A Data-Driven Multicriteria Bicycle Routing Model for Large-Scale Applications with a Perturbed Extension

Victoria Dahmen*[1] and Klaus Bogenberger[1]

[1]Chair of Traffic Engineering and Control, Technical University of Munich, Germany

Short summary

Bicycle route models often face a trade-off between computational cost and the inclusion of the many variables affecting route choice. We propose a scalable multicriteria routing model designed for large-scale applications like traffic assignment and network design. By extending the weighted shortest path approach, the framework leverages trajectory data alongside optimization and machine learning to estimate edge weights. Improvements are observed across all metrics (up to 13 %) compared to the distance-based shortest path, with the neural network performing best among the learning models. These cost estimation models are network-agnostic; thus, they can adapt to altered edge attributes and are transferable to unseen regions as they rely on open data. Moreover, a perturbed variant is presented to enhance aggregate bicycle count accuracy. By avoiding costly choice-set construction, this framework provides a computationally efficient, data-driven solution for bicycle route generation.

Keywords: bicycle routing; weighted shortest path; machine learning; multicriteria optimization; adaptive routing model; link-cost model.

1 Introduction

It has been well established that bicycle route choice is influenced by a wide range of factors (Lux & Schleinitz, 2021; Hardinghaus & Papantoniou, 2020). In the context of bicycle route choice and routing models, the complexity of the decision criteria is typically dealt with by initially narrowing down the potential routes to a smaller set of candidate routes. In large-scale applications such as traffic assignment, bicycle network design, or real-time routing services, this poses a bottleneck, as generating a route-choice set is computationally costly. In other cross-domain routing applications where runtime is relevant, there is typically one dominant variable of interest, e.g., car travel time or message-passing time in the context of internet protocols. This facilitates the use of the weighted-shortest-path (WSP) approach, which is computationally efficient when implemented via Dijkstra's algorithm or its heuristics, making it suitable for such use cases. A limitation, however, is that this method makes it difficult to account for multiple factors. As this is a crucial aspect in bicycle routing, we propose an adaptation of the WSP approach that leverages dense bicycle GPS-trajectory data in combination with optimization and learning-based models to estimate the underlying edge costs. The resulting model scales well, with route generation runtimes of only 0.5 ms and 2.8 ms for 36 km^2 and 900 km^2 networks, respectively.

Bicycle route models that offer improved computational-efficiency aim to directly generate the best route or a set of Pareto-optimal routes. Recent research uses machine learning and reinforcement learning to predict routes directly (Ding et al., 2024), though these often struggle with regional transferability. Other approaches include genetic and evolutionary algorithms (Hyodo et al., 2000; Nunes et al., 2020), as well as more efficient WSP models (Liu et al., 2019; Steinacker et al., 2022). More promisingly, integrated frameworks using informed A* or stochastic optimization of link-cost functions have achieved runtimes as low as 80–200 ms while accounting for user preferences like comfort and quietness (Hrncir et al., 2017; Schweizer et al., 2020). Despite these gains, a significant gap remains in the literature for routing models that simultaneously account for multiple criteria, achieve computational efficiency, and scale to large and unseen networks.

2 Methodology

The predictive performance of the WSP algorithm can be improved by designing custom weight functions (Liu et al., 2019; Steinacker et al., 2022), yet we propose to take this one step further. We present a two-step approach. First, an optimization algorithm uses observed bicycle trajectory data to find the edge weights that yield the best results when applying the weighted shortest-path algorithm. Second, a data-driven model learns to predict edge weights using link attributes. Lastly, the WSP approach is applied using these predicted edge weights. This is visualized in Figure 1.

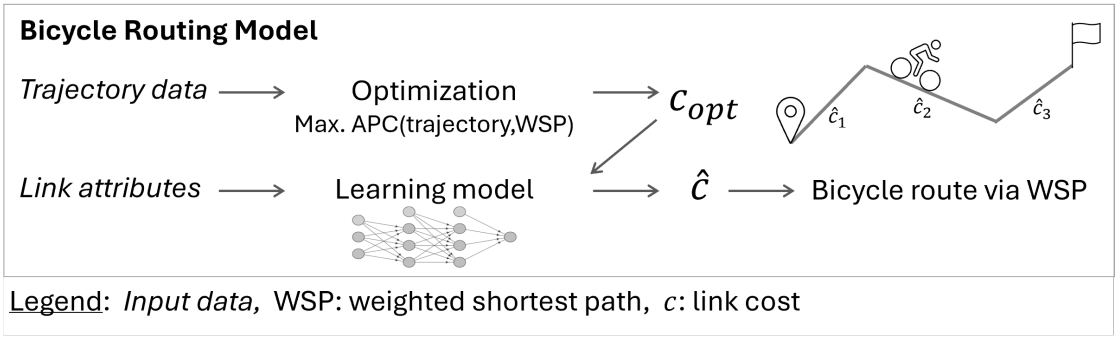


Figure 1: Overview of the bicycle routing framework.

Stage 1: Optimization

In the first stage, the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) is selected for the optimization of the link costs (i.e., edge weights, c_{opt}), as it had the best performance-runtime ratio across different algorithms. It employs a population-based heuristic that refines candidate solutions through successive generations of selection and genetic recombination. Here, the *cma* Python library is used (Hansen & Ostermeier, 2001). To evaluate the effectiveness of the weights at each iteration, the similarity between the WSP S (with edge weights c^i_{opt} at iteration i) and the observed trajectory A is assessed with the Actual Path Coverage (APC):

$$\text{APC} = \frac{\sum_{e \in A \cap S} l_e}{\sum_{e \in A} l_e} \tag{1}$$

where l_e is the length of edge e, and APC $\in [0, 1]$. The APC indicates what fraction of A -weighted by length- is captured within S; this metric is also referred to as 'overlap' in routing literature (Meister et al., 2024). Using Dijkstra's algorithm to generate the WSP, the optimization seeks to minimize the objective function: 1-mean(APC). To accelerate convergence, the edge weights c_{opt} are pre-scaled by their respective lengths l_e. This accounts for the influence of distance on route choice, allowing the CMA-ES to focus on the more nuanced effects of other factors that may affect a link's perceived cost.

Stage 2: Estimating the edge costs

The optimized edge costs c_{opt} inherently reflect cyclists' preferences towards different route characteristics. Thus, we propose a model that generalizes from edge characteristics to estimate edge costs. During the training process, the optimized edge weights c_{opt} serve as ground truth for learning to predict the estimated edge weights $\hat{c}$, as illustrated in Figure 1, while the link attributes are the input data. A total of 25 link attributes were selected as input variables based on the availability of information on OpenStreetMap (OpenStreetMap contributors, 2017) and literature:

- **Bicycle Infrastructure:** Bicycle road, Advisory/Exclusive lane, 1/2-direction cycle path, Pedestrian street, Shared foot & cycle path, Track or lane, Cycle path width, Cycling surface quality.
- **Traffic & Road-related:** Road hierarchy, Speed limit, Number of car lanes, Number of PT lines stopping, One-way street, Accessibility by car/foot, Intersection importance.
- **Environmental:** Share of green areas, Share of retail areas, Gradient, Street lighting, Parking (left/right), Pedestrian crossing.

Models of varying complexity are used to learn to estimate the edge costs: linear and ridge regression, neural network, Graph Attention Network, and Graph Convolutional Network with Residuals, and XGBRegressor. These are summarized in Table 1.

Table 1: Overview of the considered learning-models.

Category	**Model**	**Core Mechanism**	**Configuration**
Regression	LR	Linear least squares.	–
	Ridge	Linear with L2 regularization.	$\alpha = 1.0$
Neural Net	NN	Multi-layer perceptron with activation functions.	2 layers (128, 64); ReLU (last: Inv-Exp); Adam; LR: 0.001; 3k epochs; MSE.
Graph-based	GCNRes	Node feature transformation with neighbor averaging and residual skips.	4 layers; 128 channels; ReLU; LR: 0.001; 400 epochs; MSE.
	GATRes	Self-attention weighted neighbor features with multi-head residual mapping.	2 layers; 4 heads; 128 channels; ReLU; LR: 0.001; MSE.
Tree-based	XGB	Gradient-boosted ensemble of weak learners.	100 trees; Max depth: 3; LR: 0.005; $\alpha : 0.04, \lambda : 0.1$.

When training these link-cost-estimation models, only links are considered that occurred more frequently than the *occurrence threshold* N in the path data used for the previous optimization. This threshold does not apply to inference. Table 2 summarizes the metrics used alongside APC to quantify how well the generated weighted shortest paths with the predicted weights match the actual observed routes.

Table 2: Evaluation metrics for the predicted routes.

Metric	**Equation**	**Description**	**Goal**
JS	$\frac{\sum_{e \in A \cap S} l_e}{\sum_{e \in A \cup S} l_e}$	**Weighted Jaccard Similarity**: Compares common edges to the union of actual and predicted edges, weighted by length l (Jaccard, 1901; Hrncir et al., 2017).	Max.
ADE	$\frac{1}{\lvert N_A \rvert} \sum_{n_A \in N_A} \min_{n_S \in N_S} d_G(n_A, n_S)$	**Average Displacement Error**: The average shortest distance (m) between all nodes in the actual and the predicted path (Capobianco et al., 2021).	Min.
AD	$\frac{\lvert A \setminus S \rvert}{\lvert A \rvert}$	**Arc Difference**: The relative number of changes needed to transform the predicted route to the actual route (Canoy et al., 2023).	Min.
PD	$\lVert \bar{\mathbf{x}}_A - \bar{\mathbf{x}}_S \rVert_2$	**Attribute-based Path Dissimilarity**: Euclidean distance between length-weighted average attribute vectors of the two paths (Hood et al., 2011).	Min.

The Perturbed Variant

Once the link costs have been estimated for any given network, the generated routes are deterministic and suitable for use at the individual trip-level. To address some use cases, such as bicycle traffic assignment, that may benefit from a non-deterministic approach, a slightly modified model version is developed. Optionally, the link cost is perturbed by sampling it from the normal distribution centered at the link's cost with a stochastic deviation of $\alpha * link_cost$; the perturbed cost must be greater than zero. This variability is particularly relevant when considering aggregate trips or flow, to better reflect the heterogeneity in cyclists' route preferences. Figure 2 shows 50 trips

generated for the same OD pair using this perturbed model variant of WSP(NN). The value of α affects the heterogeneity of the generated routes. This behavior is expected, as the stochasticity added to the link costs in part accounts for the variance in route preferences among the cyclists. In this case, for WSP(NN), a value of $\alpha = 0.25$ was used, which yielded the highest R^2 value and accuracy, and lowest scaled-squared-error (SSE; $(q - \hat{q})^2/q)$), when comparing aggregate bicycle counts. When comparing the different link-cost-estimation models, there is an ideal α depending on the distribution of the generated link costs; however, the best R^2 value, accuracy, and SSE are observed mostly within the range of $\alpha = 0.25$ to 0.35.

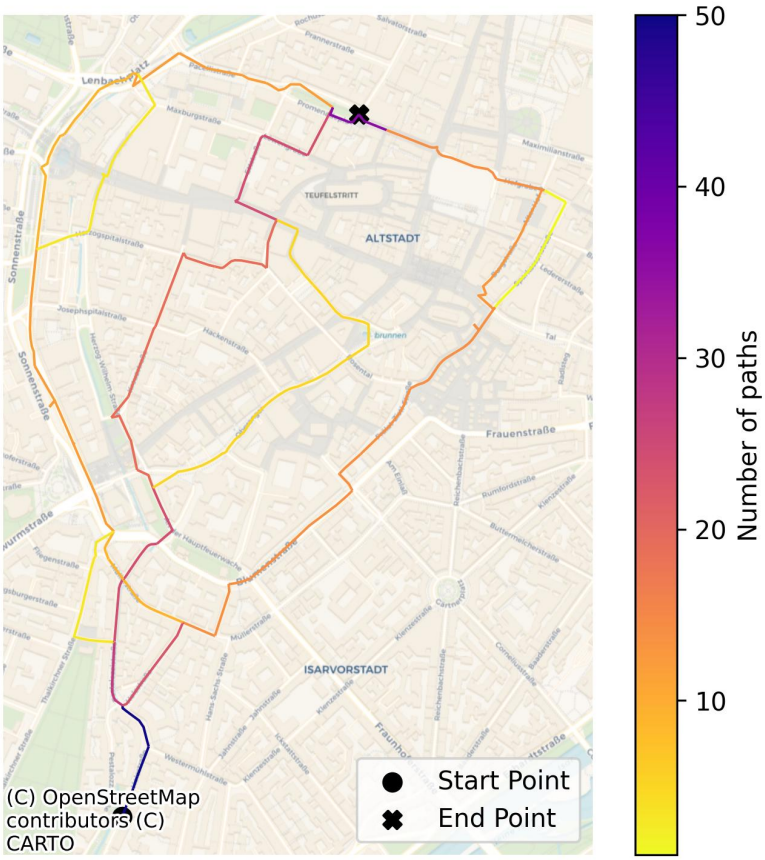


Figure 2: Example of the perturbed routing model using WSP(NN) for 50 routes for one OD pair. Darker color indicates a higher number of routes.

3 Results and discussion

In this section, the proposed methodology is applied to bicycle trajectory data from Munich, Germany. The suitability of the optimized, estimated, and perturbed edge weights is evaluated.

Data preparation

We use bicycle GPS-trajectory data from a 4.0 km by 4.0 km area in downtown Munich, Germany, that was recorded in the *Mobilität.Leben* study (Loder et al., 2023) and thoroughly preprocessed (Dahmen et al., 2024). Some additional filters are applied to achieve high internal and external validity in the bicycle routing context. Only trajectories longer than 1000 m are considered, such that there will generally be a set of routes to pick from. Trip thresholds of 3 and 200 paths are used to avoid over-representation of individual users and to exclude people unfamiliar with the region. High recording quality is ensured by removing tracks with fewer than 25 points per km or with a distance larger than 50 m between a mapmatched point and its GPS point. Furthermore, paths that are more than 1.4 times longer than the shortest path are removed, as these take a significant detour (16 % of tracks). The threshold was chosen based on a sharp drop in the detour ratio histogram just before 1.4, and it aligns closely with the 90th percentile (1.44), capturing atypical

tracks that are likely leisure trips. Lastly, duplicate tracks are removed, i.e., when two paths share more than 95 % of their links. The final data used to develop the bicycle routing model consists of 6731 bicycle trajectories in the case-study area from 356 users.

The road network was obtained from OpenStreetMap, and the multimodal edge attributes were generated using OSMNetFusion Dahmen (2025). Roads not accessible by bicycle, e.g., highways, are removed for this analysis, and adherence to traffic rules is assumed. Subsequently, the bicycle GPS-data is map-matched to the road network using the open-source *LeuvenMapMatching* (Meert & Verbeke, 2018).

Model set-up

Stage 1 - Optimizing: To validate the generalizability of the optimized edge weights to unseen trajectories, the trajectory data is partitioned into training (80 %; 5,385 tracks) and testing (20 %; 1,346 tracks) sets, grouped by user to account for personal preferences. Because the CMA-ES cannot calibrate the edge cost for links without trajectory data, these are omitted from the optimization process. Beyond the objective function, two CMA-ES parameters must be calibrated, which results in a step-size σ of 0.5, and random values between 0 and 1 for the initial weights. We utilized a dense dataset of bicycle trajectories; while this resulted in high optimization runtime (nearly 40 hours), it ensured that the weights reflect general route preferences rather than the idiosyncrasies of a smaller subset.

Stage 2 - Learning: The learning-based models take the link attributes as input variables. These are subject to min-max normalization and a 20:80 test/train split. These models require a suitable occurrence threshold N. Only edges that are used by more than N routes are included in the training data to ensure that only representative edge weights are considered. Here $N = 5$ is used, which accounts for 64 % of the edges; yet, most excluded edges are service roads. A low N preserves attribute diversity but risks including suboptimal weights, while a high N ensures weight quality at the cost of attribute sparsity. The architecture and hyperparameters of the learning models are summarized in Table 1. It is noted that for the NN early-stopping is employed and a weight-decay of 1e-3; the parameters were found by grid-search.

Overall results

For each trip, there is the observed route A and the distance-based shortest path, *Shortest*. Initially, the WSP is generated with the optimized weights c_{opt}, denoted by WSP(optim). Second, the learning-based models are trained to predict the edge weights using the edge attributes, based on the optimized edge weights, and subsequently, the WSP is computed with the predicted weights $\hat{c}$ of each learning model, denoted by WSP(NN), for example. All (weighted) shortest paths are generated using the Dijkstra algorithm via the 'dijkstra_shortest_paths' method in the high-performance *rustworkx* Python library (Treinish et al., 2022). For very large networks, a faster algorithm like A* may be more suitable.

The APC, JS, ADE, AD, and PD are computed for all models. While *Shortest* has a mean APC of 46 %, WSP(optim) is 21.5 % higher (at 56 %). Similar improvements are observed for the JS, AD, ADE, and PD; for AD, ADE, and PD lower values indicate better results. WSP(NN) is 10.6 % better (at 51 %) than *Shortest*; it performs best among the learning models. The relative improvement of each metric across all routes compared to *Shortest* is shown in Figure 3. WSP(Linear) is in second place overall, yet performs similarly to WSP(Ridge). WSP(GATRes) also performs reasonably well, only slightly inferior to WSP(Linear) and WSP(Ridge). The other graph-based approach and WSP(XGB) are inferior.

As the runtime and scalability of this approach are key motivators, we report the execution times for the case-study network and additional networks. The code was implemented in Python 3.10.7 and run on a laptop with the following specifications: 11th Gen Intel(R) Core(TM) i7-11800H CPU @ 2.30 GHz with 8 cores and 32 GB RAM. For the link-cost-estimation models the training and prediction took 26 s and 0.04 s (for all 7,029 edges), respectively. This must only be carried out when there is a change in the network or a new network is used. The generation of the WSP routes has the same runtime for the optimized weights and those predicted by the learning models, as merely the edge weights differ. The route generation takes 0.461 ms per route for the case-study network of 16 km^2 (7,029 edges and 2,836 nodes). As the runtime depends on the network size, two

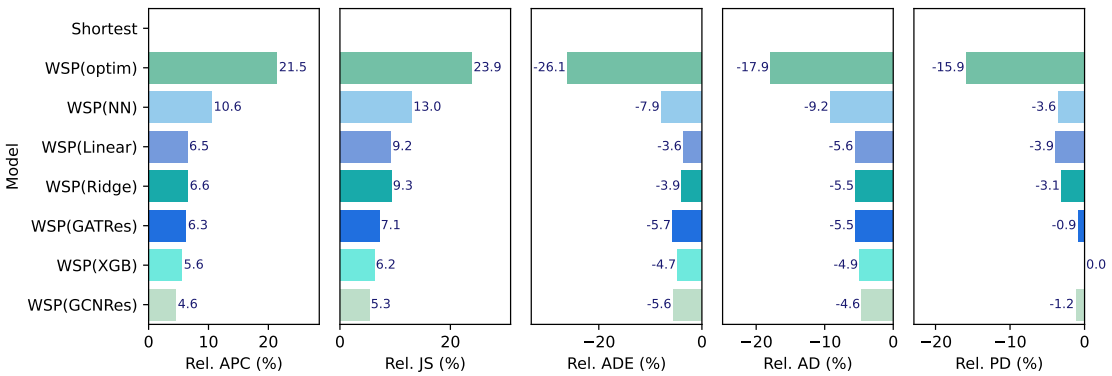
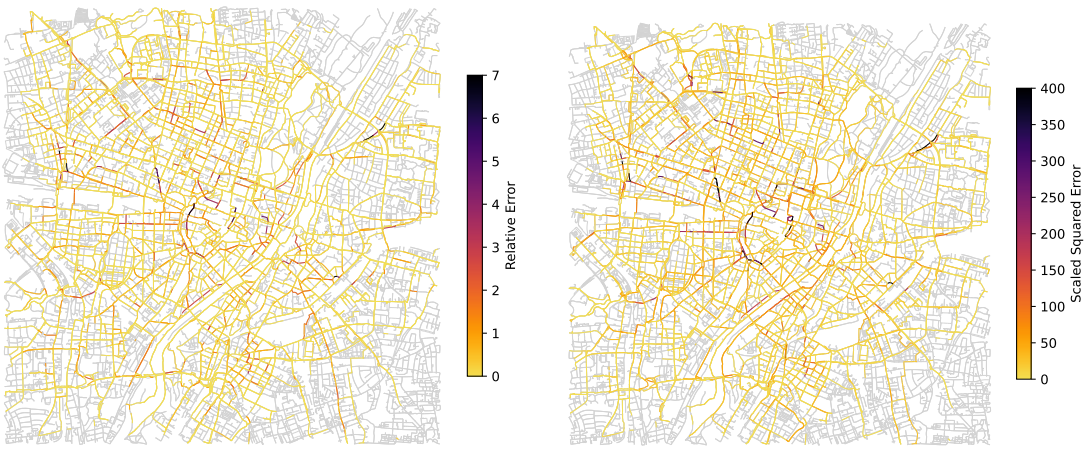


Figure 3: Results of the bicycle routing framework using different learning models.

other networks were evaluated. A 36 km^2 network (16k edges, 6k nodes, suburban Munich) took 0.500 ms, and a large network exceeding 900 km^2 (207k edges, 80k nodes, Montreal Metropolitan Area) required 2.823 ms; thus, the model scales well.

Perturbed variant

While the previous analyses were based on individual paths, the results of the perturbed model variant are assessed in a more aggregate manner, as here the aim is to account for user heterogeneity. The aggregate bicycle counts per link for the generated routes are compared with those for the observed routes. The actual and generated routes exhibit similar patterns in terms of total counts. The discrepancies between the actual and generated aggregate bicycle counts are shown in Figure 4 in terms of relative error and SSE. The SSE across all links is 23.5 for the WSP(NN) routing model compared to 36.6 for *Shortest*, which is a 36 % improvement. Furthermore, the R^2 values of the link counts from the routes generated with WSP(NN) (0.748 for all links; 0.629 for links with count>10) are much higher than those of *Shortest* (0.596; 0.407); by 26 % and 55 %, respectively.



(a) Relative error

(b) Scaled squared error

Figure 4: Evaluation of the estimated bicycle counts compared to the observed bicycle counts per link (except links with negligible observed bicycle counts of < 10 bicycles).

4 Conclusions

While reduced network complexity can substantially improve the runtime of bicycle routing models, such models commonly involve computationally costly choice-set generation. It is important to account for a wide range of factors in bicycle routing, yet it is difficult to maintain the multicriteria aspect when trying to drastically reduce the runtime of bicycle routing models. This is

complicated further when regional transferability -without retraining- is of interest. To address this gap, we propose a bicycle routing model that is computationally efficient (0.5 ms/route), scales well with network size, only relies on open data (once trained) and is thus network-agnostic. The approach is a two-step process based on the WSP (Dijkstra). Initially, the optimized weights of the links are found through optimization using bicycle trajectory data. Subsequently, various approaches are used to learn to generalize the relationship between a link's perceived weight and its attributes, enabling the prediction of edge weights of unseen or altered links. Across various metrics, we see an improvement (of up to 13 %) compared to the distance-based shortest path. This indicates that the models have indeed picked up on cyclists' preferences, despite their heterogeneity.

A limitation of this approach is that adherence to traffic rules is assumed, thus traffic violations in the trajectory data add noise to the data; however, this is partially filtered out by removing poorly mapmatched trips. While the model is designed to be inherently transferable across network versions and regions; this should be evaluated further, as this motivates the use of predicted over optimized edge weights. Moreover, the model behavior could be analyzed in depth with regard to trip purpose, trip length, and type of bicycle.

Acknowledgements

Victoria Dahmen acknowledges funding by the TUM Georg Nemetschek Institute Artificial Intelligence for the Built World.

References

Canoy, R., Bucarey, V., Mandi, J., & Guns, T. (2023, September). Learn and route: learning implicit preferences for vehicle routing. *Constraints*, *28*(3), 363–396. doi: 10.1007/s10601-023-09363-2

Capobianco, S., Millefiori, L. M., Forti, N., Braca, P., & Willett, P. (2021). Deep learning methods for vessel trajectory prediction based on recurrent neural networks. *IEEE Transactions on Aerospace and Electronic Systems*, *57*(6), 4329-4346. doi: 10.1109/TAES.2021.3096873

Dahmen, V. (2025). *OSMNetFusion: A Topologically-Simplified Multimodal Information-Rich OpenStreetMap Network - v1.0.0.* Zenodo. Retrieved from `https://doi.org/10.5281/zenodo.17485919` doi: 10.5281/zenodo.17485919

Dahmen, V., Álvarez Ossorio Martínez, S., Loder, A., & Bogenberger, K. (2024). Making large-scale semi-passive gps travel diaries valuable: a quality enhancement method. In *103rd annual meeting of the transportation research board (trb 2024)*. doi: 10.13140/RG.2.2.28580.04487/1

Ding, M., Guo, Y., Huang, Z., Lin, B., & Luo, H. (2024, September). GROM: A generalized routing optimization method with graph neural network and deep reinforcement learning. *Journal of Network and Computer Applications*, *229*, 103927. doi: 10.1016/j.jnca.2024.103927

Hansen, N., & Ostermeier, A. (2001, 06). Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, *9*(2), 159-195. doi: 10.1162/106365601750190398

Hardinghaus, M., & Papantoniou, P. (2020). Evaluating cyclists' route preferences with respect to infrastructure. *Sustainability*, *12*(8). doi: 10.3390/su12083375

Hood, J., Sall, E., & and, B. C. (2011). A GPS-based bicycle route choice model for San Francisco, California. *Transportation Letters*, *3*(1), 63–75. doi: 10.3328/TL.2011.03.01.63-75

Hrncir, J., Zilecky, P., Song, Q., & Jakob, M. (2017, March). Practical Multicriteria Urban Bicycle Routing. *IEEE Transactions on Intelligent Transportation Systems*, *18*(3), 493–504. doi: 10.1109/TITS.2016.2577047

Hyodo, T., Suzuki, N., & Takahashi, K. (2000, January). Modeling of Bicycle Route and Destination Choice Behavior for Bicycle Road Network Plan. *Transportation Research Record: Journal of the Transportation Research Board*, *1705*(1), 70–76. (Num Pages: 7) doi: 10.3141/1705-11

Jaccard, P. (1901, 01). Distribution de la Flore Alpine dans le Bassin des Dranses et dans quelques régions voisines. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, *37*, 241-72. doi: 10.5169/seals-266440

Liu, S., Shen, Z.-J., & Ji, X. (2019, January). Urban Bike Lane Planning with Bike Trajectories. *Manufacturing & Service Operations Management*. (Num Pages: 39)

Loder, A., Cantner, F., Dahmen, V., & Bogenberger, K. (2023). The Mobilität.Leben Study: a Year-Long Mobility-Tracking Panel. *arXiv*. doi: 10.48550/ARXIV.2308.04973

Lux, & Schleinitz. (2021). *Entwicklung einer Methodik zur Untersuchung der Determinanten der Routenwahl von Radfahrern* (Vol. Heft 321; S. Lux & K. Schleinitz, Eds.). Bremen: Fachverlag NW in der Carl Schünemann Verlag GmbH. doi: 0673

Meert, W., & Verbeke, M. (2018). Hmm with non-emitting states for map matching. In *Proceedings of the European Conference on Data Analysis (ECDA).* (Paderborn, Germany)

Meister, A., Liang, Z., Felder, M., & Axhausen, K. W. (2024). Comparative study of route choice models for cyclists. *Journal of Cycling and Micromobility Research*, *2*, 100018. doi: https://doi.org/10.1016/j.jcmr.2024.100018

Nunes, P., Moura, A., & Santos, J. (2020). Evolutionary Approach for the Multi-objective Bike Routing Problem. In E. Lalla-Ruiz, M. Mes, & S. Voß (Eds.), *Computational Logistics* (Vol. 12433, pp. 311–325). Cham: Springer International Publishing. (Series Title: Lecture Notes in Computer Science) doi: 10.1007/978-3-030-59747-4_20

OpenStreetMap contributors. (2017). *Planet dump retrieved from https://planet.osm.org.* `https://www.openstreetmap.org`.

Schweizer, J., Rupi, F., & Poliziani, C. (2020). Estimation of link-cost function for cyclists based on stochastic optimisation and gps traces. *IET Intelligent Transport Systems*, *14*(13), 1810-1814. doi: https://doi.org/10.1049/iet-its.2019.0683

Steinacker, C., Storch, D.-M., Timme, M., & Schröder, M. (2022, January). Demand-driven design of bicycle infrastructure networks for improved urban bikeability. *Nature Computational Science*, *2*, 655–664. (Num Pages: 10) doi: 10.1038/s43588-022-00318-w

Treinish, M., Carvalho, I., Tsilimigkounakis, G., & Sá, N. (2022). rustworkx: A high-performance graph library for python. *Journal of Open Source Software*, *7*(79), 3968. doi: 10.21105/joss.03968