A Neighborhood-Based Method for Counterfactual Path Explanation

Linxin Yang[1], **Zhongshan Gao**[1], **Dong Zhang**[2], **Juan Chen**[2], **Haohan Huang**[2], **Jingyi Zhao**[1], **Xiaodong Luo**[1],

[1]Shenzhen Research Institute of Big Data
[2]Huawei

Abstract

The Counterfactual Routing Competition requires us to generate a modified map such that a specified counterfactual, or foil, route becomes the optimal path. In our submission, we adopted a set of simple yet effective techniques to construct such maps that align with the target foil route. Our framework begins with a basic one-opt strategy, where the edges not belonging to the foil are iteratively evaluated and selectively removed if doing so improves the route similarity score. To enhance performance, we incorporate additional modular components, including an N-opt heuristic to further reduce graph error and a min-cut-based algorithm designed to reconstruct the foil route while minimizing structural changes to the graph. As a result, our approach efficiently achieves near-theoretical optimal graph error within the allowed route error threshold on the provided test cases.

1 Related Works

The permissible actions in this competition are modifications to edge properties, namely, adjusting height or width to activate/deactivate an edge, or changing its type to alter its weighted length. The central challenge is thus to identify which edges to disable to render the foil route optimal with minimal change. Recent approaches such as [Kikuta *et al.*, 2024; Wang and Wang, 2022] focus on edge-level explainability, scoring edge importance in a routing context based on objectives like travel time efficiency. While these methods effectively highlight critical edges, they do not directly aim to disable edges to change a route's optimality.

Incorporating heuristics into classic path-finding algorithms can further enhance performance. For example, [Gschwind *et al.*, 2018] proposes a bidirectional labeling technique in Dijkstra-like routines for routing problems. By constructing separate cost matrices that satisfy both the Delivery Triangle Inequality (DTI) forward and the Pickup Triangle Inequality (PTI) backward, this method enables strong dominance pruning in both directions. These label sets naturally identify edges not present in any non-dominating path/edge, making them prime targets for pruning.

Moreover, well-known combinatorial optimization tricks can be integrated as modular enhancements. The 2opt edge swap operator, a staple in tour improvement, offers effective local enhancements at very low computational cost [Nagata and Bräysy, 2009; Vidal *et al.*, 2012]. Similarly, Adaptive Large Neighborhood Search, which dynamically removes and reinserts edges or requests, is also widely adopted ([Zhao *et al.*, 2025; Vidal *et al.*, 2013]) and delivers SoTA for various routing scenarios.

2 Our Approaches

2.1 Problem Settings

Given a map $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ consisting of vertices $\mathcal{V}$ and edges $\mathcal{E}$, the goal is to construct a modified map $\mathcal{G}^c = \{\mathcal{V}^c, \mathcal{E}^c\}$ that minimizes the structural change $d_g(\mathcal{G}, \mathcal{G}^c)$, while ensuring that the shortest path on $\mathcal{G}^c$ matches closely a given foil route $\mathcal{P}$. This is equivalent to the following optimization problem:

$$\begin{aligned} arg\min_{\mathcal{G}^c} \quad & d_g(\mathcal{G}, \mathcal{G}^c) \\ s.t. \quad & S(\mathcal{P}, \mathcal{P}^c) \leq \tau \end{aligned} \tag{1}$$

where $S(\mathcal{P}, \mathcal{P}^c)$ denotes the similarity error between the generated path and the foil route, $\mathcal{P}^c = \phi(\mathcal{G}^c)$ is the shortest path for the given map found by router $\phi(\cdot)$, and τ is the allowed threshold.

2.2 Label-based Presolve

A simple yet effective strategy for solving model (1) is to eliminate all routes that violate the similarity constraint. To implement this, we first run a customized Dijkstra algorithm in both forward and backward directions, as described in Algorithm 1, to generate the corresponding label sets $\{\mathcal{L}_+, \mathcal{L}_-\}$, where each node v is annotated with its shortest distance from the origin $(\mathcal{L}_+(v))$ and to the destination $(\mathcal{L}_-(v))$, respectively.

Then, for each node $v \in \mathcal{V}$, we compute the estimated shortest path length through v as $\tilde{d} = \mathcal{L}_+(v) + \mathcal{L}_-(v)$, representing the sum of the shortest distance from the origin to v and from v to the destination. If $\tilde{d}$ exceeds the length of the foil route by more than a predefined threshold, node v can be safely discarded, as it cannot lie on any feasible route that satisfies the similarity constraint and is significantly shorter than the foil route. Finally, we can obtain the pruned graph $\tilde{\mathcal{G}}$.

Algorithm 1 Customized Dijkstra with labeling

Input 1: Graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$
Input 2: Source v_s and Sink v_d
Output 1: Shortest path $\mathcal{P}^*$
Output 2: Labeling set $\mathcal{L}$

```
 1: Let Priority Queue $Q \leftarrow \{(0, [v_s], v_s)\}$.
 2: Let Dictionary $\mathcal{L} \leftarrow \varnothing$
 3: while $Q \neq \varnothing$ do
 4:     Let $(d, \mathcal{P}, v) \leftarrow pop(Q)$
 5:     if $v = v_d$ then
 6:         $\mathcal{P}^* = \mathcal{P}$
 7:         BREAK
 8:     end if
 9:     for $v_n \in N(v)$ do
10:         if $v_n \notin \mathcal{L}$ OR $d + D(v, v_n) < \mathcal{L}(v_n)$ then
11:             $\mathcal{L}(v_n) \leftarrow d + D(v, v_n)$
12:             $Q \leftarrow Q \bigcup \{d + D(v, v_n), \mathcal{P} + [v_n], v_n\}$
13:         end if
14:     end for
15: end while
16: return $\mathcal{P}^*, \mathcal{L}$
```

2.3 Iterative One-opt Framework

At this stage, the task becomes aligning the map's shortest path with the given foil path. This process can be decomposed into two key steps: **i)** enabling the edges that constitute the foil route, and **ii)** disabling edges that may be involved in shorter alternative paths.

For the first step, enabling the foil route is straightforward: we adjust the width and height attributes of each edge according to the user model to ensure they are traversable. In cases where certain edges on the foil route cannot be activated due to numerical issues, we invoke the Dijkstra algorithm to efficiently compute a substitute subpath that connects the endpoints of the problematic edge.

For the second step, while several strategies exist, we adopt a lightweight, iterative approach that empirically maximizes efficiency. This is primarily because recomputing the shortest path after each map update is computationally inexpensive. In particular, we employ the One-opt heuristic, which evaluates the effect of disabling a single edge at a time. Since each edge evaluation is independent, this process is parallelized for further acceleration. Nevertheless, the search space may remain substantial. To further prune it, we limit the candidate edges to those that lie on the current shortest path but are not present in the foil route. The complete scheme of the iterative one-opt framework is presented in Algorithm 2

2.4 Improving Heuristics

At this stage, the modified graph $\mathcal{G}^c$ should yield a shortest path that is similar to the foil path, but may include redundant modifications. To minimize such unnecessary changes, we apply the N-opt heuristic, which attempts to restore multiple previously modified edges simultaneously and evaluates whether the resulting graph still satisfies the similarity constraint. While this procedure is inherently parallelizable, it was not implemented in our final submission due to time constraints.

Algorithm 2 Iterative One-opt Framework

Input 1: Presolved graph $\tilde{\mathcal{G}}$
Input 2: Foil route $\hat{\mathcal{P}}$
Parameter: Similarity threshold ϵ
Output: Resulting Graph $\mathcal{G}^c$

```
 1: for $i \in [1..|\hat{\mathcal{P}}| - 1]$ do
 2:     if $(v_i, v_{i+1})$ is not traversable then
 3:         Enable edge $(v_i, v_{i+1})$
 4:         if Failed then
 5:             Replace $(v_i, v_{i+1})$ with $\phi(v_i, v_{i+1})$
 6:         end if
 7:     end if
 8: end for
 9: Initialize similarity error $E = \infty$
10: Update shortest path $\mathcal{P} \leftarrow \phi(\mathcal{G}^c)$
11: while $E > \epsilon$ do
12:     Update candidate edges $C \leftarrow \mathcal{P} \setminus \hat{\mathcal{P}}$
13:     Obtain step $c^* \leftarrow arg\min_e S(\hat{\mathcal{P}}, \phi(\mathcal{G}^c \setminus e))$
14:     Update shortest path $\mathcal{P} \leftarrow \phi(\mathcal{G}^c \setminus e)$
15:     Update similarity error $E \leftarrow S(\hat{\mathcal{P}}, \mathcal{P})$
16:     Update graph $\mathcal{G}^c \leftarrow \mathcal{G}^c \setminus e$
17: end while
18: return $\mathcal{G}^c$
```

In addition, we introduce another refinement targeting edges that were enabled to support the foil path. Since the resulting path is only required to remain within a similarity threshold, we perform a One-opt procedure on these edges. For each enabled edge, we attempt to restore it to its original state and recompute the similarity error. If the updated graph remains within the allowable error margin, the restoration is accepted. This step helps reduce unnecessary modifications while preserving feasibility.

Additionally, we implemented a minimum-cut-based algorithm designed to enhance the efficiency of the iterative One-opt framework. In this approach, candidate edges are partitioned into groups based on their connectivity. For each group, we compute a minimum cut over the corresponding subgraph segment and disable the identified cut edges. By repeatedly applying this process, the shortest path can be progressively steered toward the foil route. While this method is theoretically more structured and allows for coordinated pruning, we observed that its runtime performance was slightly slower than the naive iterative One-opt approach in practice. As a result, it was retained as a backup strategy rather than used in the final submission.

3 Results

We report the performance of our algorithm on a subset of the provided test cases, as shown in Table 1. All experiments were conducted on a system equipped with an AMD Ryzen 9700X CPU and 48GB of RAM. To avoid potential scheduling overhead, the maximum number of parallel threads was limited to 6. As shown in Table 1, our approach achieves consistently low graph error while keeping the similarity er-

instance	Graph Err.	Sim. Err.	Time(s)
osdpm0_1	1	0	15.7
osdpm0_2	1	0.0139	22.1
osdpm0_3	2	0.0391	45.0
osdpm0_4	2	0.0076	18.8
osdpm0_5	2	0	68.5

Table 1: Computational result of our approach on several test cases

ror well within the acceptable threshold. Furthermore, the solution times are significantly below the allowed time limit, demonstrating the practical efficiency and scalability of our method.

References

[Gschwind *et al.*, 2018] Timo Gschwind, Stefan Irnich, Ann-Kathrin Rothenbächer, and Christian Tilk. Bidirectional labeling in column-generation algorithms for pickup-and-delivery problems. *European Journal of Operational Research*, 266(2):521–530, 2018.

[Kikuta *et al.*, 2024] Daisuke Kikuta, Hiroki Ikeuchi, Kengo Tajiri, and Yuusuke Nakano. Routeexplainer: an explanation framework for vehicle routing problem. In *Pacific-asia conference on knowledge discovery and data mining*, pages 30–42. Springer, 2024.

[Nagata and Bräysy, 2009] Yuichi Nagata and Olli Bräysy. Edge assembly-based memetic algorithm for the capacitated vehicle routing problem. *Networks: An International Journal*, 54(4):205–215, 2009.

[Vidal *et al.*, 2012] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624, 2012.

[Vidal *et al.*, 2013] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & operations research*, 40(1):475–489, 2013.

[Wang and Wang, 2022] Zheng Wang and Shen Wang. Xrouting: Explainable vehicle rerouting for urban road congestion avoidance using deep reinforcement learning. In *2022 IEEE International Smart Cities Conference (ISC2)*, pages 1–7. IEEE, 2022.

[Zhao *et al.*, 2025] Jingyi Zhao, Claudia Archetti, Tuan Anh Pham, and Thibaut Vidal. Large neighborhood and hybrid genetic search for inventory routing problems. *arXiv preprint arXiv:2506.03172*, 2025.