Introducing Total Harmonic Resistance for Graph Robustness Under Edge Deletions

Lukas Berner(✉) and Henning Meyerhenke

Department of Computer Science, Humboldt-Universität zu Berlin,
Unter den Linden 6, 10099 Berlin, Germany
{lukas.berner,meyerhenke}@hu-berlin.de

Abstract. Assessing and improving the robustness of a graph G are critical steps in network design and analysis. To this end, we consider the optimisation problem of removing k edges from G such that the resulting graph has minimal robustness, simulating attacks or failures.

In this paper, we propose total harmonic resistance as a new robustness measure for this purpose – and compare it to the recently proposed forest index [Zhu et al., IEEE Trans. Inf. Forensics and Security, 2023]. Both measures are related to the established total effective resistance measure, but their advantage is that they can handle disconnected graphs. This is also important for originally connected graphs due to the removal of the k edges. To compare our measure with the forest index, we first investigate exact solutions for small examples. The best k edges to select when optimizing for the forest index lie at the periphery. Our proposed measure, in turn, prioritizes more central edges, which should be beneficial for most applications. Furthermore, we adapt a generic greedy algorithm to our optimization problem with the total harmonic resistance. With this algorithm, we perform a case study on the Berlin road network and also apply the algorithm to established benchmark graphs. The results are similar as for the small example graphs above and indicate the higher suitability of the new measure.

Keywords: Graph robustness optimization · infrastructure protection · total harmonic resistance · forest index · effective resistance

1 Introduction

The analysis of network[1] topologies, a major subarea of data science on network data, is key to understanding the functionality, dynamics, and evolution of networks [5,26]. An important property of a network in this context is its *robustness*, i. e., its ability to withstand failures of its components (or the extent of this ability) [5]. As an example, a typical question is whether a network remains (mostly) connected if a certain fraction of its vertices and/or edges are

[1] We use the terms *network* and *graph* interchangeably.

A. Bifet et al. (Eds.): ECML PKDD 2024, LNAI 14946, pp. 287–302, 2024.
https://doi.org/10.1007/978-3-031-70365-2_17

deleted [26, Ch. 15]. Despite the widespread use of vertex deletions, edge deletions can be more appropriate depending on the modeled phenomenon. Examples include, among others, road blocks in street or public transportation networks; pollution in water distribution networks; disruption of gas pipelines, energy grids, or computer/telecommunication networks. Such deletions may occur as a result of failure or of an attack; robustness thus is a critical design issue that arises in many application areas [13], e. g., various public infrastructures [9,19,33,37].

Due to economic reasons, it is unrealistic that all network components can be protected with the same effort. Thus, with the protection of critical infrastructure as application in mind, we consider the following optimization problem: given a graph $G = (V, E)$ and a budget of k graph edges to be removed, find the subset $S \subset E$ such that the robustness of $G' = (V, E \setminus S)$ is minimized. This problem, which we call k-GRoDel (short for *graph robustness problem with k deletions*), models a concurrent attack (or failure). The solution indicates which set of edges should be particularly safeguarded, e. g., segments in a water distribution network. Clearly, for a particular application, one must also instantiate this generic problem with a sensible notion of robustness.

Not surprisingly, numerous robustness measures have been proposed in the literature [5,33]. For the related problem of optimizing the robustness by *adding* k edges (called k-GRIP in Ref. [30], short for *graph robustness improvement problem*), *total effective resistance* was established as a meaningful robustness measure in various scenarios [12,29,31,36]. Effective resistance is a pairwise metric on the vertices of G; intuitively, it becomes small if there are many short paths between two vertices. Two disconnected vertices have infinite effective resistance, though. When total graph resistance were used in k-GRoDel, a trivial solution to maximize it would thus be to disconnect G. Yet, from an application's point of view, disconnecting a small part from the vast majority of the graph may be less problematic than a bottleneck (or a disconnection) between two large parts. Liu et al. [22] handles this issue by demanding that G is still connected after edge removal. Given an infrastructure scenario, this is a rather unnatural assumption. Zhu et al. [38] address the issue by proposing the forest index, $R_f(G)$, as robustness measure. Instead of effective resistance, $R_f(G)$ sums up the closely related forest distance [11] for all vertex pairs. Forest distance is derived from the number of certain rooted forests in G. It yields finite distance values also for disconnected vertex pairs.

Contribution. We show in this paper that k-GRoDel using the forest index favors peripheral edges in many networks. We deem this behavior unintuitive and, most importantly, undesirable for most applications. That is why we propose *total harmonic resistance* $R_h(\cdot)$ instead. This measure adds up the reciprocal of effective resistance for all vertex pairs, leading to a zero contribution of disconnected pairs (details in Sect. 2). The use of $R_h(\cdot)$ may seem like a straightforward extension to handle deletions given that the use of reciprocals is known for the popular (harmonic) closeness centrality (based on the ordinary graph distance) to handle disconnectedness [26]. Nonetheless, we are to our knowledge the first to investigate this notion of robustness (also see Sect. 3).

To substantiate the higher suitability of $R_h(\cdot)$ compared to the forest index in k-GRoDel, we first examine optimal solutions for small graphs with the two measures (Sect. 4). They clearly show that $R_h(\cdot)$ favors more central edges than the forest index and also finds balanced cuts in examples with suitable k.

Since exact solutions for either k-GRoDel measure are expensive to compute, we adapt in Sect. 5 the general greedy algorithm used in several previous papers for related problems. For $R_h(\cdot)$, our greedy algorithm differentiates between bridge edges and other edges. When a bridge edge is removed, a simple update operation for the Laplacian pseudoinverse does not work. For these cases, we thus provide specialized update functions. For the forest index, we derive a connection to total effective resistance, which allows the re-use of optimized Laplacian pseudoinverse solvers instead of more general matrix inversion solvers.

Our experiments (Sect. 6) include a case study on the road network of (a part of) Berlin, Germany, as well as numerous public benchmark graphs used before in related work. They show: (i) visually, the case study results indicate that the greedy solution for $R_h(\cdot)$ prefers more central edges than the one with the forest index. Maybe not surprisingly, one can find an even better solution regarding $R_h(\cdot)$ by choosing natural cut edges (river bridges in the road network) manually, which underlines the expressiveness of the new measure; (ii) for the benchmark graphs, a ranking based on closeness centrality confirms that greedy solutions of $R_h(\cdot)$ lead to more central edges than the forest index in most cases, too.

2 Problem Statement and a New Robustness Measure

2.1 Problem Statement and Notation

The input to k-GRoDel is an integer $k \in \mathbb{N}$ and a simple undirected graph $G = (V, E)$ with $|V| = n$, $|E| = m$. Given $S \subset E$, let $G' = (V, E \setminus S)$ be the graph with the edges from S removed. k-GRoDel aims at finding S with $|S| = k$ such that the robustness of G' is minimized (i. e., $R_h(G')$ is minimized or $R_f(G')$ is maximized, respectively).

We use well-known matrix representations of graphs. $\mathbf{L}_G = \mathbf{D} - \mathbf{A} \in \mathbb{R}^{n \times n}$ is the Laplacian matrix of G, where $\mathbf{D}$ is the vertex degree matrix and $\mathbf{A}$ is the adjacency matrix. $\mathbf{L}_G$ is symmetric (since G is undirected) and has zero row and column sums ($\mathbf{L1} = 0 = \mathbf{1}^T\mathbf{L}$). Since $\mathbf{L}_G$ is not invertible, the Moore-Penrose pseudoinverse $\mathbf{L}^\dagger$ is used instead (cf. [8]). When G has multiple components, $\mathbf{L}_G$ is a (permuted) block diagonal matrix where each block corresponds to one component of G.

2.2 Robustness Measures

Effective Resistance. Viewing the graph as an electrical circuit where each edge is a resistor, the effective resistance is the potential difference between two nodes u

and v when injecting [extracting] a unit current at u [v] [17]. It can be computed via $\mathbf{L}^\dagger$: $\mathbf{r}_G(u,v) = \mathbf{L}_G^\dagger[u,u] - 2\mathbf{L}_G^\dagger[u,v] + \mathbf{L}_G^\dagger[v,v]$ for nodes in the same component of G. For disconnected pairs, the resistance is infinite. As a robustness measure, one can take the sum over all pairwise effective resistances to compute the total effective resistance $R_r(G) = \sum_{u<v} \mathbf{r}_G(u,v)$, which has previously been used as optimization target for k-GRIP [29,31,36] in graphs with only one component. Combining both equations above results in a simple trace-based formula [8]:

$$R_r(G) = n \cdot \mathrm{tr}(\mathbf{L}_G^\dagger). \tag{1}$$

Forest Index (FI). To address the issue of disconnected graphs, other robustness measures are required. The *forest index*, based on the *forest distance* [11] $d_G^f(\cdot,\cdot)$, was proposed by Zhu et al. [38]. Similar to effective resistance, the forest distance is based on the *forest matrix* $\Omega = (L+I)^{-1}$, with $d_G^f(u,v) = \Omega[u,u] - 2\Omega[u,v] + \Omega[v,v]$. The forest distance is closely related to effective resistance (for details see Sect. 5), but yields finite values also for disconnected vertex pairs. Similar to total effective resistance, the forest index is the sum of the forest distance (instead of the effective resistance) of all ordered vertex pairs (u,v):

$$R_f(G) := \sum_{u<v} d_G^f(u,v). \tag{2}$$

With an argument analogous to the one for total effective resistance, the forest index can be expressed using the trace as well:

$$R_f(G) = n \cdot \mathrm{tr}(\Omega) - n. \tag{3}$$

Total Harmonic Resistance (THR). We now propose a new measure to handle disconnected graphs, *total harmonic resistance*. This measure is again based on the effective resistance; this time one sums up the reciprocal of all pairwise effective resistances – therefore *harmonic*:

$$R_h(G) := \sum_{u<v} \frac{1}{\mathbf{r}_G(u,v)}. \tag{4}$$

For vertex pairs where the effective resistance is infinite (i. e., vertices lie in different components), we define the reciprocal to be zero. The reciprocity in this sum makes computations more difficult compared to the other two metrics.

3 Related Work

Due to its high relevance in numerous application areas as well as a rich assortment of research questions, robustness in networks has been an active research area for several decades [13]. We thus point the interested reader to recent surveys for a broader overview [13,27]. Concerning robustness measures, the survey by Freitas et al. [13] categorizes them into three classes: (i) based on structural

(combinatorial) properties, (ii) spectral properties of the adjacency matrix, and (iii) spectral properties of the Laplacian matrix. Total effective resistance belongs to the third class as it can be computed by the sum of the Laplacian (inverse) eigenvalues. Chan and Akoglu [10] propose a budget-constrained edge rewiring mechanism to address six different spectral measures – a related optimization problem, yet different from k-GRODEL. Note that Oehlers and Fabian [27] focus on communication networks and use a more fine-grained categorization than Freitas et al. within their context.

Failures of components can result from various reasons, e. g., from natural disasters, attacks, or wear. The targeted attack models surveyed by Freitas et al. [13] refer to vertex removals and are based on vertex degrees and centrality scores. In general, vertex removals are the predominant failure model in the literature; Newman [26, Ch. 15] discusses percolation (removal of a fraction of the nodes), for example. An important question in this context is after which fraction the graph becomes disconnected or, more generally, when the giant component dissolves. One can address this question analytically in generative models (e. g., [26]) and/or empirically with real-world data (e. g., [6]). As a prime example, an influential paper by Albert et al. [1] and follow-up work led to the popular belief that scale-free networks are "robust-yet-fragile", i. e., robust against uniform vertex deletion and fragile against targeted attacks that remove high-degree vertices. Recent work by Hasheminezhad and Brandes [15] puts this view into a more nuanced perspective: robustness depends primarily on the graph's minimum degree, not a power-law degree distribution.

As mentioned in Sect. 1, edge deletions are natural to model failures in numerous applications. Liu et al. [22], who study the problem of minimizing one node's information centrality when removing k edges, argue that edge deletions are less intrusive than vertex deletions and that they provide a more fine-granular control of disruptions. To measure how easy two vertices can reach each other via alternative paths, numerous works use effective resistance [12,22,29,30,36], whereas Zhu et al. [38] use forest distance [11] as summands of their forest index, a metric related to effective resistance. Both robustness measures express with lower values that more alternative pathways exist. A small total forest distance (as well as a small total effective resistance) thus means that many vertex pairs can reach each other via many alternative short paths. Obviously, this is a desirable property for a robustness measure in a number of applications, e. g., when it comes to routing information or goods [27]. Forest distance has recently been used for forest closeness centrality [14,16]. There and when used as part of the forest index, it has the advantage (compared to the ordinary graph distance or effective resistance) to be able to handle disconnected graphs without changes.

An exact solution of the k-GRODEL problem with total harmonic resistance or a related measure is likely infeasible for instances of non-trivial size: (i) similar optimization problems have been shown to be $\mathcal{NP}$-hard [14,18], including the single-vertex variant of Liu et al. [22] with information centrality, and (ii) mathematical programming, even when applied to related problems with simpler objective functions, can usually solve instances with only hundreds or at

most a few thousand vertices in reasonable time [2]. Empirically, however, the related problem of adding k edges to minimize total effective resistance can be solved adequately (yet in general not optimally) with a standard greedy algorithm [35]. we developed in our previous work [30,31] heuristics to accelerate the greedy algorithm for the k-GRIP problem (with usually tolerable losses in solution quality). Even more closely related, Liu et al. [22] and Zhu et al. [38] use greedy strategies to identify k edges to delete from G while optimizing for a robustness measure (information centrality vs forest index). We thus expect an adapted greedy algorithm to work similarly well for our variant of k-GRoDel.

4 Comparison of Exact Solutions

To investigate the difference between forest index and total harmonic resistance as robustness measures for k-GRoDel, we analyze exact solutions for a collection of small examples. These examples consist of different graph classes: grid graphs and variants thereof, random graphs [26, Part III] generated using the Barabási-Albert model (parameters: $k = 3$, $n_{max} = 18$), and random graphs generated with the Watts-Strogatz model (parameters: $n = 16$, deg $= 3$, $p = 0.7$). For each example, we compute the exact solutions of the optimization problem for both robustness measures. Results for the grid-like grahps are visualized in Fig. 1. Due to symmetry in the grid-like graphs, there are often multiple optimal solutions; for simplicity, we only show one of these solutions.

Visually, the figures suggest that the forest index (FI) finds edges in the periphery, while THR finds central edges. THR also seems to be more robust regarding changes to low-degree nodes in the periphery of a graph: THR finds the same solution for the 4x7 grid and for the hotdog grid, while the FI solution changes.

To further support our claim about periphery vs center, we compute a centrality score for an edge set as follows: Given the closeness centrality $c(\cdot)$ of all nodes in $G = (V, E)$ and a set of edges $S \subset E$, we rank the nodes by their closeness centrality and convert their rank into a relative (quantile) score $s \in [0, 1]$, where the most central node has score 1 and the least central node has score 0. Then, for each edge $e = (u, v) \in S$, we take the mean score of both incident nodes and call this the score of that edge $s(e)$. Edges which are central in the graph have a larger score than less central edges. Finally, we define the score of S as the mean of all edge scores in the set. Scores for all solution sets are listed in Table 1. The centrality scores of the solutions further support our claim: for all graphs of all three types, the score of the THR solution is higher (i. e., more central) than in the FI solution. This even holds when comparing the best FI solution to the worst THR solution in this metric.

Discussion. We would like to note that the observed behavior of the forest index is not according to our original intuition (which was similar to the one given by Zhu et al. [38]) *before* working on this paper. Broadly speaking, we generally expected the forest index to be maximized when the number of disconnected

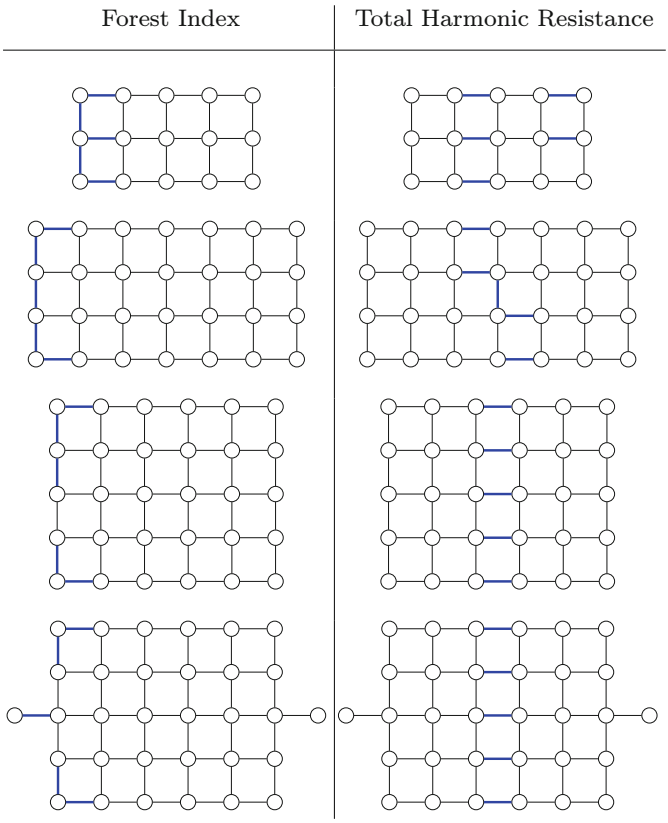

Fig. 1. Optimal solutions for $k = 5$ on grid-like graphs using FI (left column) and THR (right column) as resistance measures. Edges highlighted in blue belong to the solution set. (Color figure online)

vertex pairs is maximized, because this leads to many high terms in the sum. The optimal solution (for appropriate k) on a grid graph would then be a balanced cut in the middle. Instead, the optimal solution when using the forest index is a set of edges at the boundary of the grid, disconnecting just a few vertices from a large component. While such peripheral edges may be desirable in some applications, we argue that in most scenarios more central edges – whose deletion ideally even leads to several connected components – are beneficial from an attacker's point of view. We further explore this in our case study on (parts of) the Berlin road network in Sect. 6. To be able to process non-trivial instances, we propose to adapt a generic greedy algorithm in the next section.

5 Greedy Heuristic for k-GRoDel

We adapt the general greedy algorithm previously used for many related problems. The basic idea of this algorithm is to iteratively pick the edge with best marginal loss until k edges are found (see Algorithm 1).

Table 1. Solution set centrality scores as defined in Sect. 4. Since multiple optimal solutions exist for some graphs, the score is computed for each solution and aggregated in this table.

graph	BA1		BA2		BA3		grid5x3		grid5x6	
opt	FI	THR	FI	THR	FI	THR	FI	THR	FI	THR
min	0.31	0.40	0.29	0.47	0.32	0.40	0.24	0.53	0.09	0.69
mean	0.33	0.40	0.29	0.47	0.34	0.40	0.24	0.60	0.10	0.69
max	0.37	0.40	0.29	0.47	0.36	0.40	0.24	0.67	0.13	0.69
graph	grid7x4		hotdog5x6		WS1		WS2		WS3	
opt	FI	THR	FI	THR	FI	THR	FI	THR	FI	THR
min	0.11	0.76	0.14	0.71	0.34	0.49	0.27	0.34	0.16	0.28
mean	0.11	0.76	0.14	0.71	0.34	0.49	0.27	0.34	0.16	0.28
max	0.11	0.76	0.14	0.71	0.34	0.49	0.27	0.34	0.16	0.28

The greedy algorithm starts by computing the Laplacian pseudoinverse, which is required to compute the effective resistance and hence the loss when removing an edge from G. Then, in the main loop, the algorithm iterates all edges in G and computes the loss for each one. The best edge is picked, and G and $\mathbf{L}^\dagger$ are updated. The main loop runs for k iterations.

To implement the greedy algorithm, we need a formula to compute the marginal loss when removing an edge (Line 7) as well as a way to update $\mathbf{L}^\dagger$ after choosing an edge to compute the objective function in the next iteration (Line 10). These depend on the robustness metric used and will be derived in the next section.

For submodular functions the greedy framework can be combined with lazy evaluation [24] to speed up the computation. This lazy evaluation stores all candidates in a priority queue with their most recent loss value and instead of evaluating the loss for all candidates in each iteration of the main loop, it iteratively evaluates (and updates) only the top candidates' loss value until the top candidate is a candidate that has been evaluated in the current iteration of the main loop. Effectively, lazy evaluation reduces the number of evaluations significantly, while still providing a quality guarantee for submodular problems. Even though k-GRoDel is not known to be submodular for THR and is not submodular for FI [38], we still apply this technique because practical experience has proven to lead to good results even for non-submodular problems [25].

Combining the lazy evaluation technique with the general greedy algorithm and THR-based loss and update function leads to `GreedyTHR`: first, compute $\mathbf{L}_G^\dagger$.

Algorithm 1. Greedy algorithm for k-GRoDel

```
1: function Greedy(G, k)
2:     Input: Graph G = (V, E), k ∈ ℕ
3:     Output: G_k – graph after k edge deletions
4:     G_0 ← G
5:     Compute L†
6:     for r ← 0, ..., k − 1 do                          ▷ main loop
7:         Compute loss(e) ∀e ∈ E                        ▷ evaluation step
8:         e* ← argmax_{e∈E} loss(e)
9:         G_{r+1} = G_r \ e*
10:        Update(L†, G_{r+1})                           ▷ update step
11:    end for
12:    return G_{r+1}
13: end function
```

This takes $\mathcal{O}(n^3)$ time (with standard tools in practice). Then, compute the loss for all edges of G and set up a priority queue of all edges by their respective loss value. In the main loop, get the top entry from the priority queue (using lazy evaluation), remove that edge from G and update $\mathbf{L}^\dagger_{G_r}$.

5.1 Total Harmonic Resistance Loss After Deleting an Edge

We now derive an update formula and state the loss formula for THR, which are required for the greedy algorithm.

For the Update step (Line 10) there are efficient ways to compute $\mathbf{L}^\dagger_{G'}$: Removing an edge $e = \{a, b\} \in E$ from G results in $G' = (V, E \setminus \{e\})$ and $\mathbf{L}_{G'} = \mathbf{L}_G - (\mathbf{e}_a - \mathbf{e}_b)(\mathbf{e}_a - \mathbf{e}_b)^T$, where $\mathbf{e}_i$ is the i-th unit vector. One can apply the Sherman-Morrison-Formula [34] (which holds for $\mathbf{L}$ and $(\mathbf{e}_a - \mathbf{e}_b)$ as well) to write:

$$\begin{aligned}\mathbf{L}^\dagger_{G'} &= \mathbf{L}^\dagger_G + \frac{\mathbf{L}^\dagger_G(\mathbf{e}_a - \mathbf{e}_b)(\mathbf{e}_a - \mathbf{e}_b)^T\mathbf{L}^\dagger_G}{1 - (\mathbf{e}_a - \mathbf{e}_b)^T\mathbf{L}^\dagger_G(\mathbf{e}_a - \mathbf{e}_b)} \\ &= \mathbf{L}^\dagger_G + \frac{\mathbf{L}^\dagger_G(\mathbf{e}_a - \mathbf{e}_b)(\mathbf{e}_a - \mathbf{e}_b)^T\mathbf{L}^\dagger_G}{1 - \mathbf{r}_G(a, b)}\end{aligned} \tag{5}$$

There are limitations to using the Sherman-Morrison-Formula for updates: if the removed edge is a bridge, $\mathbf{r}_G(\cdot, \cdot) = 1$ [23] and hence the denominator in Eq. 5 is 0. In case the edge is not a bridge though, we can apply the Sherman-Morrison-Formula.

To handle the case of a bridge edge e, some more involved computation is required. Recall that $\mathbf{L}$ is a (permuted) block diagonal matrix where each block corresponds to a component of G (see Sect. 2). Removing e causes the corresponding block in $\mathbf{L}$ to be split into two blocks – one for each component. All other blocks of $\mathbf{L}$ are not modified by this edge removal. Since the pseudoinverse of a block diagonal matrix is the block matrix build from the pseudoinverse of

each block, $\mathbf{L}^\dagger_{G'}$ can be found by computing the pseudoinverse of the two blocks related to e and re-using the other pseudoinverse blocks from $\mathbf{L}^\dagger_G$.

One has to keep track of a mapping from each node to its component (since in general $\mathbf{L}$ is permuted and we need to know which row/column belongs to which block) which takes $\mathcal{O}(n+m)$ time. For simplicity, we re-compute this after removing a bridge edge (because the pseudoinversion step dominates the running time), but in principle it is possible to dynamically update the connected components instead. The running time of the update step is either $\mathcal{O}(c^2)$ (non-bridge edge) or $\mathcal{O}(\max(n+m, c^3))$ (bridge edge), where c is the size of the block matrix (resp. component of G) that contains e.

For the loss function, the basic formula is $\text{loss}(a,b) := R_h(G) - R_h(G') = \sum_{u<v} \frac{1}{\mathbf{r}_G(u,v)} - \frac{1}{\mathbf{r}_{G'}(u,v)}$. This formula depends on values in $\mathbf{L}^\dagger_G$ and $\mathbf{L}^\dagger_{G'}$ (via $\mathbf{r}_G(u,v) = \mathbf{L}^\dagger_G[u,u] + \mathbf{L}^\dagger_G[v,v] - 2\mathbf{L}^\dagger_G[u,v]$). Since this is a sum of reciprocal values, deriving an efficient formula proves difficult; we do not know of a closed formula analogous to the forest index or effective resistance yet. Using the basic formula to compute the loss requires computing $\mathbf{L}^\dagger_{G'}$ which we have discussed above. Computing the loss when $\mathbf{L}^\dagger_{G'}$ is given takes $\mathcal{O}(n^2)$ time and the loss is computed up to $\mathcal{O}(km)$ times in the greedy algorithm (in the worst case, the loss is computed for each edge even though we use lazy evaluation). This leads to $\mathcal{O}(kmn^2 \cdot \max(n+m, c^3))$ time overall for the loss computation. The running time varies a lot depending on the size and number of the components in G and the number of bridge edges.

5.2 Forest Index Loss After Deleting an Edge

For our experiments, we are also interested in results from the greedy algorithm for FI. Since the experimental setup by Zhu et al. [38] is to our knowledge not publicly available, we implemented our own version of this algorithm. There are two differences in our implementation compared to the algorithm description given in their paper though: (i) we exploit a connection between forest index and effective resistance to convert the FI computation back to a problem that is based on the Laplacian matrix. This allows re-use of specialized Laplacian pseudoinverse solvers. (ii) we use the lazy evaluation technique described in Sect. 5, even though FI is not submodular. As mentioned, this technique usually yields good results even for non-submodular problems and in our preliminary experiments we observed no difference in the solution quality.

To derive a marginal loss formula for the forest index, we use a theorem on the connection between effective resistance and forest distance; it allows to reduce the forest index formula (based on forest distance) back to total effective resistance and this reduction facilitates the reuse of some other theorems and algorithms:

Theorem 1. *Given $G = (V, E)$, define the* augmented Graph *$G_* = (V_*, E_*)$ with a universal vertex u^* which is connected to all other vertices: $V_* = V \cup \{u^*\}$ and $E_* = E \cup \{(v, u^*) : v \in V\}$.*

Then $d^f_G(u,v) = \mathbf{r}_{G_*}(u,v)\ \forall u, v \in V$.

The proof (with a slight change in the forest distance definition that does not affect the validity of the result) can be found in Ref. [11, Proposition 7]. From Theorem 1 the following result can be derived:

Proposition 1. *The forest index can be written in terms of the effective resistance of the augmented graph:* $R_f(G) = n \cdot \mathrm{tr}(\mathbf{L}^\dagger_{G_*}) - (n+1) \cdot \mathbf{L}^\dagger_{G_*}[u^*, u^*]$.

Proof. See Appendix A.1. The main idea is to extend the forest index sum by adding a zero term, which then includes the trace of $\mathbf{L}^\dagger_{G_*}$.

Edge Removal. We can now use Proposition 1 to write the forest index $R_f(G)$ in terms of the augmented graph G_* and $\mathbf{L}^\dagger_{G_*}$, which allows us to compute the marginal loss via $\mathbf{L}^\dagger_{G_*}$ when removing an edge from G.

Removing an edge $\{a, b\} \in E$ from G results in $G' = (V, E \setminus \{\{a, b\}\})$ and $\mathbf{L}_{G'_*} = \mathbf{L}_{G_*} - (\mathbf{e}_a - \mathbf{e}_b)(\mathbf{e}_a - \mathbf{e}_b)^T$, where $\mathbf{e}_i$ is the i-th unit vector. Apply the Sherman-Morrison-Formula [34] to write:

$$\mathbf{L}^\dagger_{G'_*} = \mathbf{L}^\dagger_{G_*} + \frac{\mathbf{L}^\dagger_{G_*}(\mathbf{e}_a - \mathbf{e}_b)(\mathbf{e}_a - \mathbf{e}_b)^T \mathbf{L}^\dagger_{G_*}}{1 - \mathbf{r}_{G_*}(a, b)} \tag{6}$$

To calculate the $\mathrm{loss}(a, b) := R_f(G') - R_f(G)$ when removing $e = \{a, b\}$ from G, we can use Eqs. (1) and (6) and the connection to total effective resistance (Proposition 1):

Proposition 2. *The marginal loss for the forest index when removing edge* (a, b) *from* G *is:*

$$\begin{aligned} \mathit{loss}(a, b) = &\frac{n}{1 - \mathbf{r}_{G_*}(a, b)} \cdot \left\| \mathbf{L}^\dagger_{G_*}[:, a] - \mathbf{L}^\dagger_{G_*}[:, b] \right\|^2 \\ &- \frac{n+1}{1 - \mathbf{r}_{G_*}(a, b)} \cdot (\mathbf{L}^\dagger_{G_*}[u^*, a] - \mathbf{L}^\dagger_{G_*}[u^*, b])^2. \end{aligned}$$

Proof. See Appendix A.2.

We use these loss and update formulae in our greedy algorithm, which allows us to re-use existing code. Running times for the loss and update computation are $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$ respectively, which results in a overall running time of $\mathcal{O}(kmn)$ and $\mathcal{O}(kn^2)$ for k iterations of the greedy algorithm.

6 Experimental Results

6.1 Experimental Setup

We conduct experiments to evaluate the quality of the greedy solution for THR to the greedy solution for FI (`GreedyTHR` and `GreedyFI`). Our algorithms are implemented in C++ using the NetworKit toolkit [3] as a graph library. We also

build upon the previous work in Refs. [30,31]. To solve linear systems and compute the pseudoinverse, we use the LAMG solver from NetworKit. SimExPal [4] is used to manage our experiments and analyze the results. All experiments are run on a machine with an Intel Xeon 6126 CPU and 192 GB RAM. Code and the experimental setup are available on github: https://github.com/bernlu/GRoDel-THR-FI.

Table 2 lists all networks used in our case study and benchmark study with their approximate number of nodes and edges. For the following analysis, we split them into two groups: *small* graphs with $|V| < 50K$ and *large* graphs with $|V| > 50K$. These networks are taken from SNAP [21], Networkrepository [32] and KONECT [20]. For our experiments, we perform preprocessing on these graphs to turn them into simple graphs by removing self-loops, multi-edges and edge weights; we use the largest connected component of each graph. We set the accuracy parameter ϵ of our LAMG solver (which we use to compute $\mathbf{L}^{\dagger}$) to 10^{-5}.

Table 2. Graph instances used for experiments, their vertex and edge counts after preprocessing, and the mean closeness centrality of the greedy THR and FI solutions.

Graph	$\|V\|$	$\|E\|$	THR	FI
euro-road	1K	1.3K	0.661	0.160
EmailUniv	1K	5.4K	0.405	0.526
air-traffic-control	1.2K	2.4K	0.579	0.063
inf-power	4K	6K	0.858	0.257
web-spam	4K	37K	0.457	0.039
Bcspwr10	5.3K	8.2K	0.301	0.740
Erdos992	6K	7.5K	0.643	0.691
Reality	6.8K	7.6K	0.825	0.508
Mitte-Berlin-Germany	1K	1.5K	0.648	0.334
Treptow-Köpenick-Berlin-Germany	3.6K	5.2K	0.733	0.283

6.2 Case Study: Berlin Districts

For our case study, we use the road networks of two Berlin districts (Table 2). We choose these networks because road networks in general are easy to visualize and understand intuitively; Berlin specifically has some rivers flowing through the city which create cuts for many districts. These river bridges make for a natural solution to k-GRoDel which we will use as a manually chosen solution to compare to the greedy solutions.

Our graphs are generated from OpenStreetMap [28] data using the osmnx [7] python library. We convert the data into a simple graph and use our NetworKit-based greedy algorithm to find the solution for both THR and FI.

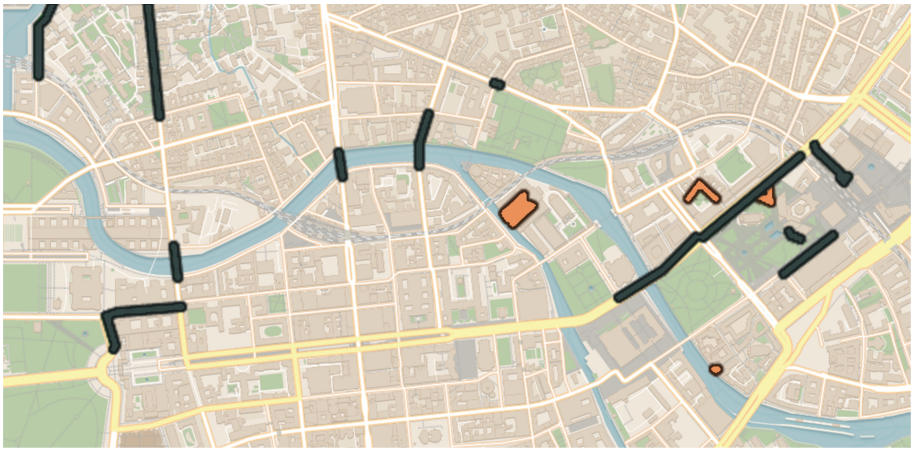

Fig. 2. Berlin case study result. Grey edges are the `GreedyTHR` solution; orange edges are the `GreedyFI` solution. The image is cropped – not all edges in the solution are displayed. `GreedyTHR` finds four of seven river bridges while `GreedyFI` mostly finds residential roads. Image created using OpenStreetMap [28]

The solutions for the *Mitte* district are drawn on a map for visual inspection (Fig. 2). One can clearly see that `GreedyTHR` finds some of the river bridges and other main roads, while `GreedyFI` finds less important streets. We also compare the solutions to the hand-picked solution that consists of the seven river bridges in this district (12 edges in total in our network because of multi-lane bridges). The THR of this manual solution is larger than that of the `GreedyTHR` solution, even though the greedy solution was computed for $k = 20$ edges – this further indicates that THR is a metric that prioritizes edges in a way we consider desirable. In contrast to the previous observation, the FI score of the manual solution is *worse* than the solution (of the same size) found by `GreedyFI`; this is another hint that FI prioritizes edges in the periphery of a network. We have also investigated other districts of Berlin, with very similar results: `GreedyTHR` finds some bridges and large streets; `GreedyFI` finds edges in the periphery and a manual choice of river bridges is better than the greedy solution.

6.3 Benchmark Results

For the benchmark graphs, we evaluate the results by comparing the average closeness centrality of the solutions using the same we method described for the exact solutions in Sect. 4.

Results are available in Table 2. For most benchmark graphs we observe that the `GreedyTHR` solution is considerably more central than the `GreedyFI` solution; on average, the `GreedyTHR` solution is about 25% more central in the closeness centrality metric. In the *Bcspwr10* graph `GreedyTHR` provides a considerably

less central solution than `GreedyFI` – though there is no obvious reason for this result.

Regarding running times, with a timeout of 12 h we found solutions for graphs with up to 6.8K nodes or 13K edges. We observe that the network structure (esp. the amount of bridge edges) has significant impact on running times – which is expected given the two ways to compute the update step, where the update for bridge edges is much more expensive. As expected, running times for `GreedyFI` are 2-4 orders of magnitude lower than `GreedyTHR`. The reason for this is that we can use the much more efficient loss formula using the trace of $\mathbf{L}^\dagger$ for `GreedyFI` while we do not know of an analogous formula for `GreedyTHR`.

7 Conclusions

With the protection of large infrastructure in mind, we considered the k-GRoDel problem to identify a set of k particularly vulnerable edges in a graph. To this end, we proposed total harmonic resistance as objective function and compared it against the recently proposed forest index.

We show with small examples where we compute the exact solution that total harmonic resistance prioritizes more central edges than the forest index. We adapt the general greedy algorithm for similar optimization problems to k-GRoDel with total harmonic resistance and show in a case study on the Berlin road network that THR favors more central edges in larger examples as well. Finally, we run benchmark experiments which show that THR mostly favors more central edges than FI in a range of different network types. We note that the greedy algorithm for THR has higher time complexity than the greedy algorithm for FI and our experiments confirm this in practice.

In the future, we would like to focus on speeding up the greedy algorithm for THR by improving the update and loss formulae and by finding other, faster heuristics. These are highly complex problems because of the reciprocity in the objective function – which prevents re-use of most of the results and techniques used for related robustness measures like total effective resistance or forest index.

Acknowledgments. We would like to thank Rob Kooij from TU Delft for insightful discussions on total harmonic resistance and many related measures.

References

1. Albert, R., Jeong, H., Barabási, A.L.: Error and attack tolerance of complex networks. Nature **406**(6794), 378–382 (2000)
2. Angriman, E., Becker, R., D'Angelo, G., Gilbert, H., van der Grinten, A., Meyerhenke, H.: Group-harmonic and group-closeness maximization - approximation and engineering. In: Proceedings of the Symposium on Algorithm Engineering and Experiments, ALENEX, pp. 154–168. SIAM (2021). https://doi.org/10.1137/1.9781611976472.12

3. Angriman, E., van der Grinten, A., Hamann, M., Meyerhenke, H., Penschuck, M.: Algorithms for large-scale network analysis and the networkit toolkit. In: Bast, H., Korzen, C., Meyer, U., Penschuck, M. (eds.) Algorithms for Big Data. LNCS, vol. 13201, pp. 3–20. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-21534-6_1
4. Angriman, E., et al.: Guidelines for experimental algorithmics: a case study in network analysis. Algorithms **12**(7), 127 (2019). https://doi.org/10.3390/a12070127
5. Barabási, A.L., Pósfai, M.: Network Science. Cambridge University Press, Cambridge (2016)
6. Beygelzimer, A., Grinstein, G., Linsker, R., Rish, I.: Improving network robustness by edge modification. Physica A **357**(3), 593–612 (2005). https://doi.org/10.1016/j.physa.2005.03.040. https://www.sciencedirect.com/science/article/pii/S0378437105003523
7. Boeing, G.: OSMnx: new methods for acquiring, constructing, analyzing, and visualizing complex street networks. Comput. Environ. Urban Syst. **65**, 126–139 (2017)
8. Bozzo, E., Franceschet, M.: Resistance distance, closeness, and betweenness. Soc. Netw. **35**(3), 460–469 (2013). https://doi.org/10.1016/j.socnet.2013.05.003
9. Cats, O., Koppenol, G.J., Warnier, M.: Robustness assessment of link capacity reduction for complex networks: application for public transport systems. Reliab. Eng. Syst. Saf. **167**, 544–553 (2017)
10. Chan, H., Akoglu, L.: Optimizing network robustness by edge rewiring: a general framework. Data Min. Knowl. Disc. **30**, 1395–1425 (2016)
11. Chebotarev, P.Y., Shamis, E.: The forest metrics of a graph and their properties. Automation Remote Control C/C of Avtomatika I Telemekhanika **61**(8; Issu 2), 1364–1373 (2000)
12. Ellens, W., Spieksma, F., Van Mieghem, P., Jamakovic, A., Kooij, R.: Effective graph resistance. Linear Algebra Appl. **435**(10), 2491–2506 (2011)
13. Freitas, S., Yang, D., Kumar, S., Tong, H., Chau, D.H.: Graph vulnerability and robustness: a survey. IEEE Trans. Knowl. Data Eng. **35**(6), 5915–5934 (2022)
14. van der Grinten, A., Angriman, E., Predari, M., Meyerhenke, H.: New approximation algorithms for forest closeness centrality - for individual vertices and vertex groups. In: Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021, pp. 136–144. SIAM (2021). https://doi.org/10.1137/1.9781611976700.16
15. Hasheminezhad, R., Brandes, U.: Robustness of preferential-attachment graphs. Appl. Netw. Sci. **8**(1), 32 (2023). https://doi.org/10.1007/s41109-023-00556-5
16. Jin, Y., Bao, Q., Zhang, Z.: Forest distance closeness centrality in disconnected graphs. In: 2019 IEEE International Conference on Data Mining (ICDM), pp. 339–348. IEEE Computer Society (2019). https://doi.org/10.1109/ICDM.2019.00044. https://doi.ieeecomputersociety.org/10.1109/ICDM.2019.00044
17. Klein, D., Randic, M.: Resistance distance. J. Math. Chem. **12**, 81–95 (1993). https://doi.org/10.1007/BF01164627
18. Kooij, R.E., Achterberg, M.A.: Minimizing the effective graph resistance by adding links is NP-hard. arXiv preprint arXiv:2302.12628 (2023)
19. Koç, Y., Warnier, M., Van Mieghem, P., Kooij, R.E., Brazier, F.M.: A topological investigation of phase transitions of cascading failures in power grids. Phys. A **415**, 273–284 (2014)
20. Kunegis, J.: KONECT: the koblenz network collection. In: Carr, L., et al. (eds.) 22nd International World Wide Web Conference, WWW 2013, pp. 1343–1350. International World Wide Web Conferences Steering Committee/ACM (2013). https://doi.org/10.1145/2487788.2488173

21. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection (2014). http://snap.stanford.edu/data
22. Liu, C., Zhou, X., Zehmakan, A.N., Zhang, Z.: A fast algorithm for moderating critical nodes via edge removal. IEEE Trans. Knowl. Data Eng. **36**(4), 1385–1398 (2024). https://doi.org/10.1109/TKDE.2023.3309987
23. Mavroforakis, C., Garcia-Lebron, R., Koutis, I., Terzi, E.: Spanning edge centrality: Large-scale computation and applications. In: Proceedings of the 24th International Conference on World Wide Web, pp. 732–742. International World Wide Web Conferences Steering Committee (2015)
24. Minoux, M.: Accelerated greedy algorithms for maximizing submodular set functions. In: Stoer, J. (ed.) Optimization Techniques, pp. 234–243. Springer, Heidelberg (1978). https://doi.org/10.1007/BFb0006528
25. Minoux, M.: Networks synthesis and optimum network design problems: models, solution methods and applications. Networks **19**(3), 313–360 (1989). https://doi.org/10.1002/net.3230190305
26. Newman, M.: Networks, 2nd edn. Oxford University Press, Oxford (2018)
27. Oehlers, M., Fabian, B.: Graph metrics for network robustness-a survey. Mathematics **9**(8) (2021). https://doi.org/10.3390/math9080895. https://www.mdpi.com/2227-7390/9/8/895
28. OpenStreetMap contributors: OpenStreetMap database (2017). https://www.openstreetmap.org
29. Pizzuti, C., Socievole, A.: A genetic algorithm for enhancing the robustness of complex networks through link protection. In: Aiello, L.M., Cherifi, C., Cherifi, H., Lambiotte, R., Lió, P., Rocha, L.M. (eds.) COMPLEX NETWORKS 2018. SCI, vol. 812, pp. 807–819. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05411-3_64
30. Predari, M., Berner, L., Kooij, R., Meyerhenke, H.: Greedy optimization of resistance-based graph robustness with global and local edge insertions. Soc. Netw. Anal. Mining (2023, to appear). Also available as arXiv preprint 2309.08271
31. Predari, M., Kooij, R., Meyerhenke, H.: Faster greedy optimization of resistance-based graph robustness. In: IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2022, Istanbul, Turkey, 10–13 November 2022, pp. 1–8. IEEE (2022)
32. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: AAAI (2015). http://networkrepository.com
33. Rueda, D.F., Calle, E., Marzo, J.L.: Robustness comparison of 15 real telecommunication networks: Structural and centrality measurements. J. Netw. Syst. Manage. **25**(2), 269–289 (2017)
34. Sherman, J., Morrison, W.J.: Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. Ann. Math. Stat. **21**(1), 124–127 (1950)
35. Summers, T., Shames, I., Lygeros, J., Dörfler, F.: Topology design for optimal network coherence. In: 2015 European Control Conference (ECC), pp. 575–580. IEEE (2015)
36. Wang, X., Pournaras, E., Kooij, R.E., Mieghem, P.V.: Improving robustness of complex networks via the effective graph resistance. Eur. Phys. J. B **87**, 1–12 (2014)
37. Yazdani, A., Jeffrey, P.: Complex network analysis of water distribution systems. Chaos **21**, 016111 (2011)
38. Zhu, L., Bao, Q., Zhang, Z.: Measures and optimization for robustness and vulnerability in disconnected networks. IEEE Trans. Inf. Forensics Secur. **18**, 3350–3362 (2023)