Submission to the Counterfactual Routing Competition @ IJCAI-25

André Artelt
Bielefeld University, Germany
aartelt@techfak.uni-bielefeld.de

Abstract

This document describes my submission to the Counterfactual Routing Competition (CRC), held at IJCAI-2025. My proposed algorithm constitutes a greedy approach that finds an exact solution if it exists.

1 Introduction

The Counterfactual Routing Competition (CRC) addresses the need for explainable route planning in personalized routing for wheelchair users. While classic path planners, such as Dijkstra, can generate optimal personalized routes, they lack transparency about why certain routing decisions are made. This lack of transparency can hinder trust, especially among users with limited mobility, who often have unique personal routing requirements and constraints. In addition to distance, aspects such as curb height, sidewalk width, and the number of crossings can significantly influence a route's suitability, and any trade-offs should be clearly communicated. The competition aims to clarify why the planner favored a particular factual route over an alternative one, considering the user's constraints and preferences, by offering counterfactual explanations.

The competition is part of a larger project involving organizers from the municipality of Amsterdam (Netherlands). Most importantly, it relies on real-world data (maps) generated by the municipality of Amsterdam, utilizing various methods such as point-cloud-based obstacle and width measurements, as well as identifying suitable crossings from satellite imagery.

The remainder of this report is structured as follows: First, we briefly describe the competition problem from a technical point of view (Section 2). Based on this, we introduce and discuss our proposed algorithm in Section 3. Finally, this report closes with a summary and conclusion (Section 4).

2 The Problem

The competition models a map as a directed graph with multi-attribute edges, including attributes such as length, path type, curb height and width. In addition, user preferences such as the maximum curb height, a crossing weight factor, a walk-bike preference weight factor, and the minimum sidewalk width are given as well and must be satisfied by the routing method. The routing is implemented by the Dijkstra algorithm [Dijkstra, 2022], which is used for computing the shortest path between a start and destination in this graph. For running the Dijkstra algorithm, all edges that violate the user preferences of the maximum curb height and minimum sidewalk width are removed from the graph. Next, the edge weights (single scalars) are computed by considering the length of the edge and multiplying it by the user-specific crossing weight factor and the user-specific walk-bike preference weight factor. Finally, all edge weights are normalized.

The task to be solved in this competition is as follows:

> Given a graph (model of a map), a start and end node, user preferences, a fact route – i.e., the shortest path between the start and end node under the given user preferences –, and an alternative/foil route. The task is to find a minimal set of modifications to the graph such that the foil route becomes the shortest path (or as similar as possible to the shortest path) as computed by the Dijkstra algorithm.

3 My Algorithm

Given that neither the length/weight of an edge nor the given user preferences can be changed, it becomes apparent that only the deletion of edges can influence the shortest path. In order to delete an edge, either the curb height or the sidewalk width must be changed such that it violates the user preferences.

With this intuition in mind, we propose the following greedy strategy for modifying the graph:

1. Make sure all edges from the foil path are included in the graph by adjusting the curb height, (obstacle-free) sidewalk width, and path type if necessary.
2. Iteratively removing all edges that lead to a different shortest path by adjusting the attributes CURB_HEIGHT_MAX or OBSTACLE_FREE_WIDTH_FLOAT such that they violate the user preferences:
   (a) Start with the first edge that differs between the fact and foil.

(b) Remove edges until those two edges are the same.
(c) Move on to the next edge that differs and repeat

This greedy algorithm will find a feasible solution (i.e., turning the fact into the foil) if it exists. However, it will fail for cases in which it is impossible to turn the fact into the foil. For those cases, we include a fallback to the evolutionary baseline[1] provided by the competition organizers.

We tested this algorithm on the provided scenarios and observed that in approx. 90% of the cases, our algorithm finds an exact solution – i.e., the modifications to the graph turn the fact into the foil route. Only 3 scenarios do not have a feasible solution, and our algorithm falls back to the evolutionary baseline method.

The implementation of the proposed algorithm is available on GitHub: https://github.com/AndreArtelt/Submission-to-CRC-IJCAI25

4 Conclusion & Summary

This report described my submission to the Counterfactual Routing Competition (CRC). My proposed algorithm does not use any AI technology but constitutes a greedy approach that is able to find an exact solution if it exists. The proposed algorithm shows a strong performance on the published available scenarios.

Despite the strong performance, a formal proof on the correctness of the proposed algorithm remains to be done. Furthermore, a formal study of the problem, in particular feasibility and approximability, could provide valuable insights for the future design of routing methods. We leave those aspects as future work.

Acknowledgments

This research was supported by the Ministry of Culture and Science NRW (Germany) as part of the Lamarr Fellow Network. This publication reflects the views of the authors only.

References

[Dijkstra, 2022] Edsger W Dijkstra. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: his life, work, and legacy*, pages 287–290. 2022.

[1]We manually tuned the hyperparameter to ensure that it does not exceed the runtime limit but still outputs good solutions.