An MIP-guided search algorithm for the counterfactual routing problem

Xiangdong Lv[1] , **Junhao Shi**[2] , **Yixiao Huang**[2]
[1]Geek+ Robotics
[2]SF Technology
lvxiangdong2qq@163.com, 372353289@qq.com, HuangYixiao@outlook.com

Abstract

This paper reports a solution method to solve the problem proposed in the counterfactual routing competition 2025 (CRC25) of IJCAI. We propose a hybrid matheuristics to quickly generate counterfactual explanations to the *foil routes* with a combination of mathematical programming and a search algorithm. Numerical experiments demonstrate the advantages of the proposed algorithm.

1 Introduction

With the wide application of machine learning methods, sometimes it is critical to be able to generate human-understandable explanation to gain users' trust on the machine learning models, especially for high-stake decisions.

Following this trends, it is also important to provide explanations to the operations research/combinatorial optimization models. This paper considers generate counterfactual explanations in the context of personalized routing problem, proposed in the counterfactual routing competition 2025 (CRC25) of IJCAI.[1] The problem considers an optimization problem that to change the attributes of arcs on the graph such that the shortest path of the modified graph matches a pre-specified path.

We first formulate this problem as a mixed-integer program (MIP) and propose a tailored best-first search algorithm. We further propose an MIP-guided best-first search algorithm that exploits the information of the MIP solution.

The remainder of the paper is organized as follows. We first review the related work in Section 2. Then, we describe the counterfactual problem and provide the corresponding mathematical model in Section 3. We propose the search and MIP-guided search algorithms in Section 4. We show the numerical results in Section 5 and conclude this paper in Section 6.

2 Related Work

The counterfactual routing problem is related to the literature on counterfactual explanation, which is an important topic in the field of explainable artificial intelligence (XAI).

[1]The website of CRC 25 is https://sites.google.com/view/crc25-ijcai/home.

While the idea of counterfactual explanation stems from and still focuses on the field of machine learning [Verma *et al.*, 2024], some studies also focus on explaining the optimization problems.

[Korikov *et al.*, 2021] show that the inverse combinatorial optimization is suitable for generating the associated counterfactual explanations. They further investigate methods to solve the problem under two common objective functions: the weighed sum of binary variables and the weighted sum of general integer variables.

[Forel *et al.*, 2023] adapt the idea of counterfactual explanations to the data-driven optimization problems. They develop tailored algorithm to find alternative context (scenarios) where another solution outperforms the optimal solution.

[Kurtz *et al.*, 2025] consider the counterfactual explanations for linear optimization, where they formally define three type of types of counterfactual explanations: strong, weak, and relative. Among them, they show that weak counterfactual explanations can be equivalently converted to a linear optimization problem by dualizing the primal problem.

We also want to mention that the counterfactual routing problem is also related to the interdiction on shortest path problems, which is to deactivate a limited number of arcs/nodes on the graph to maximize the distance of the shortest path [Smith, 2010]. It can be formulated as a mixed-integer program by dualizing the shortest path problem [Khachiyan *et al.*, 2008].

3 Problem Description and Mathematical Model

We first describe the problem in details and propose a mixed-integer program.

Consider a graph $G = (V, E)$ where V is the set of nodes and E is the set of edges.Let d_{ij} denote the length of edge (i, j). Each edge is associated with a few extra attributes, where the width, curb height, and path type are modifiable. Changing an attribute of an edge is regarded as calling a graph operator once.

For a user wants traverse from node s to node t, the user is associated with several parameters, e.g., minimum acceptable sidewalk width, maximum acceptable curb height, preference on footpaths or bike paths. Among them, the minimum acceptable sidewalk width and the maximum acceptable curb height decide whether an edge is feasible for this user. The preference on footpaths or bike paths is included in the objective, where the arc with preferred path type is associated with a discount α in the edge length.

Given a prespecified *foil route*, the counterfactual routing problem is to find a counterfactual map where the associated optimal route is similar enough (within a threshold δ) to the *foil route* with the number of operators called minimized. The similarities of two routes are calculated as follows.

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

Note that we will temporarily ignore the threshold of route errors in the remainder of this section and propose a mathematical model to solve the exact counterfactual routing problem. However, we will consider the threshold in algorithm design discussed in Section 4.

Inspired by the shortest path interdiction formulation [Smith, 2010] and the weak counterfactual explanations for linear optimization [Kurtz *et al.*, 2025], we formulate the counterfactual routing problem as a mixed integer program embedded with a dual shortest path formulation.

While the original decisions are to modify the width and height of each edge, we simplify it by only defining two state: feasible state and infeasible state. We define the decision variables as follows

$$y_{ij} = \begin{cases} 1, & \text{if edge } (i,j) \text{ is feasible,} \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{if type of edge } (i,j) \text{ is changed,} \\ 0, & \text{otherwise,} \end{cases}$$

$$\pi_i \in \mathbb{R}^+ : \text{shortest distance to node } i \text{ from node } s.$$

We introduce an additional parameter $\Delta^+_{ij}/\Delta^-_{ij}$ to represent the number of operators to make arc (i,j) feasible/infeasible, respectively. The calculation of Δ^+_{ij} and Δ^-_{ij} is intuitive. For example, if arc (i,j) is infeasible in both height and width, then we have $\Delta^+_{ij} = 2$ and $\Delta^-_{ij} = 0$.

The mixed-integer program of the *exact* counterfactual routing problem is as follows.

$$\min \sum_{(i,j)\in A} [\Delta^+_{ij} y_{ij} + \Delta^-_{ij}(1 - y_{ij}) + x_{ij}] \quad (1)$$

$$\text{s.t.} \quad \pi_j - \pi_i \le d_{ij} + \delta^d_{ij} x_{ij} + M(1 - y_{ij}), \quad \forall (i,j) \in A, \quad (2)$$

$$\pi_j - \pi_i \ge d_{ij} + \delta^d_{ij} x_{ij} - M(1 - y_{ij}), \quad \forall (i,j) \in \mathcal{R}_{\text{foil}}, \quad (3)$$

$$y_{ij} = 1, \quad \forall (i,j) \in \mathcal{R}_{\text{foil}}. \quad (4)$$

Expression (1) minimizes the total number of operators. Constraints (2) guarantee that when arc (i,j) is feasible, the label of node j is greater than or equal to node i plus the weight of arc (i,j). Constraints (3) ensure that for each arc (i,j) of the foil route, the label of node j is exactly equal to node i plus the weight of arc (i,j). Constraints (4) require that all arcs on the foil route must be feasible. Note that we can simplify constraints (3) to

$$\pi_j - \pi_i \ge c_{ij} + d_{ij} x_{ij}, \ \forall (i,j) \in \mathcal{R}_{\text{foil}},$$

as the feasibility of arcs on the foil route is guaranteed.

4 Methodology

Given start s and end t vertices, the *fact route* $\mathcal{R}_{\text{fact}}$ is the shortest path from s to t. The *foil route* $\mathcal{R}_{\text{foil}}$ is an alternative path. Our goal is to find a minimal set of edge modifications $\mathcal{M}$ such that

1. $\mathcal{R}_{\text{foil}}$ becomes feasible (satisfies width/height constraints)
2. $\mathcal{R}_{\text{foil}}$ becomes the shortest path within error threshold

We formulate this as a best-first search problem:

- **Node**: A subproblem defined by:
  - Modified graph $G' = (V, E')$
  - Current solution $\mathcal{R}_{\text{new}}$
  - Current modification cost
- **Subproblem**: For first fork-point d and merge-point r, define:

$$\mathcal{R}^{\text{sub}}_{\text{fact}} = \mathcal{R}_{\text{fact}}[d \to r]$$
$$\mathcal{R}^{\text{sub}}_{\text{foil}} = \mathcal{R}_{\text{foil}}[d \to r]$$

Modify attributes on $\mathcal{R}^{\text{sub}}_{\text{fact}}$ to redirect shortest path.

Algorithm 1 Best-First Search for counterfactual routing problem

```
Input: Graph $G$, fact route $\mathcal{R}_{\text{fact}}$, foil route $\mathcal{R}_{\text{foil}}$
Output: Minimal modification set $\mathcal{M}$ making $\mathcal{R}_{\text{foil}}$ shortest
    path
 1: queue ← PriorityQueue()
 2: best_solution ← $\emptyset$
 3: Add initial node (no modifications) to queue
 4: while queue not empty $\wedge$ ¬timeout do
 5:     $n$ ← queue.pop()
 6:     $G_n, \mathcal{M}_n$ ← ApplyModifications($G, n$)
 7:     $\mathcal{R}_{\text{new}}$ ← ShortestPath($G_n, s, t$)
 8:     if RoutesEqual($\mathcal{R}_{\text{new}}, \mathcal{R}_{\text{foil}}$) then
 9:         Update best_solution if $|\mathcal{M}_n| < |$best_solution$|$
10:         continue
11:     else if ShouldPrune($n$, best_solution) then
12:         continue
13:     end if
14:     $(d, r)$ ← FindForkAndMerge($\mathcal{R}_{\text{new}}, \mathcal{R}_{\text{foil}}$)
15:     $\mathcal{R}_{\text{sub}}$ ← $\mathcal{R}_{\text{new}}[d \to r]$
16:     $\mathcal{C}$ ← ScoreEdges($\mathcal{R}_{\text{sub}}, G$)
17:     $k$ ← BranchingFactor($n$.depth)
18:     for $e \in$ TopK($\mathcal{C}, k$) do
19:         $n'$ ← CreateChildNode($n, e$)
20:         queue.push($n'$, (RouteError($n'$), GraphError($n'$)))
21:     end for
22: end while
23: return best_solution
```

ScoreEdges For each edge $e = (i, j)$ in $\mathcal{R}_{\text{sub}}$:

$$\text{DetourRatio}(e) = \frac{\text{Len}(\text{SP}(G \setminus \{e\}, i, j))}{w(e)}$$

$$\text{Betweenness}(e) = \sum_{u \in V \setminus \{t\}} \frac{\sigma_{ut}(e)}{\sigma_{ut}}$$

$$\text{DegreeScore}(e) = \frac{\deg^-(i) + \deg^+(j)}{2}$$

$$\text{TerminalScore}(e) = \begin{cases} 1 & \text{if } i = d \text{ or } j = r \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Score}(e) = \sum \omega_i \cdot \text{NormalizedFeat}_i$$

BranchingFactor Determined by search depth:

$$k(n) = \begin{cases} 6 & \text{if depth}(n) < 3 \\ 2 & \text{if } 3 \le \text{depth}(n) \le 6 \\ 1 & \text{if depth}(n) > 6 \end{cases}$$

Pruning Prune node n if:

- $\text{RouteError}(\mathcal{R}_{\text{new}}, \mathcal{R}_{\text{foil}}) > \text{best\_solution.error}$
- $|\mathcal{M}_n| \ge |\text{best\_solution.modifications}|$

4.1 Modifications to ScoreEdges

The key modification lies in the edge scoring function ScoreWithMipGuidance. We augment the original scoring function by introducing a new binary feature, *MIPGuidance*, which indicates whether an edge is recommended by the MIP solution:

$$\text{MIPGuidance}(e) = \begin{cases} 1 & \text{if } e \in \mathcal{M}_{\text{MIP}} \\ 0 & \text{otherwise} \end{cases}$$

The new composite score for edge e becomes:

$$\text{Score}(e) = \sum \omega_i \cdot \text{NormalizedFeat}_i + \omega_{\text{MIP}} \cdot \text{MIPGuidance}(e)$$

where ω_{MIP} is a fixed weight.

4.2 MIP-Guided Algorithm

The remainder of the search framework, including node structure, branching strategy, and pruning criteria, remains unchanged from Algorithm 1. The only difference is the use of the modified ScoreEdges function that integrates MIP-based recommendations to prioritize promising edges.

Algorithm 2 MIP-Guided Best-First Search

```
Input: Graph $G$, fact route $\mathcal{R}_{\text{fact}}$, foil route $\mathcal{R}_{\text{foil}}$, MIP guidance set $\mathcal{M}_{\text{MIP}}$
Output: Minimal modification set $\mathcal{M}$ making $\mathcal{R}_{\text{foil}}$ shortest path
 1: queue ← PriorityQueue()
 2: best_solution ← $\emptyset$
 3: Add initial node (no modifications) to queue
 4: while queue not empty ∧ ¬timeout do
 5:     $n$ ← queue.pop()
 6:     $G_n, \mathcal{M}_n$ ← ApplyModifications($G, n$)
 7:     $\mathcal{R}_{\text{new}}$ ← ShortestPath($G_n, s, t$)
 8:     if RoutesEqual($\mathcal{R}_{\text{new}}, \mathcal{R}_{\text{foil}}$) then
 9:         Update best_solution if $|\mathcal{M}_n|$ < |best_solution|
10:         continue
11:     else if ShouldPrune($n$, best_solution) then
12:         continue
13:     end if
14:     $(d, r)$ ← FindForkAndMerge($\mathcal{R}_{\text{new}}, \mathcal{R}_{\text{foil}}$)
15:     $\mathcal{R}_{\text{sub}}$ ← $\mathcal{R}_{\text{new}}[d \to r]$
16:     $\mathcal{C}$ ← ScoreWithMipGuidance($\mathcal{R}_{\text{sub}}, G, \mathcal{M}_{\text{MIP}}$)
17:     $k$ ← BranchingFactor($n$.depth)
18:     for $e \in$ TopK($\mathcal{C}, k$) do
19:         $n'$ ← CreateChildNode($n, e$)
20:         queue.push($n'$, (RouteError($n'$), GraphError($n'$)))
21:     end for
22: end while
23: return best_solution
```

5 Numerical Experiments

To demonstrate the efficiency of the propose algorithm, we conduct numerical experiments based on the 25 test instances provided by the competition organizers.

We have implemented the algorithms in Python 3.8. We solve the mixed-integer programming by calling the Scipy package. All the experiments are performed on a laptop with a processor Apple M4 at 4.41 GHz and 16 GB memory.

5.1 Analysis of the search algorithm

We first conduct a component-based analysis on the search algorithm to derive the added values of some important algorithmic components. For each component investigated, we delete the component, rerun the algorithm, and compare the results with those of the full search algorithm shown in Algorithm 1.

Algorithm	# ins. solved	Graph error	Total time (s)
Full search	**25**	2.80	91.34
Search w/o detour	23	3.40	111.92
Search w/o BC	24	2.56	87.30
Search w/o degrees	**25**	2.68	90.23

Table 1: Result of search algorithms

We show the results in Table 1, where "search w/o detour" refers to the search algorithm excluding the detour ratio, "search w/o BC" refers to the algorithm excluding the betweenness centrality, and "search w/o degrees" refers to the algorithm excluding the degree score. Among them, search w/o detour and search w/o BC failed to find feasible results for all the instances, which demonstrate the values of considering detour ratio and betweenness centrality.

While we find including the degree score is valuable in preliminary tests, the results in Table 1 show that the degree score may lead to suboptimal solutions.

5.2 Analysis of the MIP-guided search algorithm

We further investigated the value of including the MIP solution as a warm start, by comparing the performances of Algorithm 2 and Algorithm 1, shown in Table 2.

	Full search	MIP-guided search
MIP time (s)	-	8.71
Search time (s)	91.34	87.12
Total time (s)	91.34	95.83
# nodes explored	41.52	39.08
# ins. solved on root node	1	9

Table 2: Result of MIP-guided search

Since both algorithms result in the same solutions, we focus on the comparison of efficiency. We show that including the MIP solution leads to the reduction of the number of nodes explored and the search time. Furthermore, eight more instances are solved after only exploring the root node by including the MIP solution. However, as it takes 8.71 seconds on average to solve the MIP instances, the total solution time of the MIP-guided search is longer. Nonetheless, we think the MIP-guided search is valuable as we can further reduce the time of solving the MIP.

6 Conclusion

In this paper, we tackle the counterfactual routing problem to generate explanation in the context of personalized route planning. We formally formulate the problem as a mixed-integer program and propose a best-first search algorithm. Further, we propose an MIP-guided search algorithm by including the MIP solution, which is able to reduce the number of nodes explored and the search time.

We believe such methodology has the potential to generalize to generate counterfactual explanation for other combinatorial optimization problems.

References

[Forel *et al.*, 2023] Alexandre Forel, Axel Parmentier, and Thibaut Vidal. Explainable data-driven optimization: from context to decision and back again. In *International Conference on Machine Learning*, pages 10170–10187. PMLR, 2023.

[Khachiyan *et al.*, 2008] Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled Elbassioni, Vladimir Gurvich, Gabor Rudolf, and Jihui Zhao. On short paths interdiction problems: Total and node-wise limited interdiction. *Theory of Computing Systems*, 43(2):204–233, 2008.

[Korikov *et al.*, 2021] Anton Korikov, Alexander Shleyfman, and Chris Beck. Counterfactual explanations for optimization-based decisions in the context of the gdpr. In *ICAPS 2021 workshop on explainable AI planning*, 2021.

[Kurtz *et al.*, 2025] Jannis Kurtz, Ş İlker Birbil, and Dick den Hertog. Counterfactual explanations for linear optimization. *European Journal of Operational Research*, 2025.

[Smith, 2010] J Cole Smith. Basic interdiction models. *Wiley encyclopedia of operations research and management science*, 2010.

[Verma *et al.*, 2024] Sahil Verma, Varich Boonsanong, Minh Hoang, Keegan Hines, John Dickerson, and Chirag Shah. Counterfactual explanations and algorithmic recourses for machine learning: A review. *ACM Computing Surveys*, 56(12):1–42, 2024.