Counterfactual Explanations for Personalized Route Planning

ZeJian Chen[1] , **Jintao You**[2] , **Xiaodong Luo**[3]
[1]College of Urban Transportation and Logistics, Shenzhen Technology University, Shenzhen, 518118, China
[2]School of Advanced Engineering, Great Bay University, Dongguan 523000, PR China
[3]Shenzhen International Center for Industrial and Applied Mathematics, Shenzhen Research Institute of Big Data, Shenzhen 518172, PR China
a529976447@gmail.com, youjintao@gbu.edu.cn, xiaodongluo@cuhk.edu.cn

Abstract

Counterfactual explanations offer insights into *why* a particular route is considered optimal by revealing how slight changes to the environment could make an alternative route optimal instead. We propose a two-stage approach that combines greedy heuristics with an ALNS-based refinement to generate minimal counterfactual map edits that would make an alternative route optimal.

1 Introduction

Modern route-planning algorithms underpin everything from autonomous-vehicle navigation to pedestrian way-finding apps. Yet these systems are often perceived as opaque "black boxes," especially when their recommended paths diverge from a user's intuition. Such opacity can undermine confidence and slow adoption [Chakraborti *et al.*, 2017]. Explainable AI planning (XAIP) seeks to restore trust by translating low-level computations into concepts people can understand. A particularly promising XAIP technique is *counterfactual explanation*, which clarifies a decision by pinpointing the minimal map changes that would cause a different route to become optimal.

2 Related Work

Counterfactual explanations of route optimality. Brandão et al. [2021] were the first to formulate the question of why one path is optimal rather than another as an inverse shortest path optimization problem. By formulating both *full* and *incremental* inverse-optimization problems, they leverage mixed-integer solvers to compute minimal sets of terrain-type changes that would make a user's foil route optimal. Their methods achieve speedups of several orders of magnitude over generic XAIP search and, in user studies, produce explanations that align closely with human intuitions. In a complementary vein, Chakraborti et al. [2017] introduced *model reconciliation* for AI planning, identifying the smallest model updates needed to reconcile a planner's output with a user's expectations. While this approach is highly general, its search-based nature can incur significant overhead on large navigation graphs. Inspired by these contributions, our work avoids heavy optimization and explicit model-space search. Instead, we employ a lightweight, two-stage procedure combining greedy edge removals with an Adaptive Large Neighborhood Search (ALNS) to quickly isolate minimal map edits. This heuristic/meta-heuristic hybrid maintains computational efficiency while directly targeting the aspects of the foil route that are suboptimal.

3 Problem Definition

The following problem definition is adapted from the official CRC Competition rules and guidelines [CRC, 2025].

3.1 Graph Representation and Attributes

We represent a simplified map of a neighborhood in Amsterdam as a directed graph $G = (V, E)$, where V denotes locations (nodes) and E denotes roads or paths (edges). Each edge $e_{i,j} \in E$ has multiple attributes $f_{at}(e_{i,j})$ for each $at \in \mathcal{A}$, where $\mathcal{A}$ is the set of all possible attributes (e.g., length, curb height, free width, path type).

3.2 Task Overview

In the Counterfactual Routing Competition (CRC), participants receive:

- A map G (provided as a GeoDataFrame with attributes).
- A router implementing Dijkstra's algorithm under a user model U.
- A *fact route* $P_f = \pi_U^r(G)$, the optimal path under U.
- A *foil route* P^t, an alternative path that the user expected.

The user's question is:

> *"Why is the fact route optimal for me, and not the foil route?"*

We seek a *counterfactual map* G' that is as similar as possible to G, in which the foil route would be optimal. The changes from G to G' serve as an explanation for why P_f was chosen instead of P^t.

3.3 Formal Definition

Let $d_g(G, G')$ denote the *graph distance metric*, i.e., the number of graph modifications (edge attribute edits) needed to

transform G into G'. Let $d_r(r, r')$ denote the *route distance metric*:

$$d_r(r, r') \;=\; 1 - 2\,\frac{\sum_{e \in r \cap r'} d_e}{\sum_{e \in r} d_e + \sum_{e \in r'} d_e}\,,$$

where d_e is the length of edge e. A foil route is *optimal* on G' if $d_r\big(P^t, \pi_U^r(G')\big) \le \delta$ for a small threshold δ. The counterfactual map is then defined by:

$$G' \;=\; \arg\min_{\hat{G}}\; d_g(G, \hat{G}) \quad \text{s.t.} \quad d_r\big(P^t, \pi_U^r(\hat{G})\big) \;\le\; \delta\,.$$

3.4 User Model and Router

Each user model U is specified by:

- `min_sidewalk_width`: minimum acceptable width.
- `max_curb_height`: maximum acceptable curb height.
- `crossing_weight`: penalty factor for crossings.
- `walk_bike_preference`: {`walk`, `bike`}.
- `weight_bike_preference_factor`: strength of the walk/bike preference.

Edges that violate width or curb constraints are excluded from the graph before routing. The weights of the remaining edges combine length, crossing penalties, and mode preferences. Running Dijkstra's algorithm on this weighted graph yields $\pi_U^r(G)$.

3.5 Graph Operators

Participants may modify only the following edge attributes:

- **Width (numerical)**: adjust `obstacle_free_width_float` to a value within $[0.6, 2.0]$ m.
- **Curb height (numerical)**: adjust `curb_height_max` to a value within $[0, 0.2]$ m.
- **Path type (categorical)**: switch `path_type` between `walk` and `bike`.

Any attempted operation that falls outside these bounds is considered invalid. No edges may be added or removed; only attribute edits are allowed.

4 Method

We treat the map G as a black box. Given any perturbed version of G, we rerun Dijkstra's algorithm under user model U to obtain a new fact route. Comparing this new route to the user's fixed foil route yields a *route error*, which drives our stopping criterion and informs our perturbation heuristics.

4.1 Greedy Algorithm A: Feasibility-First Pruning

1. **Foil Inclusion.** Ensure every edge on P_{foil} is marked as included (`include`=1) in G.
2. **Disagreement Set.** Let $\Delta = \{e \in P_f \setminus P_{\text{foil}}\}$ be the set of edges present in the current fact route but not in the foil route.

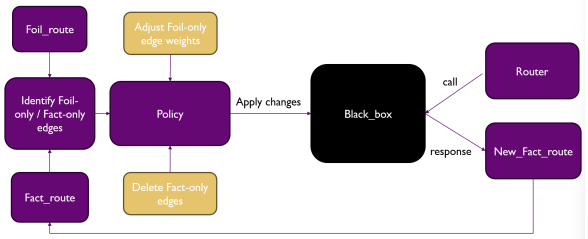

Figure 1: Overview of our black-box perturbation framework. Given the original map and foil route, the approach: (1) identifies fact-only and foil-only edges; (2) applies a policy to perturb edge attributes; (3) queries the black-box router for a new fact route; and (4) iterates until the route error meets the threshold.

3. **Evaluate Hypotheses.** For each $e \in \Delta$, simulate a perturbation (e.g., reduce its width or increase its curb height) that would flip its `include` flag from 1 to 0 (i.e., effectively remove e from the traversable graph). Then rerun the router on the perturbed graph to compute:
   - the new fact route P_f';
   - the resulting route error $d_r(P_f', P_{\text{foil}})$.
4. **Select Best Edge.** Pick the edge whose simulated removal yields the largest improvement, according to a heuristic score that combines route error reduction and perturbation cost.
5. **Apply Perturbation.** Apply that perturbation to G. Update $P_f \leftarrow P_f'$ and repeat from Step 2.
6. **Terminate.** Stop once $d_r(P_f, P_{\text{foil}}) \le \delta$ or Δ becomes empty.

4.2 Greedy Algorithm B: Direct Edge Pruning

To avoid any bias from enforcing foil feasibility first, we run a second greedy pass without the *Foil Inclusion* step:

- Start from the original map G without forcibly including the foil route's edges.
- Iteratively remove disagreement edges as in Greedy A, selecting and applying the best perturbation in each round.
- This may temporarily disconnect P_{foil}. Discard any candidate for which the route error cannot be evaluated.
- Terminate when $d_r(P_f, P_{\text{foil}}) \le \delta$, using the same error threshold as above.

4.3 ALNS Refinement

We initialize the ALNS phase with the best map G^0 obtained from Greedy A. Each iteration of ALNS consists of the following phases:

1. **Destroy Phase (Solution Destruction).** Randomly select a fraction $\rho \in \{25\%, 50\%, 75\%, 100\%\}$ of the current edge-attribute edits and revert them to their original values in G. This "destroys" part of the current counterfactual solution, producing a map G_{d} in which the selected edits have been removed.

2. **Repair Phase (Two Operator Types).**
   On G_{d}, repeatedly apply one of the following repair operators:
   - *Width/Height Repair:* choose one edge in $P_f \setminus P_{\mathrm{foil}}$ and adjust its `obstacle_free_width` or `curb_height` so that its `include` flag flips from 1 to 0 (as in Greedy A).
   - *Type Repair:* choose one such edge and switch its `path_type` (walk $\leftrightarrow$ bike) to alter its weight.

   Continue applying these operators until the route error on the map falls below the incumbent best.
3. **Acceptance.**
   If the repaired map G' yields a strictly lower route error, accept G' as the new incumbent solution and reward the successful operators. Otherwise, revert to the previous incumbent solution and penalize the ineffective operators.
4. **Adaptation & Iteration.**
   Update the selection probabilities for the destroy and repair operators based on their recent performance. Repeat until a time limit is reached or the residual route error reaches zero.

By alternating large-scale destroy moves (width/height flips) with focused repair moves (path-type changes), and continually adapting operator probabilities, ALNS efficiently escapes local minima and converges on a minimal set of edge-attribute edits that collectively render the foil route optimal.

5 Experiments

We benchmark our two-stage framework on five publicly available CRC [CRC, 2025] demo maps, each covering a real Amsterdam road network with detailed accessibility annotations. For every instance we record:

- **Greedy Perturbations** – edge–attribute edits made by the initial greedy pruning;
- **Route Error (Init/Final)** – residual route error (RE), i.e. $1-$similarity, between the fact and foil routes before and after refinement (lower is better);
- **ALNS Perturbations** – additional edits inserted by the ALNS stage;
- **Time (s)** and **ALNS Iterations** – runtime and iteration count of the ALNS search.

Table 1 summarises the quantitative results. Across all five maps, our method drives the RE down from as high as 0.62 to below 0.05 (often exactly 0), using at most six greedy edits plus five ALNS edits, all well within the five-minute competition limit.

Instance	Greedy Perturb.	Init RE	Final RE	ALNS Perturb.	Time (s)	Iters. ALNS
osdpm_4_1	6	0.6241	0.02496	5	270.06	626
osdpm_4_2	5	0.4587	0.04309	4	270.30	579
osdpm_4_3	3	0.2636	0.04973	3	134.11	640
osdpm_4_4	1	0.2393	0.00000	1	2.50	0
osdpm_4_5	1	0.6082	0.00000	1	4.58	0

Table 1: Results on five CRC demo maps. "Greedy/ALNS Perturb.": edge–attribute edits introduced by the greedy and ALNS stages; "Init/Final RE": residual route error (0 means the foil is already optimal); "Time" and "ALNS Iters.": runtime and iteration count for the ALNS stage.

6 Conclusion and Future Work

We have presented a lightweight framework for generating counterfactual explanations in personalized route planning. By sequentially applying greedy pruning and a metaheuristic refinement, our approach produces intuitive modifications that make a user's alternate route (foil route) optimal with minimal changes to the map. Future work will include conducting user studies to assess the interpretability of the explanations, and extending the approach to dynamic scenarios such as live traffic updates.

References

[Brandão *et al.*, 2021] M. Brandão, A. Coles, and D. Magazzeni. Explaining path plan optimality: Fast explanation methods for navigation meshes using full and incremental inverse optimization. In *Proceedings of the 31st International Conference on Automated Planning and Scheduling (ICAPS)*, pages 23–31, 2021.

[Chakraborti *et al.*, 2017] T. Chakraborti, S. Sreedharan, Y. Zhang, and S. Kambhampati. Plan explanations as model reconciliation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 156–163, 2017.

[CRC, 2025] Counterfactual routing competition (crc), ijcai–25. https://sites.google.com/view/crc25-ijcai/home, 2025. Accessed: 2025-07-24.