efficient generation of problem instances $\boldsymbol{x}$ to ensure training efficiency, a need met by our data generation framework.

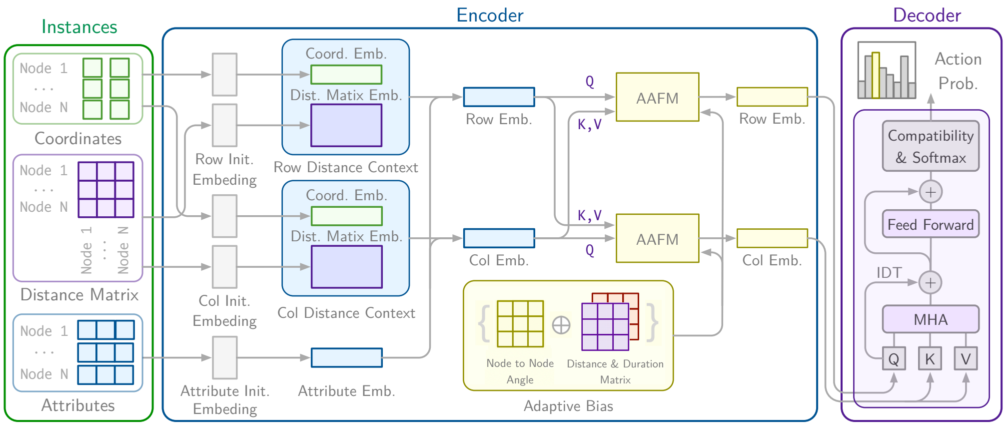

Figure 2: Our proposed RRNCO model for real-world routing.

3 The RRNCO Model

Our model, depicted in Fig. 2, features an encoder-decoder architecture designed to handle real-world routing complexities. Our innovations focus on the encoder, enhancing its ability to process asymmetric, multi-modal data efficiently. A more detailed description of the model architecture is provided in the Appendix A

3.1 Encoder

3.1.1 Adaptive Node Embedding (ANE)

The ANE module creates comprehensive node representations by fusing complementary spatial features: global distance matrix information and local coordinate-based geometry. To maintain computational efficiency while capturing the most relevant relationships, we employ a selective sampling strategy. For each node i, we sample k neighboring nodes with probability inversely proportional to their distance, $p_{ij} \propto 1/d_{ij}$. This prioritizes local structure. These sampled distances are then projected into an embedding $\mathbf{f}_{\text{dist}}$. Separately, raw coordinates are projected to capture geometric relationships, yielding an embedding $\mathbf{f}_{\text{coord}}$. We combine these complementary representations using a learned Contextual Gating mechanism:

$$\mathbf{g} = \sigma(\text{MLP}([\mathbf{f}_{\text{coord}}; \mathbf{f}_{\text{dist}}])) \tag{2}$$

$$\mathbf{h} = \mathbf{g} \odot \mathbf{f}_{\text{coord}} + (1 - \mathbf{g}) \odot \mathbf{f}_{\text{dist}} \tag{3}$$

This mechanism allows the model to adaptively weigh the importance of coordinate-based versus distance-based features for each node, enabling a more nuanced spatial representation. To effectively handle asymmetric routing scenarios, we follow the approach introduced in [20] and generate dual embeddings for each node: row embeddings $\mathbf{h}^r$ and column embeddings $\mathbf{h}^c$. These are then fused with other node features (e.g., demand) to produce the final combined representations for the encoder.

3.1.2 Neural Adaptive Bias (NAB) for AAFM

RRNCO uses an Adaption Attention-Free Module (AAFM) [22] to model inter-node relationships. While the original AAFM defines its adaptation bias A heuristically (e.g., based on log-distance), we introduce NAB, a mechanism that *learns* this bias directly from data. NAB is the first approach to jointly model multiple asymmetric matrices, processing a distance matrix $\mathbf{D}$, an angle matrix $\mathbf{\Phi}$ (for directional relationships), and an optional duration matrix $\mathbf{T}$. Each matrix is passed through an MLP to get embeddings $\mathbf{D}_{emb}$, $\mathbf{\Phi}_{emb}$, and $\mathbf{T}_{emb}$. These are then fused using a multi-channel contextual gating mechanism that learns to weigh each modality:

$$\mathbf{G} = \text{softmax}\left(\frac{[\mathbf{D}_{\text{emb}};\ \mathbf{\Phi}_{\text{emb}};\ \mathbf{T}_{\text{emb}}]\,\mathbf{W}_G}{\exp(\tau)}\right) \tag{4}$$

$$\mathbf{H} = \mathbf{G}_1 \odot \mathbf{D}_{\text{emb}} + \mathbf{G}_2 \odot \mathbf{\Phi}_{\text{emb}} + \mathbf{G}_3 \odot \mathbf{T}_{\text{emb}} \tag{5}$$

The fused representation $\mathbf{H}$ is projected to a scalar to form the final adaptive bias matrix $\mathbf{A} = \mathbf{H}\mathbf{w}_{out}$. This resulting matrix serves as a learned inductive bias that captures the complex interplay between distance, duration, and direction. This learned bias $\mathbf{A}$ is then used in the AAFM operation:

$$\text{AAFM}(Q, K, V, A) = \sigma(Q) \odot \frac{\exp(A) \cdot (\exp(K) \odot V)}{\exp(A) \cdot \exp(K)} \tag{6}$$

After several AAFM layers, this process yields final node representations that encode rich, asymmetric patterns from the real-world routing network.

3.2 Decoder

Our decoder architecture synthesizes designs from ReLD [23] and MatNet [20] to construct solutions autoregressively. At each step, it uses the encoder's rich node embeddings and a context vector representing the current partial route (e.g., last visited node, remaining capacity). A multi-head attention mechanism generates a query, which is then used in a compatibility layer to compute the selection probability for the next node. This layer incorporates a negative logarithmic distance heuristic, guiding the model to prioritize nearby feasible nodes, thereby efficiently exploring the solution space.

4 Real-World VRP Dataset

A primary barrier to practical NCO is the lack of realistic datasets. Existing benchmarks are typically synthetic and symmetric, failing to capture real-world complexities like one-way streets or traffic-dependent travel times. To bridge this gap, we developed a large-scale dataset for real-world VRPs. Our data generation pipeline uses the OpenStreetMap Routing Engine (OSRM) to create topological maps for 100 diverse cities worldwide, each with corresponding asymmetric distance and duration matrices. We also designed an efficient online subsampling method to generate a virtually unlimited number of VRP instances for training our RL agent, ensuring the data faithfully represents real-world challenges. The complete data generation methodology, including city selection criteria and our subsampling framework, is detailed in the Appendix B

5 Experiments

5.1 Experimental Setup

Classical Baselines. In the experiments, we compare three SOTA traditional optimization approaches: LKH3[24]: a heuristic algorithm with strong performance on (A)TSP problems, PyVRP[7]: a specialized solver for VRPs with comprehensive constraint handling capabilities; and Google OR-Tools[25]: a versatile optimization library for CO problems.

Learning-Based Methods. We compare against SOTA NCO methods divided in two categories. *1) Node-only encoding learning methods*: POMO[11], an end-to-end multi-trajectory RL-based method based on attention mechanisms; MTPOMO[26], a multi-task variant of POMO; MVMoE[27], a mixture-of-experts variant of MTPOMO; RF[28]: an RL-based foundation model for VRPs; ELG[29], a hybrid of local and global policies for routing problems; BQ-NCO[30]: a decoder-only transformer trained with supervised learning; LEHD[31]: a supervised learning-based heavy decoder model. *2) Node and edge encoding learning methods*: GCN[32]: a graph convolutional network with encoding of edge information for routing; MatNet[20]: an RL-based solver encoding edge features via matrices and GOAL[33]: a generalist agent trained via supervised learning for several CO problems, including routing problems.

Training Configuration. We perform training runs under the same settings for fair comparison for our model, MatNet for ATSP and ACVRP, and GCN for ACVRP. Node-only models do not necessitate retraining since our datasets are already normalized in the $[0, 1]^2$ coordinates ranges (with locations sampled uniformly), and we do not retrain supervised-learning models since they would necessitate labeled data. We train with the Adam optimizer with an initial learning rate of 4×10^{-4}, which decays by a factor of 0.1 at epochs 180 and 195. Training is completed within 24 hours on

Table 1: Performance comparison across real-world routing tasks and distributions. We report costs and gaps calculated with respect to best-known solutions ($*$) from traditional solvers. Horizontal lines separate traditional solvers, node-only methods, node-and-edge methods, and our RRNCO. Lower is better ($\downarrow$).

		In-distribution			Out-of-distribution (city)			Out-of-distribution (cluster)		
Task	**Method**	**Cost**	**Gap (%)**	**Time**	**Cost**	**Gap (%)**	**Time**	**Cost**	**Gap (%)**	**Time**
ATSP	LKH3	38.387	$*$	1.6h	38.903	$*$	1.6h	12.170	$*$	1.6h
	POMO	51.512	34.192	10s	50.594	30.051	10s	30.051	146.926	10s
	ELG	51.046	32.976	42s	50.133	28.866	42s	23.017	89.131	42s
	BQ-NCO	55.933	45.708	25s	54.739	40.706	25s	27.872	129.022	25s
	LEHD	56.099	46.140	13s	54.811	40.891	13s	27.819	128.587	13s
	MatNet	39.915	3.981	27s	40.548	4.228	27s	12.886	5.883	27s
	GOAL	41.976	9.350	91s	42.590	9.477	91s	13.654	12.194	91s
	Ours	39.077	1.797	22s	39.783	2.262	22s	12.450	2.301	22s
ACVRP	PyVRP	69.739	$*$	7h	70.488	$*$	7h	22.553	$*$	7h
	OR-Tools	72.597	4.097	7h	73.286	3.969	7h	23.576	4.538	7h
	POMO	85.888	23.156	16s	85.771	21.682	16s	34.179	51.549	16s
	MTPOMO	86.521	24.063	16s	86.446	22.640	16s	34.287	52.029	16s
	MVMoE	86.248	23.672	22s	86.111	22.164	22s	34.135	51.356	22s
	RF	86.289	23.731	17s	86.261	22.377	16s	34.273	51.967	16s
	ELG	85.951	23.247	67s	85.741	21.639	66s	34.027	50.873	67s
	BQ-NCO	93.075	33.462	30s	92.467	31.181	30s	40.110	77.848	30s
	LEHD	93.648	34.284	17s	93.195	32.214	17s	40.048	77.573	17s
	GCN	90.546	29.836	17s	90.805	28.823	17s	34.417	52.605	17s
	MatNet	74.801	7.258	30s	75.722	7.425	30s	24.844	10.158	30s
	GOAL	84.341	20.938	104s	84.097	19.307	104s	34.318	52.166	104s
	Ours	72.145	3.450	25s	72.999	3.562	25s	23.280	3.224	25s
ACVRPTW	PyVRP	118.056	$*$	7h	118.513	$*$	7h	39.253	$*$	7h
	OR-Tools	119.681	1.377	7h	120.147	1.379	7h	39.903	1.655	7h
	POMO	132.883	12.559	18s	132.743	12.007	17s	50.503	28.661	18s
	MTPOMO	133.135	12.773	17s	132.921	12.158	18s	50.372	28.328	18s
	MVMoE	132.871	12.549	24s	132.700	11.971	23s	50.333	28.227	24s
	RF	132.887	12.563	18s	132.731	11.997	18s	50.422	28.455	18s
	GOAL	134.699	14.098	107s	135.001	13.912	107s	47.966	22.197	107s
	Ours	122.693	3.928	35s	123.249	3.996	35s	41.077	4.647	35s

$4\times$ NVIDIA A100 40GB GPUs using a batch size of 256, processing 100,000 instances per epoch. The model follows a Attention Free Transformer(AFT) based architecture with 128-dimensional embeddings, 512-dimensional feedforward layers, and 12 AFT layers. The training dataset consists of 80 cities with instances randomly generated from subsampled real-world city base map topologies with the remaining 20 reserved for OOD testing.

Testing Protocol. The test data consists of in-distribution evaluation for 1) *In-dist*: new instances generated from the 80 cities seen during training, 2) *OOD (city)* out-of-distribution generalization over new city maps and 3) *OOD (cluster)* out-of-distribution generalization to new location distributions across maps. The test batch size is 32, and a data augmentation factor of 8 is applied to all models except supervised learning-based ones, i.e., LEHD, BQ-NCO, and GOAL. All evaluations are conducted on an NVIDIA A6000 GPU paired with an Intel(R) Xeon(R) CPU @ 2.20GHz.

5.2 Main Results

Table 1 presents the performance of our model against all baselines on Asymmetric TSP (ATSP), Asymmetric CVRP (ACVRP), and ACVRPTW. The results unequivocally demonstrate that RRNCO achieves state-of-the-art performance among all neural solvers across every task and distribution. It consistently finds higher-quality solutions (lower cost) while remaining computationally efficient. Notably, a single RRNCO model handles all VRP variants, showcasing its adaptability and strong generalization capabilities in both in-distribution and out-of-distribution scenarios.

Table 2: Comparison of routing solvers and their training data generators on real-world data.

Method	Data Gen.	In-dist		OOD City		OOD Clust.	
		Cost	Gap%	Cost	Gap%	Cost	Gap%
LKH3	–	38.39	$*$	38.90	$*$	12.17	$*$
MatNet	ATSP	80.86	110.70	81.04	108.30	27.78	128.23
RRNCO	Noise	41.35	7.72	42.01	7.98	13.66	12.20
MatNet	Real	39.92	3.98	40.55	4.23	12.89	5.88
RRNCO	Real	**39.08**	**1.80**	**39.78**	**2.26**	**12.45**	**2.30**

5.3 Analyses

Ablation Study. We perform an ablation study on our proposed model components in Fig. 3 to validate their contributions. We evaluate performance when using only raw coordinates, only sampled distances, our full Adaptive Node Embedding (ANE), and the complete model with the Neural Adaptive Bias (NAB). The results show that ANE and NAB perform the best, systematically improving solution quality. The improvement is particularly pronounced in the out-of-distribution (OOD) settings, highlighting their role in enhancing generalization. Remarkably, in the challenging OOD (cluster) distribution, the addition of NAB provides a relative improvement of over 15%, confirming its effectiveness in capturing complex, unseen spatial relationships.

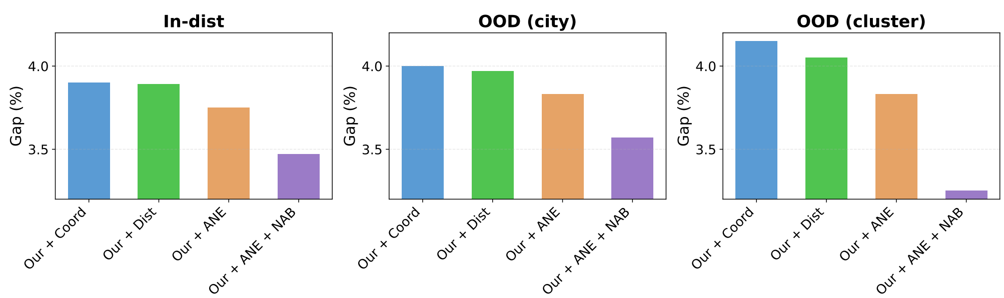

Figure 3: Study of our proposed model with different initial contexts: coordinates, distances, Adaptive Node Embedding (ANE), and Neural Adaptive Bias (NAB). ANE and NAB perform best, particularly in out-of-distribution (OOD) cases.

Importance of Real-World Data Generators. We study the impact of the training data generator on performance in real-world test settings. We compare models trained on a standard symmetric ATSP generator from MatNet [20], a generator that adds random noise to break symmetries, and our proposed real-world data generator. As shown in Table 2, training on data that mirrors real-world asymmetric properties is crucial. Models trained on symmetric or noisy data perform poorly when evaluated on our realistic benchmark. In contrast, training on our proposed real-world data leads to dramatic improvements in performance for both MatNet and our RRNCO model across all in-distribution and OOD settings, underscoring the necessity of our data generation framework to bridge the sim-to-real gap.

6 Conclusion

We introduced RRNCO, a novel NCO architecture designed to bridge the sim-to-real gap in vehicle routing. Our model explicitly handles the asymmetric and multi-modal travel costs of real-world networks through two key innovations: an Adaptive Node Embedding (ANE) that efficiently fuses coordinate and distance features, and a Neural Adaptive Bias (NAB) mechanism that jointly learns from distance, duration, and directional data. To validate our model, we built and are releasing a large-scale dataset with realistic routing instances from 100 cities. On this challenging benchmark, RRNCO achieves state-of-the-art performance among NCO methods. By open-sourcing our model and dataset, we aim to accelerate progress towards practical, deployable neural optimization solutions.

References

[1] Research and Markets. Size of the global logistics industry from 2018 to 2023, with forecasts until 2028 (in trillion u.s. dollars). `https://www.statista.com/statistics/943517/logistics-industry-global-cagr/`, 2024. Accessed: January 22, 2025. The forecast is based on a 2023 market size of approximately $9.41 trillion and a CAGR of 5.6%, which implies an estimated global industry value of roughly $10.5 trillion in 2025.

[2] Gilbert Laporte and Yves Nobert. Exact algorithms for the vehicle routing problem. In *North-Holland mathematics studies*, volume 132, pages 147–184. Elsevier, 1987.

[3] Thibaut Vidal. Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. *Computers & Operations Research*, 140:105643, 2022.

[4] Laurent Perron and Vincent Furnon. OR-Tools. Google, 2023.

[5] David Applegate, Robert Bixby, Václav Chvátal, and William Cook. Concorde TSP solver. `http://www.math.uwaterloo.ca/tsp/concorde.html`, 2003.

[6] Niels A Wouda and Leon Lan. Alns: A python implementation of the adaptive large neighbourhood search metaheuristic. *Journal of Open Source Software*, 8(81):5028, 2023.

[7] Niels A Wouda, Leon Lan, and Wouter Kool. PyVRP: A high-performance VRP solver package. *INFORMS Journal on Computing*, 2024.

[8] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

[9] Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems! *International Conference on Learning Representations*, 2019.

[10] Xuan Wu, Di Wang, Lijie Wen, Yubin Xiao, Chunguo Wu, Yuesong Wu, Chaoyu Yu, Douglas L Maskell, and You Zhou. Neural combinatorial optimization algorithms for solving vehicle routing problems: A comprehensive survey with perspectives. *arXiv preprint arXiv:2406.00415*, 2024.

[11] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

[12] Minsu Kim, Junyoung Park, and Jinkyoo Park. Sym-NCO: Leveraging symmetricity for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 35:1936–1949, 2022.

[13] Fu Luo, Xi Lin, Zhenkun Wang, Tong Xialiang, Mingxuan Yuan, and Qingfu Zhang. Self-improved learning for scalable neural combinatorial optimization. *arXiv preprint arXiv:2403.19561*, 2024.

[14] Haoran Ye, Jiarui Wang, Helan Liang, Zhiguang Cao, Yong Li, and Fanzhang Li. Glop: Learning global partition and local construction for solving large-scale routing problems in real-time. *AAAI 2024*, 2024.

[15] Jianan Zhou, Yaoxin Wu, Zhiguang Cao, Wen Song, and Jie Zhang. Collaboration! Towards robust neural methods for vehicle routing problems. 2023.

[16] André Hottung, Paula Wong-Chung, and Kevin Tierney. Neural deconstruction search for vehicle routing problems. *arXiv preprint arXiv:2501.03715*, 2025.

[17] Eneko Osaba. Benchmark dataset for the Asymmetric and Clustered Vehicle Routing Problem with Simultaneous Pickup and Deliveries, Variable Costs and Forbidden Paths. *Data in Brief*, 29:105142, January 2020.

[18] Daniela Thyssens, Tim Dernedde, Jonas K Falkner, and Lars Schmidt-Thieme. Routing arena: A benchmark suite for neural routing solvers. *arXiv preprint arXiv:2310.04140*, 2023.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[20] Yeong-Dae Kwon, Jinho Choo, Iljoo Yoon, Minah Park, Duwon Park, and Youngjune Gwon. Matrix encoding networks for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 34:5138–5149, 2021.

[21] OpenStreetMap contributors. Openstreetmap database [data set]. `https://www.openstreetmap.org`, 2025. Accessed: 2025-02-06.

[22] Changliang Zhou, Xi Lin, Zhenkun Wang, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. Instance-conditioned adaptation for large-scale generalization of neural combinatorial optimization. *arXiv preprint arXiv:2405.01906*, 2024.

[23] Ziwei Huang, Jianan Zhou, Zhiguang Cao, and Yixin XU. Rethinking light decoder-based solvers for vehicle routing problems. In *The Thirteenth International Conference on Learning Representations*, 2025.

[24] Keld Helsgaun. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, 12:966–980, 2017.

[25] Laurent Perron and Frédéric Didier. CP-SAT, 2024.

[26] Fei Liu, Xi Lin, Qingfu Zhang, Xialiang Tong, and Mingxuan Yuan. Multi-task learning for routing problem with cross-problem zero-shot generalization. *arXiv preprint arXiv:2402.16891*, 2024.

[27] Jianan Zhou, Zhiguang Cao, Yaoxin Wu, Wen Song, Yining Ma, Jie Zhang, and Chi Xu. MVMoE: Multi-task vehicle routing solver with mixture-of-experts. In *International Conference on Machine Learning*, 2024.

[28] Federico Berto, Chuanbo Hua, Nayeli Gast Zepeda, André Hottung, Niels Wouda, Leon Lan, Junyoung Park, Kevin Tierney, and Jinkyoo Park. Routefinder: Towards foundation models for vehicle routing problems. *arXiv preprint arXiv:2406.15007*, 2024.

[29] Chengrui Gao, Haopu Shang, Ke Xue, Dong Li, and Chao Qian. Towards generalizable neural solvers for vehicle routing problems via ensemble with transferrable local policy. *IJCAI*, 2024.

[30] Darko Drakulic, Sofia Michel, Florian Mai, Arnaud Sors, and Jean-Marc Andreoli. Bq-nco: Bisimulation quotienting for generalizable neural combinatorial optimization. *arXiv preprint arXiv:2301.03313*, 2023.

[31] Fu Luo, Xi Lin, Fei Liu, Qingfu Zhang, and Zhenkun Wang. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. *Advances in Neural Information Processing Systems*, 36:8845–8864, 2023.

[32] Lu Duan, Yang Zhan, Haoyuan Hu, Yu Gong, Jiangwen Wei, Xiaodong Zhang, and Yinghui Xu. Efficiently Solving the Practical Vehicle Routing Problem: A Novel Joint Learning Approach. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3054–3063, Virtual Event CA USA, August 2020. ACM.

[33] Darko Drakulic, Sofia Michel, and Jean-Marc Andreoli. Goal: A generalist combinatorial optimization agent learner. *arXiv preprint arXiv:2406.15079*, 2024.

[34] Andrius Barauskas, Agnundefined Brilingaitundefined, Linas Bukauskas, Vaida Čeikutundefined, Alminas Čivilis, and Simonas Šaltenis. Test-data generation and integration for long-distance e-vehicle routing. *Geoinformatica*, 27(4):737–758, January 2023.

[35] Aldy Gunawan, Graham Kendall, Barry McCollum, Hsin Vonn Seow, and Lai Soon Lee. Vehicle routing: Review of benchmark datasets. *Journal of the Operational Research Society*, 72:1794 – 1807, 2021.

[36] Dennis Luxen and Christian Vetter. Real-time routing with openstreetmap data. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS ’11, pages 513–516, New York, NY, USA, 2011. ACM.

[37] Nitin R Chopde and Mangesh Nichat. Landmark based shortest path detection by using a* and haversine formula. *International Journal of Innovative Research in Computer and Communication Engineering*, 1(2):298–302, 2013.

[38] Hina Ali and Khalid Saleem. Generating large-scale real-world vehicle routing dataset with novel spatial data extraction tool. *PLOS ONE*, 19(6):e0304422, June 2024.

[39] Fahui Wang and Yanqing Xu. Estimating o–d travel time matrix by google maps api: implementation, advantages, and implications. *Annals of GIS*, 17(4):199–209, 2011.

[40] Ao Qu and Cathy Wu. Revisiting the correlation between simulated and field-observed conflicts using large-scale traffic reconstruction. *Accident Analysis & Prevention*, 210:107808, 2025.

[41] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.

[42] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[43] Wouter Kool, Herke van Hoof, Joaquim Gromicho, and Max Welling. Deep policy dynamic programming for vehicle routing problems. In *International conference on integration of constraint programming, artificial intelligence, and operations research*, pages 190–213. Springer, 2022.

[44] Nathan Grinsztajn, Daniel Furelos-Blanco, Shikha Surana, Clément Bonnet, and Tom Barrett. Winner takes it all: Training performant rl populations for combinatorial optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

[45] André Hottung and Kevin Tierney. Neural large neighborhood search for the capacitated vehicle routing problem. *arXiv preprint arXiv:1911.09539*, 2019.

[46] Yining Ma, Jingwen Li, Zhiguang Cao, Wen Song, Le Zhang, Zhenghua Chen, and Jing Tang. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. *Advances in Neural Information Processing Systems*, 34:11096–11107, 2021.

[47] Gerhard Reinelt. TSPLIB—a traveling salesman problem library. *INFORMS Journal on Computing*, 3(4):376–384, 1991.

[48] Ivan Lima, Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Anand Subramanian, and Thibaut Vidal. CVRPLIB - capacitated vehicle routing problem library. `http://vrp.galgos.inf.puc-rio.br/`, 2014.

[49] Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems*, 31, 2018.

[50] Federico Berto, Chuanbo Hua, Junyoung Park, Laurin Luttmann, Yining Ma, Fanchen Bu, Jiarui Wang, Haoran Ye, Minsu Kim, Sanghyeok Choi, Nayeli Gast Zepeda, André Hottung, Jianan Zhou, Jieyi Bi, Yu Hu, Fei Liu, Hyeonah Kim, Jiwoo Son, Haeyeon Kim, Davide Angioni, Wouter Kool, Zhiguang Cao, Jie Zhang, Kijung Shin, Cathy Wu, Sungsoo Ahn, Guojie Song, Changhyun Kwon, Lin Xie, and Jinkyoo Park. RL4CO: an Extensive Reinforcement Learning for Combinatorial Optimization Benchmark. *arXiv preprint arXiv:2306.17100*, 2024.

A Detailed Model Architecture

A.1 Encoder

A.1.1 Adaptive Node Embedding

The Adaptive Node Embedding module synthesizes distance-related features with node characteristics to create comprehensive node representations. A key aspect of our approach is effectively integrating two complementary spatial features: distance matrix information and coordinate-based relationships. For distance matrix information, we employ a selective sampling strategy that captures the most relevant node relationships while maintaining computational efficiency. Given a distance matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$, we sample k nodes for each node i according to probabilities inversely proportional to their distances:

$$p_{ij} = \frac{1/d_{ij}}{\sum_{j=1}^{N} 1/d_{ij}} \tag{7}$$

where d_{ij} represents the distance between nodes i and j. The sampled distances are then transformed into an embedding space through a learned linear projection:

$$\mathbf{f}_{\text{dist}} = \text{Linear}(\mathbf{d}_{\text{sampled}}) \tag{8}$$

Coordinate information is processed separately to capture geometric relationships between nodes. For each node, we first compute its spatial features based on raw coordinates. These features are then projected into the same embedding space through another learned linear transformation:

$$\mathbf{f}_{\text{coord}} = \text{Linear}(\mathbf{x}_{\text{coord}}) \tag{9}$$

To effectively combine these complementary spatial representations, we employ a Contextual Gating mechanism:

$$\mathbf{h} = \mathbf{g} \odot \mathbf{f}_{\text{coord}} + (1 - \mathbf{g}) \odot \mathbf{f}_{\text{dist}} \tag{10}$$

where $\odot$ is the Hadamard product and $\mathbf{g}$ represents learned gating weights determined by a multi-layer perceptron (MLP):

$$\mathbf{g} = \sigma(\text{MLP}([\mathbf{f}_{\text{coord}}; \mathbf{f}_{\text{dist}}])) \tag{11}$$

This gating mechanism allows the model to adaptively weigh the importance of coordinate-based and distance-based features for each node, enabling more nuanced spatial representation. To handle asymmetric routing scenarios effectively, we follow the approach introduced in and generate dual embeddings for each node: row embeddings $\mathbf{h}^r$ and column embeddings $\mathbf{h}^c$. These embeddings are then combined with other node characteristics (such as demand or time windows) through learned linear transformations to produce the combined node representations:

$$\mathbf{h}^r_{\text{comb}} = \text{MLP}([\mathbf{h}^r; \mathbf{f}_{\text{node}}]) \tag{12}$$

$$\mathbf{h}^c_{\text{comb}} = \text{MLP}([\mathbf{h}^c; \mathbf{f}_{\text{node}}]) \tag{13}$$

where $\mathbf{f}_{\text{node}}$ represents additional node features such as demand or time windows, which are transformed by an additional linear layer. This dual embedding approach allows the RRNCO model to better capture and process asymmetric relationships in real-world routing scenarios.

A.1.2 Neural Adaptive Bias for AAFM

Having established comprehensive node representations through our adaptive embedding approach, RRNCO employs an Adaption Attention-Free Module (AAFM) based on to model complex inter-node relationships. The AAFM operates on the dual representations $\mathbf{h}^r_{\text{comb}}$ and $\mathbf{h}^c_{\text{comb}}$ to capture asymmetric routing patterns through our novel Neural Adaptive Bias (NAB) mechanism. The AAFM operation is defined as:

$$\text{AAFM}(Q, K, V, A) = \sigma(Q) \odot \frac{\exp(A) \cdot (\exp(K) \odot V)}{\exp(A) \cdot \exp(K)} \tag{14}$$

where $Q = \mathbf{W}^Q \mathbf{h}^r_{\text{comb}}$, $K = \mathbf{W}^K \mathbf{h}^c_{\text{comb}}$, $V = \mathbf{W}^V \mathbf{h}^c_{\text{comb}}$, with learnable weight matrices $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V$. While defines the adaptation bias A heuristically as $-\alpha \cdot \log(N) \cdot d_{ij}$ (with learnable α, node count N, and distance d_{ij}), we introduce a Neural Adaptive Bias (NAB) that learns asymmetric relationships directly from data. NAB processes distance matrix $\mathbf{D}$, angle matrix $\mathbf{\Phi}$ for directional relationships, and optionally duration matrix $\mathbf{T}$, enabling joint modeling of spatial-temporal asymmetries inherent in real-world routing.

Let $\mathbf{W}_D, \mathbf{W}_\Phi, \mathbf{W}_T \in \mathbb{R}^E$:

$$\mathbf{D}_{emb} = \text{ReLU}(\mathbf{D}\mathbf{W}_D)\mathbf{W}'_D \tag{15}$$

$$\mathbf{\Phi}_{emb} = \text{ReLU}(\mathbf{\Phi}\mathbf{W}_\Phi)\mathbf{W}'_\Phi \tag{16}$$

$$\mathbf{T}_{emb} = \text{ReLU}(\mathbf{T}\mathbf{W}_T)\mathbf{W}'_T \tag{17}$$

We then apply contextual gating to fuse these heterogeneous information sources. When duration information is available, we employ a multi-channel gating mechanism with softmax normalization:

$$\mathbf{G} = \text{softmax}\left(\frac{[\mathbf{D}_{\text{emb}};\ \mathbf{\Phi}_{\text{emb}};\ \mathbf{T}_{\text{emb}}]\,\mathbf{W}_G}{\exp(\tau)}\right) \tag{18}$$

where $[\mathbf{D}_{emb}; \mathbf{\Phi}_{emb}; \mathbf{T}_{emb}] \in \mathbb{R}^{B\times N\times N\times 3E}$ is the concatenation of all embeddings, $\mathbf{W}_G \in \mathbb{R}^{3E\times 3}$ is a learnable weight matrix, and τ is a learnable temperature parameter. The fused representation is computed as:

$$\mathbf{H} = \mathbf{G}_1 \odot \mathbf{D}_{\text{emb}} + \mathbf{G}_2 \odot \mathbf{\Phi}_{\text{emb}} + \mathbf{G}_3 \odot \mathbf{T}_{\text{emb}} \tag{19}$$

Finally, the adaptive bias matrix $\mathbf{A}$ is obtained by projecting the fused embedding $\mathbf{H}$ to a scalar value:

$$\mathbf{A} = \mathbf{H}\mathbf{w}_{out} \in \mathbb{R}^{B\times N\times N} \tag{20}$$

where $\mathbf{w}_{out} \in \mathbb{R}^E$ is a learnable weight vector. The resulting $\mathbf{A}$ matrix serves as a learned inductive bias that captures complex asymmetric relationships arising from the interplay between distances, directional angles, and travel durations. This Neural Adaptive Bias is then incorporated into the Adaptation Attention Free Module (AAFM) operation as follows:

$$\text{AAFM}(Q, K, V, \mathbf{A}) = \sigma(Q) \odot \frac{\exp(\mathbf{A}) \cdot (\exp(K) \odot V)}{\exp(\mathbf{A}) \cdot \exp(K)}$$

The Neural Adaptive Bias (NAB), applied through the Adaptation Attention Free Module (AAFM), yields final node representations h_F^r and h_F^c after l passes through AAFM. These representations result from RRNCO's encoding process, leveraging joint modeling of distance, angle, and duration to capture complex asymmetric patterns in real-world routing networks.

A.2 Decoder

A.2.1 Decoder Architecture

The decoder architecture combines key elements from the ReLD and MatNet to effectively process the dense node embeddings generated by the encoder and construct solutions for vehicle routing problems. At each decoding step t, the decoder takes as input the row and column node embeddings (h_F^r, h_F^c) produced by the encoder and a context vector $h_c = [h^r_{a_{t-1}}, D^t] \in \mathbb{R}^{d_h + d_{attr}}$, where $D^t \in \mathbb{R}^{d_{attr}}$ represents dynamic features that capture the state variable s^t. To aggregate information from the node embeddings, the decoder applies a multi-head attention (MHA) mechanism, using the context vector h_c as the query and $H^t \in \mathbb{R}^{|F^t|\times d_h}$ as the key and value:

$$h'_c = \text{MHA}(h_c, W^{key}h_F^c, W^{val}h_F^c). \tag{21}$$

The ReLD model introduces a direct influence of context by adding a residual connection between the context vector h_c and the refined query vector h'_c:

$$h'_c = h'_c + \text{IDT}(h_c), \tag{22}$$

where $\text{IDT}(\cdot)$ is an identity mapping function that reshapes the context vector to match the dimension of the query vector, allowing context-aware information to be directly embedded into the representation. To further enhance the decoder performance, an MLP with residual connections is incorporated to introduce non-linearity into the computation of the final query vector q_c:

$$q_c = h'_c + \text{MLP}(h'_c). \tag{23}$$

The MLP consists of two linear transformations with a ReLU activation function, transforming the decoder into a transformer block with a single query that can model complex relationships and adapt the embeddings based on the context. Finally, the probability p_i of selecting node $i \in F^t$ is calculated by applying a compatibility layer with a negative logarithmic distance heuristic score:

$$p_i = \left[\text{Softmax} \left(C \cdot \tanh \left(\frac{(q_c)^T W^\ell h_F^c}{\sqrt{d_h}} - \log(\text{dist}_i) \right) \right) \right]_i \tag{24}$$

where C is a clipping hyperparameter, d_h is the embedding dimension, and dist_i denotes the distance between node i and the last selected node a_{t-1}. This heuristic guides the model to prioritize nearby nodes during the solution construction process. The combination of ReLD's architectural modifications and MatNet's decoding mechanism with our rich, learned encoding enables the RRNCO model to effectively leverage static node embeddings while dynamically adapting to the current context, leading to improved performance on various vehicle routing problems.

The dataset will be available for download through the HuggingFace portal, and the link will be available on a Github repository. Our code is licensed under the MIT license.

B Real-World VRP Dataset Generation

Existing methodologies often require integrating massive raw datasets (e.g., traffic simulators and multi-source spatial data) – for instance, [34] rely on simplistic synthetic benchmarks, which are either resource-intensive or lack real-world complexity [35].

To address these limitations, we design a three-step pipeline to create a diverse and realistic vehicle routing dataset aimed at training and testing NCO models. First, we select cities worldwide based on multi-dimensional urban descriptors (morphology, traffic flow regimes, land-use mix). Second, we develop a framework using the Open Source Routing Machine (OSRM) [36] to create city maps with topological data, generating both precise location coordinates and their corresponding distance and duration matrices between each other. Finally, we efficiently subsample these topologies to generate diverse VRP instances by adding routing-specific features such as demands and time windows, thus preserving the inherent spatial relationships while enabling the rapid generation of instances with varying operational constraints, leveraging the precomputed distance/duration matrices from the base maps. The whole pipeline is illustrated in Fig. 4.

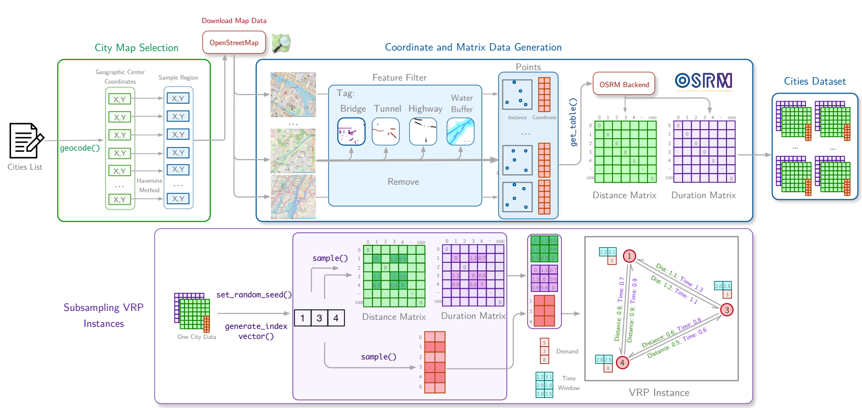

Figure 4: Overview of our RRNCO real-world data generation and sampling framework. We generate a dataset of real-world cities with coordinates and respective distance and duration matrices obtained via OSRM. Then, we efficiently subsample instances as a set of coordinates and their matrices from the city map dataset with additional generated VRP features.

B.1 City Map Selection

We select a list of 100 cities distributed across six continents, with 25 in Asia, 21 in Europe, 15 each in North America and South America, 14 in Africa, and 10 in Oceania. The selection emphasizes urban diversity through multiple dimensions, including population scale (50 large cities >1M inhabitants, 30 medium cities 100K-1M, and 20 small cities <100K), infrastructure development stages, and urban planning approaches. Cities feature various layouts, from grid-based systems like Manhattan to radial patterns like Paris and organic developments like Fez, representing different geographic and climatic contexts from coastal to mountain locations.

We prioritized cities with reliable data availability while balancing between globally recognized metropolitan areas and lesser-known urban centers, providing a comprehensive foundation for evaluating vehicle routing algorithms under diverse real-world conditions. Moreover, by including cities from developing regions, we aim to advance transportation optimization research that could benefit underprivileged areas and contribute to their socioeconomic development.

B.2 Topological Data Generation Framework

In the second stage, we generate base maps that capture real urban complexities. This topological data generation is composed itself of three key components: geographic boundary information, point sampling from road networks, and travel information computation.

Geographic boundary information We establish standardized 9 km^2 areas (3×3 km) centered on each target city's municipal coordinates, ensuring the same spatial coverage across different urban environments. Given that the same physical distance corresponds to different longitudinal spans at different latitudes due to the Earth's spherical geometry, we need a precise distance calculation method: thus, the spatial boundaries are computed using the Haversine spherical distance formulation [37]:

$$d = 2R \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right) \tag{25}$$

where d is the distance between two points along the great circle, R is Earth's radius (approximately 6,371 kilometers), ϕ_1 and ϕ_2 are the latitudes of point 1 and point 2 in radians, $\Delta\phi = \phi_2 - \phi_1$ represents the difference in latitudes, and $\Delta\lambda = \lambda_2 - \lambda_1$ represents the difference in longitudes. This enables precise spatial boundary calculations and standardized cross-city comparisons while maintaining consistent analysis areas across different geographic locations.

Point sampling from road networks Our RRNCO framework interfaces with OpenStreetMap [21] for point sampling. More specifically, we extract both road networks and water features within defined boundaries using `graph_from_bbox` and `features_from_bbox`[1]. Employing boolean indexing, the sampling process implements several filtering mechanisms to filter the DataFrame and ensure point quality: we exclude bridges, tunnels, and highways to focus on accessible street-level locations and create buffer zones around water features to prevent sampling from (close to) inaccessible areas. Points are then generated through a weighted random sampling approach, where road segments are weighted by their length to ensure uniform spatial distribution.

Travel information computation The travel information computation component leverages a locally hosted Open Source Routing Machine (OSRM) server [36] to calculate real travel distances and durations between sampled points, ensuring full reproducibility of results. Through the efficient `get_table` function in our router implementation via the OSRM table service[2], we can process a complete 1000x1000 origin-destination matrix within 18 seconds, making it highly scalable for urban-scale analyses. In contrast to commercial API-based approaches that require more than 20 seconds for 350×350 matrices [38], our open-source local OSRM implementation achieves the same computations in approximately 5 seconds. Additionally, it enables the rapid generation of multiple instances from small datasets with negligible computational cost per iteration epoch.

The RRNCO framework finally processes this routing data through a normalization strategy that addresses both unreachable destinations and abnormal travel times. This step captures real-world routing complexities, including one-way streets, turn restrictions, and varying road conditions, resulting in asymmetric distance and duration matrices that reflect actual urban travel patterns. All

[1] `https://osmnx.readthedocs.io/en/stable/user-reference.html`

[2] `https://project-osrm.org/docs/v5.24.0/api/#table-service`

computations are performed locally[3], allowing for consistent results and independent verification of the analysis pipeline.

B.3 VRP Instance Subsampling

From the large-scale city base maps, we generate diverse VRP instances by subsampling a set of locations along with their corresponding distance and duration matrices, allowing us to generate an effectively unlimited number of instances while preserving the underlying structure.

The subsampling process follows another three-step procedure:

1. *Index Selection*: Given a city dataset containing N_{tot} locations, we define a subset size N_{sub} representing the number of locations to be sampled for the VRP instance. We generate an index vector $\mathbf{s} = (s_1, s_2, \ldots, s_{N_{\text{sub}}})$ where each s_i is drawn from $\{1, \ldots, N_{\text{tot}}\}$, ensuring unique selections.

2. *Matrix Subsampling*: Using $\mathbf{s}$, we extract submatrices from the precomputed distance matrix $D \in \mathbb{R}^{N_{\text{tot}} \times N_{\text{tot}}}$ and duration matrix $T \in \mathbb{R}^{N_{\text{tot}} \times N_{\text{tot}}}$, forming instance-specific matrices $D_{\text{sub}} = D[\mathbf{s}, \mathbf{s}] \in \mathbb{R}^{N_{\text{sub}} \times N_{\text{sub}}}$ and $T_{\text{sub}} = T[\mathbf{s}, \mathbf{s}] \in \mathbb{R}^{N_{\text{sub}} \times N_{\text{sub}}}$, preserving spatial relationships among selected locations.

3. *Feature Generation*: Each VRP can have different features. For example, in Asymmetric Capacitated VRP (ACVRP) we can generate a demand vector $\mathbf{d} \in \mathbb{R}^{N_{\text{sub}} \times 1}$, such that $\mathbf{d} = (d_1, d_2, \ldots, d_{N_{\text{sub}}})^\top$, where each d_i represents the demand at location s_i. Similarly, we can extend to ACVRPTW (time windows) represented as $\mathbf{W} \in \mathbb{R}^{N_{\text{sub}} \times 2}$, where $\mathbf{W} = \{(w_1^{\text{start}}, w_1^{\text{end}}), \ldots, (w_{N_{\text{sub}}}^{\text{start}}, w_{N_{\text{sub}}}^{\text{end}})\}$, defining the valid service interval for each node.

Unlike previous methods that generate static datasets offline [32, 38], our RRNCO generation framework dynamically generates instances on the fly in few milliseconds, reducing disk memory consumption while maintaining high diversity.

Fig. 4 illustrates the overall process, showing how a city map is subsampled using an index vector to create VRP instances with distance and duration matrices enriched with node-specific features such as demands and time windows. Our approach allows us to generate a (arbitrarily) large number of problem instances from a relatively small set of base topology maps totaling around 1.5GB, in contrast to previous works that required hundreds of gigabytes of data to produce just a few thousand instances.

C Related Works

Neural Combinatorial Optimization (NCO). Neural approaches to combinatorial optimization have emerged as a promising paradigm for learning heuristics directly from data, bypassing the need for extensive domain expertise [8]. NCO methods are broadly categorized into construction and improvement methods. Construction methods build solutions sequentially. This line of work was pioneered by Pointer Networks [41] and later combined with reinforcement learning to optimize policies directly for solution quality [42]. The current state-of-the-art for construction heavily relies on autoregressive models using the Transformer architecture [9, 11], which excel at capturing complex problem structures. Other construction paradigms include non-autoregressive methods that predict solutions in a single pass [43] and population-based approaches that generate diverse sets of solutions [44]. In contrast, improvement methods start with an initial solution and iteratively refine it. This includes techniques like learning operators for local search [45] or developing Neural Large Neighborhood Search (NLNS) frameworks [46]. Our work focuses on autoregressive construction, as it provides a strong balance between inference speed and solution quality, which is critical for real-world logistics applications.

Vehicle Routing Problem (VRP) Datasets. A significant gap exists between NCO research and real-world applicability, largely due to the datasets used for training and evaluation. For decades, the community has relied on established benchmarks like TSPLIB [47] and CVRPLIB [48]. While invaluable for standardization, these datasets are typically based on symmetric Euclidean distances, assuming travel costs are equal in both directions ($d_{ij} = d_{ji}$). This simplification fails to capture

[3]Our framework can also be extended to include real-time commercial map API integrations and powerful traffic forecasting to obtain better-informed routing [39, 40], which we leave as future works.

the inherent asymmetry of real road networks caused by one-way streets, traffic patterns, and turn restrictions [17]. Some recent works have attempted to create more realistic datasets [32, 38], but they suffer from critical limitations for NCO research: they often rely on proprietary, commercial APIs, are static and cannot be generated online (a key requirement for data-hungry RL agents), can be slow to generate, and are not always publicly released. Furthermore, they often omit crucial information like travel *durations*, which can be decoupled from distance in real traffic. Our work directly addresses these gaps by providing a fast, open-source, and scalable data generation framework that produces asymmetric distance and duration matrices from real-world city topologies.

NCO for VRPs. The application of NCO to VRPs has evolved from early adaptations of recurrent models [49] to the now-dominant Transformer-based encoder-decoder architectures [9, 11]. These models have demonstrated impressive performance but are fundamentally node-centric; their attention mechanisms operate on node embeddings, making it non-trivial to incorporate rich structural information contained in edge features like a full distance matrix. This limitation is a primary contributor to the sim-to-real gap. To address this, some works have explored encoding edge information. GCN-based approaches [32] and attention via row and column embeddings of MatNet [20] introduced early ways to handle asymmetry, with GOAL [33] incorporating edge data with cross-attention. While these are steps in the right direction, they often process only a single cost matrix (e.g., distance) and do not fully exploit the multiple, correlated modalities of real-world routing costs (distance, duration, and geometry). The development of NCO architectures that can efficiently fuse multiple sources of asymmetric edge information remains an open challenge. Our proposed model, RRNCO, tackles this directly with its Adaptive Node Embedding (ANE) and Neural Adaptive Bias (NAB) mechanism, which learns a unified routing context from distance, duration, and angular relationships.

D Additional Data Information

We present a comprehensive urban mobility dataset encompassing 100 cities across diverse geographical regions worldwide. For each city, we collected 1000 sampling points distributed throughout the same size urban area. The dataset includes the precise geographical coordinates (latitude and longitude) for each sampling point. Additionally, we computed and stored complete distance and travel time matrices between all pairs of points within each city, resulting in 1000×1000 matrices per city.

The cities in our dataset exhibit significant variety in their characteristics, including population size (ranging from small to large), urban layout patterns (such as grid, organic, mixed, and historical layouts), and distinct geographic features (coastal, mountain, river, valley, etc.). The dataset covers multiple regions including Asia, Oceania, Americas, Europe, and Africa. This diversity in urban environments enables comprehensive analysis of mobility patterns across different urban contexts and geographical settings.

Table 4 on the following page provides information about our topology dataset choices.

E Hyperparameter Details

Table 3 shows the hyperparameters we employ for RRNCO. The configuration can be changed through `yaml` files as outlined in RL4CO [50], which we employ as the base framework for our codebase.

Table 4: Comprehensive City Details

City	Population	Layout	Geographic Features	Region	Split
Addis Ababa	Large	Organic	Highland	East Africa	Train
Alexandria	Large	Mixed	Coastal	North Africa	Train
Amsterdam	Large	Canal grid	River	Western Europe	Train
Almaty	Large	Grid	Mountain	Central Asia	Train
Asunción	Medium	Grid	River	South America	Test

Continued on next page

Table 4 – *Continued from previous page*

City	Population	Layout	Geographic Features	Region	Split
Athens	Large	Mixed	Historical	Southern Europe	Train
Auckland	Large	Harbor layout	Isthmus	Oceania	Train
Baku	Large	Mixed	Coastal	Western Asia	Train
Bangkok	Large	River layout	River	Southeast Asia	Train
Barcelona	Large	Grid & historic	Coastal	Southern Europe	Train
Beijing	Large	Ring layout	Plains	East Asia	Train
Bergen	Small	Fjord	Coastal mountain	Northern Europe	Train
Brisbane	Large	River grid	River	Oceania	Train
Buenos Aires	Large	Grid	River	South America	Train
Bukhara	Small	Medieval	Historical	Central Asia	Test
Cape Town	Large	Mixed colonial	Coastal&mt.	Southern Africa	Train
Cartagena	Medium	Colonial	Coastal	South America	Train
Casablanca	Large	Mixed colonial	Coastal	North Africa	Train
Chengdu	Large	Grid	Basin	East Asia	Train
Colombo	Medium	Colonial grid	Coastal	South Asia	Train
Chicago	Large	Grid	Lake	North America	Test
Christchurch	Medium	Grid	Coastal plain	Oceania	Train
Copenhagen	Large	Mixed	Coastal	Northern Europe	Train
Curitiba	Large	Grid	Highland	South America	Train
Cusco	Medium	Historic mixed	Mountain	South America	Test
Daejeon	Large	Grid	Valley	East Asia	Train
Dakar	Medium	Peninsula grid	Coastal	West Africa	Train
Dar es Salaam	Large	Coastal grid	Coastal	East Africa	Train
Denver	Large	Grid	Mountain	North America	Train
Dhaka	Large	Organic	River	South Asia	Train
Dubai	Large	Linear modern	Coastal& desert	Western Asia	Train
Dublin	Large	Georgian grid	Coastal	Northern Europe	Train
Dubrovnik	Small	Medieval walled	Coastal	Southern Europe	Train
Edinburgh	Medium	Historic mixed	Hills	Northern Europe	Train
Fez	Medium	Medieval organic	Historical	North Africa	Test
Guatemala City	Large	Valley grid	Valley	Central America	Train
Hanoi	Large	Mixed	River	Southeast Asia	Train
Havana	Large	Colonial	Coastal	Caribbean	Train
Helsinki	Large	Grid	Peninsula	Northern Europe	Train
Hobart	Small	Mountain harbor	Harbor	Oceania	Test
Hong Kong	Large	Vertical	Harbor	East Asia	Train
Istanbul	Large	Mixed	Strait	Western Asia	Train
Kigali	Medium	Hill organic	Highland	East Africa	Train
Kinshasa	Large	Organic	River	Central Africa	Train
Kuala Lumpur	Large	Modern mixed	Valley	Southeast Asia	Test
Kyoto	Large	Historical grid	Valley	East Asia	Train
La Paz	Large	Valley organic	Mountain	South America	Train
Lagos	Large	Organic	Coastal	West Africa	Train

Continued on next page

Table 4 – *Continued from previous page*

City	Population	Layout	Geographic Features	Region	Split
Lima	Large	Mixed grid	Coastal desert	South America	Train
London	Large	Radial organic	River	Northern Europe	Test
Los Angeles	Large	Grid sprawl	Coastal basin	North America	Train
Luanda	Large	Mixed	Coastal	Southern Africa	Train
Mandalay	Large	Grid	River	Southeast Asia	Train
Marrakech	Medium	Medina	Desert edge	North Africa	Train
Medellín	Large	Valley grid	Mountain	South America	Train
Melbourne	Large	Grid	River	Oceania	Train
Mexico City	Large	Mixed	Valley	North America	Test
Montevideo	Large	Grid	Coastal	South America	Train
Montreal	Large	Mixed	Island	North America	Train
Moscow	Large	Ring layout	River	Eastern Europe	Train
Mumbai	Large	Linear coastal	Coastal	South Asia	Test
Nairobi	Large	Mixed	Highland	East Africa	Train
New Orleans	Medium	Colonial	River delta	North America	Train
New York City	Large	Grid	Coastal	North America	Train
Nouméa	Small	Peninsula	Coastal	Oceania	Test
Osaka	Large	Grid	Harbor	East Asia	Test
Panama City	Large	Coastal modern	Coastal	Central America	Train
Paris	Large	Radial	River	Western Europe	Train
Perth	Large	Coastal sprawl	Coastal	Oceania	Test
Port Moresby	Medium	Harbor sprawl	Coastal hills	Oceania	Train
Porto	Medium	Medieval organic	River mouth	Southern Europe	Train
Prague	Large	Historic grid	River	Central Europe	Train
Quebec City	Medium	Historic walled	River	North America	Test
Quito	Large	Linear valley	Highland	South America	Test
Reykjavik	Small	Modern grid	Coastal	Northern Europe	Test
Rio de Janeiro	Large	Coastal organic	Mountain& coastal	South America	Train
Rome	Large	Historical organic	Seven hills	Southern Europe	Test
Salvador	Large	Mixed historic	Coastal	South America	Train
Salzburg	Small	Medieval core	River	Central Europe	Train
San Francisco	Large	Hill grid	Peninsula	North America	Train
San Juan	Medium	Mixed historic	Coastal	Caribbean	Test
Santiago	Large	Grid	Valley	South America	Train
São Paulo	Large	Sprawl	Highland	South America	Train
Seoul	Large	Mixed	River	East Asia	Train
Shanghai	Large	Modern mixed	River	East Asia	Train
Singapore	Large	Planned	Island	Southeast Asia	Train
Stockholm	Large	Archipelago	Island	Northern Europe	Train
Sydney	Large	Harbor organic	Harbor	Oceania	Train
Taipei	Large	Grid	Basin	East Asia	Train
Thimphu	Small	Valley organic	Mountain	South Asia	Train
Tokyo	Large	Mixed	Harbor	East Asia	Test
Toronto	Large	Grid	Lake	North America	Train
Ulaanbaatar	Large	Grid	Valley	East Asia	Train
Valparaíso	Medium	Hill organic	Coastal hills	South America	Train
Vancouver	Large	Grid	Peninsula	North America	Train
Vienna	Large	Ring layout	River	Central Europe	Train

Continued on next page

Table 4 – *Continued from previous page*

City	**Population**	**Layout**	**Geographic Features**	**Region**	**Split**
Vientiane	Medium	Mixed	River	Southeast Asia	Train
Wellington	Medium	Harbor basin	Coastal hills	Oceania	Train
Windhoek	Small	Grid	Highland	Southern Africa	Test
Yogyakarta	Medium	Traditional	Cultural center	Southeast Asia	Train

Table 3: Hyperparameters.

Hyperparameter	**Value**
Model	
Embedding dimension	128
Number of attention heads	8
Number of encoder layers	12
Normalization	Instance
Use graph context	False
Sample size k node encoding	25
Training	
Batch size	256
Train data size	100,000
Val data size	1,280
Test data size	1,280
RL algorithm	REINFORCE
REINFORCE baseline	POMO [11]
Optimizer	Adam
Learning rate	4e-4
Weight decay	1e-6
LR scheduler	MultiStepLR
LR milestones	[180, 195]
LR gamma	0.1
Max epochs	200
Data	
Number of cities (training)	80
Number of cities (testing)	20
Instance size n (# of locations)	100
Number of test instances	1280