A Destroy-and-Repair Heuristic for the Counterfactual Routing Problem

Dmitry Konovalov , Alexander Yuskov , Igor Kulachenko[*] , Andrey Melnikov , Igor Vasilyev , Haohan Huang , Juan Chen , Dong Zhang

HGLM Team

[*]soge.ink@gmail.com

Abstract

Within the Counterfactual Routing Competition of the IJCAI-25 conference, a user traveling from one node to another on the graph is considered. The user has a desired (*foil*) route and several parameters, affecting how they perceive the lengths of the edges and whether the edges are available. The foil route can deviate from the *fact* shortest route computed taking the user's parameters into account, and the Counterfactual Routing Problem (CRP) is to introduce the smallest number of modifications into the graph edges' attributes so that the fact and foil routes become close enough. In this work, we present two heuristic algorithms for the CRP and propose a MIP model to evaluate the algorithms' performance.

1 Introduction

The present study is motivated by the need to explain to a wheelchair user why they should choose a specific shortest route, called a *fact route*, instead of the one, hereafter referred to as the *foil route*, that appears to be more favorable at first glance. The idea is to modify the transportation graph so that the foil route becomes optimal. Then one could list the modifications in the actual graph that must be there to make the foil route optimal, but since they are absent, the user should prefer the fact one.

The problem of finding the smallest number of modifications so that the shortest route in the modified graph is close to the foil one is further referred to as Counterfactual Routing Problem or CRP for short. CRP can be considered as a bilevel problem, where two players act in a hierarchical organization. Firstly, one player, called Leader, decides how the graph must be modified, aiming to ensure that the shortest route computed by the second player, called Follower, after the Leader's decision, satisfies Leader's conditions. Such kind of models are known in the literature as interdiction shortest path problems [Israeli and Wood, 2002], and are relatively well studied now due to their broad security and other applications.

Since the condition that the fact route in the modified graph is close enough to the foil one can be unmet by a heuristic algorithm, the CRP in the competition is formulated as a bi-objective problem, where the infeasibility in terms of this condition is minimized first. When the condition is satisfied, the number of modifications must be minimized as well. So, we deal with a bi-objective lexicographical minimization problem.

To find a quality solution of the CRP in a reasonable time, two heuristic approaches are proposed. The idea of sequential exploration of promising modifications with backtracking is embodied in a Tree Search (TS) framework, inspired by Monte Carlo Tree Search in a deterministic form [Browne *et al.*, 2012]. We also develop a multi-start variant of the Destroy-and-Repair (DR) algorithm – a heuristic popularized in vehicle routing as Large Neighborhood Search (LNS) [Pisinger and Røpke, 2010] – with destroy operator (removing graph modifications) and repair operator (introducing new modifications to restore feasibility). A mixed-integer programming (MIP) model obtained by a single-level reformulation of a bilevel one is proposed to compute lower bounds for benchmarking the heuristics. The code for our TS, DR, and MIP implementations is available in the repository [HGLM, 2025].

The remainder of the paper is organized as follows. In Section 2, we formalize the problem and derive its MIP formulation. In Section 3, the TS and DR heuristics are described in detail, and their computational study is given in Section 3.4. Section 4 concludes the paper.

2 Problem Description and MIP Formulation

We are given a graph $G = (V, E)$, where V denotes the set of nodes (locations) and E denotes the set of edges (road segments connecting these locations). The length of an edge $e \in E$ is denoted by l_e.

Each edge has several attributes such as length, curb height, sidewalk width, etc. The fact routes are built by the route planner based on user's parameters aggregated in the user model. These parameters are the following:

- Minimum acceptable sidewalk width;
- Maximum acceptable curb height;
- Preferred mode of traveling: on walk paths or bike paths;
- A parameter that indicates the strength of the preference in mode of traveling.

Given the user model, the route planner computes the fact route that operates with the edges that have the appropriate width and height of the sidewalk. Instead of pure edge length values, it uses their modified values depending on the user's preferred mode of travel and necessity to cross roads.

Our decisions regarding the modifications of edge attributes can be represented by a triplet of binary values. Despite the fact that the edge attributes sidewalk width and curb height take numeric values, the route planner either takes the edge into account if its attributes fit the user parameters or discards the edge otherwise. In this case, we can either keep the present state of the edge or modify the numeric attribute forcing the planner to switch the edge's status. The decision to switch the edge's status would completely enable or disable the edge respectively. The choice of mode of traveling is binary also, since we either leave the mode as it is in the original graph or change it from Walk to Bike and vice versa. This decision affects the modified edge's length value. For all the mentioned modifications, we can keep the value zero to indicate that the attribute was not changed, and one, when the attribute was flipped.

As a result, the CRP can be formulated as a problem to find the attributes modification vector a delivering

$$\text{lex} \min_{a \in \{0,1\}^{3n}} \left(\Delta(r(a), r^f), ||a||_1\right),$$

where
$n = |E|$ is the number of edges;
$r(a)$ is the fact route in the graph modified according to a;
r^f is user's foil route;
$\Delta(r, r')$ is computed by formula $\max(0, \texttt{RouteDist}(r, r') - \delta)$, where δ is a predefined threshold, and the distance between two routes is computed as

$$\texttt{RouteDist}(r, r') = 1 - 2\frac{\sum_{e \in r \cap r'} l_e}{\sum_{e \in r} l_e + \sum_{e \in r'} l_e}$$

To write the CRP in terms of mixed-integer programming, we need to consider the following entities.

Sets. Indices i and j take their values from the index set V of graph vertices. We call a combination of edge attributes an *edge type*. The complete set of edge types would be denoted by K, and k would denote an index from this set.

Parameters. m_{ijk} equals to the number of modifications needed to change the type of the edge (i, j) from its initial one into the type k;
l_{ij} is the length of the edge (i, j);
$\bar{l}_{ij}$ equals to l_{ij} if (i, j) is in the foil route and zero otherwise;
w_{ijk} is the modified length value of the edge (i, j) if it is transformed into the type k;
δ is a maximal admissible deviation from the foil route in terms of route distance.

Variables. x_{ijk} equals one if the edge (i, j) has a type k, and zero otherwise;
t_{ijk} equals one if the fact route traverses the edge (i, j) of type k, and zero otherwise.

Having the relations

$$\texttt{GraphDistance} = D^g = \sum_{i,j,k} m_{ijk} x_{ijk}; \tag{1}$$

$$\texttt{FactRouteLength} = L^* = \sum_{i,j,k} l_{ij} t^*_{ijk}; \tag{2}$$

$$\texttt{FoilRouteLength} = \bar{L} = \sum_{i,j} \bar{l}_{ij}; \tag{3}$$

$$\texttt{CommonEdgesLength} = C = \sum_{i,j,k} \bar{l}_{ij} t^*_{ijk}; \tag{4}$$

$$\texttt{RouteDist} = D^p = 1 - 2\frac{C}{L^* + \bar{L}}; \tag{5}$$

$$\sum_k x_{ijk} = 1; \quad x_{ijk} \in \{0, 1\}. \tag{6}$$

The goal is to lexicographically minimize the vector-function $(\max(D^p - \delta, 0), D^g)$ provided that (t^*_{ijk}) is the optimal solution of the shortest path problem $\mathcal{SPP}(x)$ parametrized by (x_{ijk}).

$$\min_{(t_{ijk})} \sum_{i,j,k} w_{ijk} t_{ijk}$$

$$\sum_{i,k} t_{ijk} - \sum_{i,k} t_{jik} = \begin{cases} -1, \text{ if } j \text{ is the origin} \\ 1, \text{ if } j \text{ is the destination} \\ 0, \text{ otherwise} \end{cases} ; \quad (\alpha_j)$$

$$0 \le t_{ijk} \le x_{ijk}. \quad (\beta_{ijk})$$

Having o and d from V to be the origin and destination of the route, consider the dual of the $\mathcal{SPP}(x)$

$$\max_{(\alpha_j),(\beta_{ijk})} -\alpha_o + \alpha_d - \sum_{i,j,k} x_{ijk} \beta_{ijk} \tag{7}$$

$$\alpha_j - \alpha_i - \beta_{ijk} \le w_{ijk} \tag{8}$$

$$\beta_{ijk} \ge 0 \tag{9}$$

The optimality condition for variables (t_{ijk}) is equivalent to the condition of equity for primal and dual objectives. This condition is non-linear and must be linearized using standard techniques before passing to the MIP-solver:

$$\sum_{i,j,k} w_{ijk} t_{ijk} = -\alpha_o + \alpha_d - \sum_{i,j,k} x_{ijk} \beta_{ijk}$$

The overall single-level bi-objective reformulation of the problem is the following one joined with the relations (1)–(5):

$$\min_{(x_{ijk}),(t_{ijk}),(\alpha_j),(\beta_{ijk}),\Delta} (\Delta, D^g) \tag{10}$$

$$\Delta \ge (L^* + \bar{L})(1 - \delta) - 2C \tag{11}$$

$$\sum_k x_{ijk} = 1 \tag{12}$$

$$\sum_{i,j,k} w_{ijk} t_{ijk} = -\alpha_o + \alpha_d - \sum_{i,j,k} x_{ijk} \beta_{ijk} \tag{13}$$

$$\sum_{i,k} t_{ijk} - \sum_{i,k} t_{jik} = \begin{cases} -1, \text{ if } j = o \\ 1, \text{ if } j = d \\ 0, \text{ otherwise} \end{cases} ; \tag{14}$$

$$0 \le t_{ijk} \le x_{ijk} \tag{15}$$

$$\alpha_j - \alpha_i - \beta_{ijk} \le w_{ijk} \tag{16}$$

$$\Delta, \beta_{ijk} \ge 0 \tag{17}$$

$$x_{ijk} \in \{0, 1\}. \tag{18}$$

3 Algorithms

3.1 Selection of Relevant Modifications

To reach good performance of the algorithms, one could concentrate efforts on considering only those attributes' modifications, which are relevant to the user's situation. The selection of relevant modifications follows the logic of the problem: we limit the choice of edges only to those which are different for fact and foil routes.

For edges, belonging to the foil route but not to the fact one, we are interested in "improving" modifications: making them available by increasing their width or lowering the curb (width increases are attempted before curb-lowering). The attractiveness of these edges can be improved also by selecting a proper path type, but this modification has a lower priority since, if the edge is unavailable due to small width or high curb, then the path type is not relevant.

Edges, which appear in the fact route but do not present in the foil one, should be modified in the opposite way: we consider decreasing of their width and heightening their curbs, and, with lower priority, changing of their path type.

3.2 Tree Search

We explore binary encodings of edge-attribute modifications in a best-first manner using a priority queue $\mathcal{Q}$. The encodings in the queue are ranked in ascending lexicographic order of pairs $\big(\Delta(a),\, \|a\|_1\big)$, where a is the encoding, $\Delta(a) = \max(\texttt{RouteDist}(r(a), r^f) - \delta, 0)$ is the route distance violation (Sec. 2) and $\|a\|_1$ the number of modifications.

Move Generation

For an encoding a considered, we derive children of the form $a + e_m$, where m is a modification chosen by one of the branching rules described below and e_m is a unit vector corresponding to this modification. The branching rules used are the following.

- *Heuristic branching rule:* restricts to edges in the fact or foil route as described in Section 3.1,
- *Complete branching rule:* considers all graph edges.

Deterministic Best–First

We repeatedly dequeue the encoding with the smallest $(\Delta, \|a\|_1)$ from $\mathcal{Q}$ and expand it under the chosen branching rule. If the best child generated by the rule has $\Delta = 0$, we put it in the queue. Otherwise, we enqueue a child obtained by enabling the edges of the foil route and forbidding the edges from the fact route until $\Delta = 0$. For any encoding with $\Delta = 0$, a greedy post-processing procedure is run that removes redundant modifications. Notice that if we find an encoding a such that $\Delta(a) = 0$, then we do not need to explore the tree at depth greater than or equal to $||a||_1$ further. This allows us to claim that the Tree Search algorithm utilizing the complete branching rule finds an optimal solution of the CRP.

3.3 Destroy-and-Repair Algorithm

The heuristic described here operates with *solutions* represented by lists of modifications. It alternates operations called "destroy", where some modifications are removed from the current solution, often making it infeasible in terms of route distance violation, and "repair", where new modifications are introduced instead of the ones removed. This strategy allows to escape local minima, allowing one to explore the search space efficiently.

Destroy Operators

We tested various destroy operators. The best-performing ones will be described in this section.

The *random destroy operator* randomly removes 10–30% of modifications from the current solution.

The *local search destroy operator* at each step generates 3 to 10 neighbor solutions by randomly removing a single modification from the current one. The neighbor with a minimal route distance becomes the current solution on the next step. The procedure repeats until it removes 10–30% of the modifications.

The *population-based swap* aims to find an important subset of edges where we need to introduce modifications. It operates with several solutions simultaneously. Each solution in the population is created by removing one random modification from the current solution. Let $\mathcal{P}$ be the set of resulting routes and r^f the foil route. We then: 1) promote edges in $r^f \setminus \bigcup_{r\in\mathcal{P}} r$ by adjusting their path type, and 2) forbid edges in $\bigcap_{r\in\mathcal{P}} \big(r \setminus r^f\big)$ by modifying edge attributes in a way that makes it unavailable for the user. The population must remain small, or these intersections may become empty, and thus we use $|\mathcal{P}| = 3$.

To remove redundant modifications from the current solution, we introduce two versions of *clean-up operators*. The first version iteratively tries to remove each modification from the current solution and then checks if the obtained solution is feasible, if so it calls the considered modification *useless*. After finishing the loop, it removes all useless modifications from the current solution. The second version is very similar to the first one. The difference is that it does not memorize useless modifications but removes them on the spot.

REPAIR Operator

At the start, we flip a 20% chance – if triggered, we immediately make all edges on the foil route available; otherwise we skip straight to the iterative loop. In that loop, we perform a local search: at each iteration, we add one new modification (which may include further foil-route improvements) that most reduces the route-distance violation. We repeat until $\Delta = 0$.

Multi-Start DR Workflow

We maintain multiple search workers that periodically share best solutions to intensify exploration in promising regions. At each synchronization point, a worker intersects its current solution with either its personal best or the global best, and then applies the REPAIR operator to re-expand the search around that overlapping region. To balance intensification and diversification, we evenly split the workers: half run with *intersectFlag = true* (intersecting with the global best) and half with *intersectFlag = false* (intersecting with their personal best).

Algorithm 1 Parallel Multi-start DR Heuristic

```
function MultiStartDR($G, r^f, s_0$)
    Parameters: $\mathcal{C}$, $\mathcal{D}$, intersectFlag
    $s \leftarrow$ Repair($s_0$)                ▷ initialize current solution
    $s^* \leftarrow s$                          ▷ personal best
    while time limit not reached do
        if isSyncTime then
            $s^*_{\text{glob}} \leftarrow$ SyncBest($s^*$)
            if intersectFlag then
                $s \leftarrow$ Repair( $s \cap s^*_{\text{glob}}$ )
            else
                $s \leftarrow$ Repair( $s \cap s^*$ )
            end if
        end if
        $c \leftarrow$ randomChoice($\mathcal{C}$)    ▷ pick a clean-up op
        $d \leftarrow$ randomChoice($\mathcal{D}$)    ▷ pick a destroy op
        for all $op \in [c,$ Repair$(\cdot)$, $d,$ Repair$(\cdot)]$ do
            $s \leftarrow op(s)$
            $s^* \leftarrow \arg\min_{x \in \{s,\, s^*\}} (\Delta(x),\ \|x\|_1)$
        end for
    end while
    $s^*_{\text{glob}} \leftarrow$ SyncBest($s^*$)
    return $s^*_{\text{glob}}$
end function
```

The operators are partitioned into two families: $\mathcal{C}$, which contains the two clean-up operators, and $\mathcal{D}$, which comprises the three other destroy operators. Whenever the predicate IsSyncTime() is evaluated to *true*, the routine SyncBest(x) is called to broadcast s^*, the best-found solution of the worker, and update the shared global best s^*_{glob}. The notation $s \cap s'$ denotes the set-theoretic intersection of the modification lists. The routine RandomChoice returns an element chosen uniformly at random from its input set.

3.4 Computational Results

Table 1 summarizes our aggregated performance over eight sets of benchmark instances. All experiments were run on a desktop with an Intel Core i7-13700 CPU, 32 GB of RAM, using 23 threads for DR runs. We evaluated on:

- **demo** (3 provided demo instances)
- **osdpm** (25 Amsterdam Osdorp-Midden instances)
- **bbox1–bbox3_0.15_bike**[1]

The second column of Table 1 reports the percentage of instances for which the best-known solution (BKS) was proven optimal by solving the linearized MIP model from Section 2 to optimality, with the deviation constraint $\Delta \leq 0$ enforced. The last two columns give the average gap to the BKS (found by DR) for the 1-thread best-first tree search with heuristic branching rule (TS) and multi-start DR, respectively. DR was run ten times per instance, whereas TS was run once, as it is deterministic. Gaps are averaged over all instances in the corresponding set. The time budget for the algorithms was set to 5 minutes. Across all sets, the mean gap for DR is 0.5% versus 12% for TS. Simpler baseline heuristics examined separately were less effective.

[1]Six sets generated via https://github.com/Amsterdam-AI-Team/Accessible_Route_Planning: 10 instances each, except 9 for bbox1.

Instance set	OPT proven	Average gap to BKS	
		TS	DR
demo	100%	0%	0%
osdpm	92%	1%	0%
bbox1	67%	6%	0%
bbox1-p	40%	7%	0%
bbox2-short	90%	0%	0%
bbox2-long	20%	21%	0%
bbox3	0%	31%	2%
bbox3_0.15_bike	0%	36%	3%
TOTAL	54%	12%	0.5%

Table 1: Aggregated results

Note: Detailed results are provided in [HGLM, 2025].

4 Conclusion

We addressed the counterfactual routing problem by formally defining it, deriving a single-level reformulation, and proposing TS and multi-start DR heuristics. Computational experiments show that DR reliably finds high-quality modification sets with minimal deviation. Our method produces concise graph modifications that can serve as counterfactual explanations for personalized route choices. This has the potential to support accessible navigation for users with accessibility needs and may help surface infrastructure limitations for urban planners. Ablation studies of individual DR components and sharing strategies remain important future work.

References

[Browne *et al.*, 2012] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, March 2012.

[HGLM, 2025] HGLM. CRP Toolkit – IJCAI-25. https://github.com/alukrogi/graph-optimization, 2025.

[Israeli and Wood, 2002] Eitan Israeli and R. Kevin Wood. Shortest-path network interdiction. *Networks*, 40:97–111, 2002.

[Pisinger and Røpke, 2010] David Pisinger and Stefan Røpke. Large Neighborhood Search. In Michel Gendreau, editor, *Handbook of Metaheuristics*, pages 399–420. Springer, 2nd edition, 2010.

AI-assistance disclosure – The draft was proof-read with a generative-AI tool to refine wording, identify technical inaccuracies, and improve concision. All suggestions were manually reviewed; no content, data, or citations were generated by the model.