Optimal Cost-based Strengthening of Complex Networks

Qingnan Rong, Jun Zhang, Xiaoqian Sun*, Sebastian Wandelt, Massimiliano Zanin, and Liang Tian

Abstract—Most real-world complex systems are extremely vulnerable to targeted attacks, making their immunization an important yet challenging task. One of the most effective attack strategies is targeting articulation points specifically. In this study, we first propose a generalized definition of network robustness. Then we address the problem of strengthening network robustness with the minimum cost by exploiting the intuition behind the attack strategy based on articulation points, that is proven to be NP-hard. Accordingly, we propose a heuristic for solving this problem, subject to a cost function whose choice determines the obtained network regime. Experiments on both random and real-world networks show that our algorithm strengthens the robustness by using significantly cheaper edge additions than state-of-the-art methods. Moreover, our algorithm excels against general attack strategies by revealing the essence of strengthening network robustness, that is, increasing the size of the giant connected component optimally in the process of node removal. While considering the realistic problem, our algorithm also provides a reasonable scheme to add edges at a low cost.

Index Terms—network robustness, targeted attacks, cost, strengthening problem, heuristic.

1 Introduction

Within the large body of literature on complex networks, the topic of network disruption has received much attention. Such interest is explained by the fact that network robustness is strongly related to topological properties, like the presence of scale-free or modular structure, and is thus a fertile ground for theoretical studies [1], [2], where network robustness is defined as the ability to withstand the disruption and measured by network efficiency [3], percolation threshold [4], [5], [6], or the size of the giant connected component (GCC) [7], [8], [9], [10], etc. Besides, the estimation of the robustness of real-world networked systems is invaluable for policy-making [11].

Most of these studies have focused on designing algorithms to disrupt networks, i.e., removing a subset of nodes (or edges) to split the network into the largest number of isolated components [12]. Three types of disruptions are proposed: random, targeted, and acquaintance attack. Random attack removes nodes randomly without knowing the network structure. Targeted attack removes the most important node ordered by the node centrality, such as degree, betweenness, etc [13], [14]. If the node centrality is calculated once, it is a simultaneous targeted attack. But if the node centrality is recalculated after removing a node, it is a sequential targeted attack [8]. Acquaintance attack is between random attack and targeted attack, and destroys the network integrity by removing a small fraction of nodes without requiring specific knowledge of the network [15].

However, less attention has been devoted to the complementary problem, i.e., how to change the network topology to strengthen the robustness while keeping changes as minimum as possible. Random networks are robust against targeted attack but vulnerable against random attack. Real-world networks with power-law degree distribution are robust against random attack but vulnerable against targeted attack [4], [8], [16], [17]. Therefore, strengthening the robustness is to change the topology towards the robust one [9]. State-of-the-art methods for network strengthening can be divided into two classes: (i) Swap edges (SE) randomly and accept the change only if the robustness is strengthened [7], [18]. (ii) Add edges according to specific heuristics [19], [20], [21], [22]. Among the latter class, possibilities include adding edges between the lowest degree nodes (Preferential Addition, PrefA) [19], or following a more sophisticated strategy based on weak cores and the critical giant component (Posteriorly Addition, PostA) [21]. However, for these methods, the cost of topological modifications is measured by simply counting the number of swapped or added edges, leading to unrealistic results.

Recently, a sequential targeted attack called *articulation points-targeted attack* (APTA) [23] was proposed and proven to be more effective than other well-known attack strategies such as the high-degree adaptive attack [24] and the collective influence attack [25]. *Articulation point* (AP) is a node whose removal disconnects the network or increases the number of connected components. APTA iteratively removes the most destructive AP, whose removal causes most nodes disconnected from the GCC until no AP exists in the GCC. The residual GCC is called a *residual giant bicomponent* (RGB). The most robust network against APTA

*Corresponding author email: sunxq@buaa.edu.cn

- *Q. Rong, X. Sun and S. Wandelt are with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China (e-mail: rongqn@buaa.edu.cn; sunxq@buaa.edu.cn; wandelt@informatik.hu-berlin.de).*
- *J. Zhang is with the Advanced Research Institute of Multidisciplinary Science, Beijing Institute of Technology, Beijing 100081, China (e-mail: buaazhangjun@vip.sina.com).*
- *M. Zanin is with the Instituto de Física Interdisciplinary Sistemas Complejos (IFISC), CSIC-UIB, 07122 Palma de Mallorca, Spain (e-mail: assimiliano.zanin@gmail.com).*
- *L. Tian is with the Department of Physics and Institute of Computational and Theoretical Studies, Hong Kong Baptist University, Kowloon, Hong Kong SAR, China (e-mail: liangtian@hkbu.edu.hk).*

has no AP. Therefore, the network strengthening problem against APTA has a well-known sub-problem, *Biconnectivity Augmentation Problem* [26] whose object is to add a minimum-cost set of edges to biconnect a network, making any two nodes be connected by at least two independent paths. Two typical algorithms for solving this problem are approximation (Approx) algorithm [27] and fixed-parameter (FPT) algorithm [28]. Approx algorithm finds a solution biconnecting the network within a factor two of the optimum. FPT algorithm achieves biconnectivity using at most a certain number of new edges with the minimum cost. But they can not strengthen the robustness significantly using the limited budget that is not enough to biconnect the network.

The AP exists in the path structure of the network whose removal disconnects the network. To reduce the impact of its removal, we can cover it by adding edges. Based on this idea, we propose an effective network strengthening algorithm. We illustrate how our algorithm can be modified to account for the cost of creating new edges in terms of the distance between the connected nodes. Even with this limitation, our algorithm still outperforms state-of-the-art techniques by orders of magnitudes. Further, against general attack strategies, our algorithm excels by revealing the essence of strengthening the network robustness, that is, reducing the impact of AP removal. Our algorithm also provides a reasonable suggestion on adding edges at a low cost for the realistic problem.

2 Theoretical Definition

2.1 Background: network robustness

The first proposal for a metric assessing the network robustness was presented in Ref. [7]. Given a network with N nodes, the robustness is defined as $R = \frac{1}{N}\sum_{q=1/N}^{1} s(q)$ where $s(q)$ is the fraction of nodes in the GCC after removing qN nodes.

This robustness measurement requires an attack to remove all nodes from the network eventually. It is not generic, as some attack strategies (including APTA) only target a subset of nodes; nor realistic, as the marginal cost of disrupting additional nodes in real-world networks may be higher than the induced damages. If the attack removes a subset of nodes, R is not suitable to measure the robustness because it loses comparability. Suppose that networks A and B have the same number of nodes N. APTA removes a few nodes from network A and the RGB size is large, but it removes many nodes from network B and the RGB size is small. Obviously, network A is more robust, but its robustness R could be smaller than that of network B. More explanations can be seen in Appendix A. To make the robustness comparable, we propose a modified version of R. Suppose that an attack terminates after removing q_{max} fraction of nodes. We define the generalized robustness as

$$R_g = \frac{1}{N+1}\left(\sum_{q=0}^{q_{max}} s(q) + \sum_{q=q_{max}+1/N}^{1} s(q_{max})\right) \quad (1)$$

Then the value of R_g for network A is greater than that for network B. R_g fulfills the requirements of any R-like robustness measurement. Specifically, R_g is between zero (for a star network with infinite nodes) and one (when the attack has no effect on the network). The calculation of R_g is based on a specific attack strategy. We here leverage on the APTA for the sake of completeness. We also prove that R_g measures the network robustness against APTA more appropriately than other measurements in Appendix B.

2.2 Defining the network strengthening problem

The problem of strengthening a network is defined as:

Definition 2.1. An undirected network $G = (V, E)$ consists of a set of nodes V and a set of edges E. The network strengthening problem is adding a minimum-cost set of edges E_1 to G to make the network robustness R_g greater than or equal to t, where t belongs to $[0, 1]$.

The cost of adding edges can be arbitrary. In Ref. [29], the cost of destroying a node with degree k is assumed as $c(k) = k^{\alpha}$, where α determines the optimal attack strategy. Similarly, not all added edges have the same cost in practice. For instance, in subway or railway systems, the cost of building an edge often increases as a function of its (spatial) length. Thus, we define the cost of adding an edge (a, b) as

$$c((a, b)) = [SPL(a, b)]^{\alpha} \quad (2)$$

where $SPL(a, b)$ is the shortest path length between nodes a and b, and α belongs to $[0, \infty)$. Different α represents different practical scenarios. When α is equal to zero, $c((a, b))$ is the unit cost. The cost of added edges is simply the number of added edges, i.e., the default case discussed in the literature. When α is greater than zero, we differentiate the cost function with respect to $SPL(a, b)$, and receive

$$\frac{dc((a, b))}{dSPL(a, b)} = \alpha[SPL(a, b)]^{\alpha-1} \quad (3)$$

that is the unit distance cost. When α is equal to one, the unit distance cost is equal to one. When α is smaller (or greater) than one, the derivative is smaller (or greater) than one, and the unit distance cost decreases (respectively, increases) with the increase of the length of the added edge. Therefore, the exponent α determines the unit distance cost. Note that, the cost of added edges is calculated in the original network. In Appendix C, we reduce the biconnectivity augmentation problem [26] to the network strengthening problem to prove that the latter is NP-hard. Therefore, the way to solve this problem is to design network strengthening heuristics.

3 Methodology

3.1 Making a path biconnected

Strengthening network robustness against APTA is to reduce the impact of AP removal or eliminate APs in the network so that APTA can not remove any node. APs exist in the path structure whose removal separates the path. Meanwhile, the path is an elementary component of a network. As a prerequisite, we first propose the Alg. 1 to solve the problem of making a path biconnected with the minimum cost.

When we remove all current APs from the network, there are some small components that do not contain these APs. We call them *leaf-components*. We choose two nodes

Algorithm 1 Making a path biconnected

Input: A path a_1-a_2-...-$a_{\widetilde{N_1}}$ from the network G, where a_1 and $a_{\widetilde{N_1}}$ are non-APs, a set of real-APs $S = \{a_i | i \in \Gamma \subseteq \{2, ..., \widetilde{N_1} - 1\}\}$ in this path and a cost function c of adding edges.

Output: A set of edges E_1 biconnecting the path using the minimum cost.

1: Normalize the path by finding the shortest sub-path containing all real-APs. Renumber the path as n_1-n_2-...-n_{N_1-1}-n_{N_1}, where $n_1 = a_{min(\Gamma)-1}$ and $n_{N_1} = a_{max(\Gamma)+1}$ are non-APs, and $n_2 = a_{min(\Gamma)}$ and $n_{N_1-1} = a_{max(\Gamma)}$ are real-APs. Renumber the AP set S as S_1;
2: Find all roots, $\{(n_1, n_t) \mid t \in [3, N_1], n_{t-1} \in S_1\}$;
3: **function** CHILDREN((n_u, n_v))
4: **if** n_v is an AP **then**
5: **return** $\{(n_{v-1}, n_i) | n_{i-1} \in S_1, i - (v-1) \geq 2\}$.
6: **else**
7: Find the next uncovered AP n_w, where $\{n_x | v < x < w\}$ are not APs;
8: **return** $\{(n_{w-1}, n_i) | n_{i-1} \in S_1, i - (w-1) \geq 2\}$.
9: **end if**
10: **end function**
11: **function** DFS(root)
12: stack = [(root, [root])], $Solution = \emptyset$;
13: **while** stack $\neq \emptyset$ **do**
14: (edge, s) = stack.pop();
15: **if** edge[-1] = n_{N_1} **then**
16: Add s to $Solution$;
17: continue;
18: **end if**
19: **for** child in CHILDREN(edge) **do**
20: stack.append((child, s + [child]));
21: **end for**
22: **end while**
23: **return** $Solution$.
24: **end function**
25: Candidate_solution = $\emptyset$;
26: **for** each root (n_1, n_t) **do**
27: Add DFS((n_1, n_t)) to Candidate_solution;
28: **end for**
29: Calculate the cost of each solution in Candidate_solution using the cost function c and choose the one with the minimum cost as E_1;
30: **return** E_1.

a_1 and $a_{\widetilde{N_1}}$ randomly from two different leaf-components, respectively, and find the shortest path a_1-a_2-...-$a_{\widetilde{N_1}}$ in the original network with some *real-APs* $S = \{a_i | i \in \Gamma \subseteq \{2, ..., \widetilde{N_1} - 1\}\}$ whose removal disconnects the original network and separates their neighbors in this path. Nodes a_1 and $a_{\widetilde{N_1}}$ are non-APs in the original network because they come from the leaf-components, and other nodes can be APs or non-APs. We normalize this path by finding the shortest sub-path containing all these real-APs, that is $a_{min(\Gamma)-1}$-$a_{min(\Gamma)}$-...-$a_{max(\Gamma)}$-$a_{max(\Gamma)+1}$. We renumber this path as n_1-n_2-...-n_{N_1-1}-n_{N_1}, where $n_1 = a_{min(\Gamma)-1}$ and $n_{N_1} = a_{max(\Gamma)+1}$ are non-APs, $n_2 = a_{min(\Gamma)}$ and $n_{N_1-1} = a_{max(\Gamma)}$ are real-APs, and renumber the AP set S as S_1 (line 1). Biconnecting the path is to add the optimal set of edges to cover all APs. The candidate solutions are obtained by depth-first search (DFS). First, we define the roots as the edges $\{(n_1, n_t) | t \in [3, N_1], n_{t-1} \in S_1\}$, connecting the endpoint n_1 with the other node n_t, where n_{t-1} is an AP (line 2). Each root covers at least one AP, n_{t-1}. Afterwards, we define the children of an edge (n_u, n_v), where $v - u \geq 2$ (lines 3-10). If n_v is an AP, the children are the edges $\{(n_{v-1}, n_i) | n_{i-1} \in S_1, i - (v-1) \geq 2\}$. If n_v is not an AP, we find the next uncovered AP n_w, where $\{n_x | v < x < w\}$ are not APs, and the children are the edges $\{(n_{w-1}, n_i) | n_{i-1} \in S_1, i - (w-1) \geq 2\}$. All children cover at least one AP, n_{i-1}. Based on this definition, we avoid unnecessary costs when covering APs. In DFS (lines 11-28), for each root, we store it and an initial candidate set in a stack format. At each iteration, we pop up an edge, add its children to the stack, and update the candidate set by adding a child. If the candidate set ends with the endpoint n_{N_1}, we get a candidate solution. The above process terminates until the stack is empty. We get all candidate solutions. Finally, we calculate the cost of each solution and add the corresponding edges with the minimum cost to biconnect the path (lines 29-30).

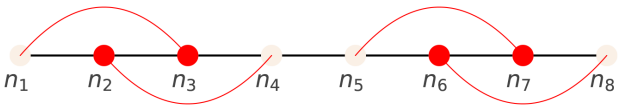


Fig. 1: A path with eight nodes. Red nodes are real-APs. The set of edges in red is a candidate solution obtained by DFS.

We illustrate our algorithm through a path in Fig. 1. Suppose that there are four real-APs in red. Roots are (n_1, n_3), (n_1, n_4), (n_1, n_7), (n_1, n_8) covering at least one AP, n_2. For instance, we start from the root (n_1, n_3) and find its children departing from the node n_2 because the next uncovered AP is n_3. The children are (n_2, n_4), (n_2, n_7), (n_2, n_8) that also cover at least one AP. We choose the child (n_2, n_4) and continue to find its children until the edge ends with the endpoint n_8. We get a candidate solution, $[(n_1, n_3), (n_2, n_4), (n_5, n_7), (n_6, n_8)]$, covering all APs.

The definition of children avoids unnecessary costs when covering APs, and we compare all possible solutions, so Alg. 1 provides a solution with the minimum cost. Starting from the root (n_1, n_t), there are at most $(N_1 - t) + (N_1 - t - 1) + \cdots + 2 + 1 = (N_1 - t + 1)(N_1 - t)/2$ children. So the complexity of DFS is $O([(N_1 - t + 1)(N_1 - t)/2 + 1]^2)$. The total complexity of Alg. 1 is $O(\sum_{t=3}^{N_1}[(N_1 - t + 1)(N_1 - t)/2 + 1]^2) < O(N_1^5)$. To reduce the complexity, we give another effective algorithm based on the Approx algorithm in Appendix D and prove that it provides the same result as Alg. 1. The complexity is $O(N_1^2)$ [27].

3.2 Making an arbitrary network biconnected.

We then biconnect an arbitrary network G. The Alg. 2 is divided into two stages. In the first stage (lines 1-20), we remove all current APs from G to find the leaf-components.

Algorithm 2 Making an arbitrary network biconnected

Input: A network G and a cost function c of adding edges.
Output: A set of edges biconnecting G.

1: Collect all APs as a set S, and set $LC = \emptyset$, $L = \emptyset$;
2: **for** $v \in S$ **do**
3: Create a copy of G as G_1;
4: Decompose G_1 to some components by removing v;
5: Add leaf-components that do not contain these APs in S to LC;
6: **end for**
7: For each leaf-component in LC, choose a node randomly and add it to L;
8: $t = 1$;
9: **while** $t \leq T$ **do**
10: $E_1 = \emptyset$;
11: **while** number of unused nodes in L is not less than two **do**
12: Choose two unused nodes from L randomly, and find the shortest path p_1 between them in G and all real-APs as a set aps_1 in this path;
13: Eliminate aps_1 in p_1 using Alg. 1 and add the corresponding edges to E_1;
14: **end while**
15: Calculate $C/\Delta N_{ap}$ of E_1;
16: $t = t + 1$;
17: **end while**
18: Add edges in E_1 with the smallest $C/\Delta N_{ap}$ to G;
19: **if** G is biconnected **then**
20: **return** E_1.
21: **else**
22: **while** G is not biconnected **do**
23: Find left APs as a set S_1, and set G_2 as a copy of G;
24: Remove a random node $v_1 \in S_1$ from G_2 and get some components;
25: Choose two nodes randomly from two random components that are not APs;
26: Find the path p_2 between these two nodes in G and the real-APs as aps_2 in this path;
27: Eliminate aps_2 in p_2 using Alg. 1 and add the corresponding edges to G and E_1;
28: **end while**
29: **return** E_1.
30: **end if**

We then generate a node-set L by choosing a node randomly from each leaf-component. At each step, we choose two unused nodes from L randomly and find the shortest path between them in G and the real-APs in this path. We add edges to cover these APs using Alg. 1. The above process is repeated until the number of unused nodes in L is smaller than two. We get a set of edges E_1 and calculate its $C/\Delta N_{ap}$, where C is the total cost of edges in E_1, ΔN_{ap} is the number of APs reduced after adding these edges to G. Finally, we repeat the whole process T times and get T edge sets. As we want to eliminate more APs with less cost, we add edges in E_1 with the smallest $C/\Delta N_{ap}$ to G. Some APs may still exist in G. In the second stage (lines 21-30), we eliminate these left APs. At each step, we remove a random AP from

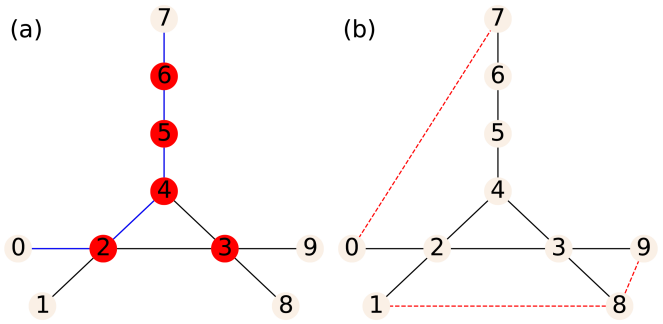


Fig. 2: (a) Red nodes are the APs. We first choose the path 0-2-4-5-6-7 in blue to add edges. (b) The added edges are shown with red dashed lines when α is equal to zero.

G and get some components. We choose two nodes that are not APs from two random components and find the shortest path between them. We biconnect this path using Alg. 1 and add the corresponding edges to E_1 and G. This process is repeated until no AP exists in the network.

We illustrate our algorithm through a small network depicted in Fig. 2. We set α to zero. Five APs are drawn with red. In the first stage, leaf-components are $\{0\}, \{1\}, \{7\}, \{8\}, \{9\}$ and the node-set L is $\{0, 1, 7, 8, 9\}$. We choose two nodes 0 and 7 and find the path 0-2-4-5-6-7 in blue in Fig. 2 (a). We add an edge $(0, 7)$ to biconnect this path by Alg. 1. The next two chosen nodes are 1 and 8, and we add an edge $(1, 8)$. Then there is only one AP left, node 3. In the second stage, we remove it from the network and get two components $\{0, 1, 2, 4, 5, 6, 7, 8\}$ and $\{9\}$. We find the path 8-3-9 between two randomly chosen nodes 8, 9 and add an edge $(8, 9)$. All added edges $\{(0, 7), (1, 8), (8, 9)\}$ are shown with red dashed lines in Fig. 2 (b).

In the first stage, suppose that we find N_2 leaf-components in the network and $N_2/2$ paths between pairs of nodes in the leaf-components. We repeat Alg. 1 T times on these paths, so the complexity of Alg. 2 is $O(N_1^2 N_2 T)$, where N_1 is the maximum number of nodes in these paths.

3.3 Decomposition-Coverage algorithm

We now introduce our *Decomposition-Coverage* (DC) algorithm in Alg. 3 to increase the robustness when the budget is smaller than that required by the previously obtained solution. In this case, the solution involves applying the Alg. 2 on the networks resulting from deliberately removing some APs; as a result, the average shortest path length within each of them will be shortened, leading to a reduction in the cost of the solution. Formally, suppose that the nodes removed by APTA are ordered as $\{n_{APTA,1}, \ldots, n_{APTA,M}\}$; an affordable solution can be found by removing the first m nodes, where $0 \leq m \leq M$, getting $M + 1$ GCCs as new networks and then using the Alg. 2 to find the added edges biconnecting these new networks. Finally, we add these edges to the original network G and calculate the corresponding cost and robustness.

We biconnect $M + 1$ new networks using the Alg. 2, so the total complexity of DC algorithm is $O(MN_1^2 N_2 T)$, where N_1 is the maximum number of nodes in the paths that

we find, N_2 is the maximum number of leaf-components in these networks, and T is the number of iterations in Alg. 2.

Algorithm 3 Decomposition-Coverage algorithm

Input: A network G and a cost function c of adding edges.
Output: The sets of cost and robustness: $Cost$, Rob.

```
Find the nodes removed by APTA. They are ordered as n_APTA = {n_APTA,1, ..., n_APTA,M} whose length is M;
m = 0, Cost = ∅, Rob = ∅;
while m ≤ M do
    Create a copy of G as G_1;
    Remove the first m nodes in n_APTA from G_1, find the GCC and take it as a new network G_new;
    Find the added edges biconnecting G_new using Alg. 2;
    Create a copy of G as G_2;
    Add these edges to G_2, calculate the cost and robustness, and then add them to Cost and Rob;
    m = m + 1;
end while
return Cost, Rob.
```

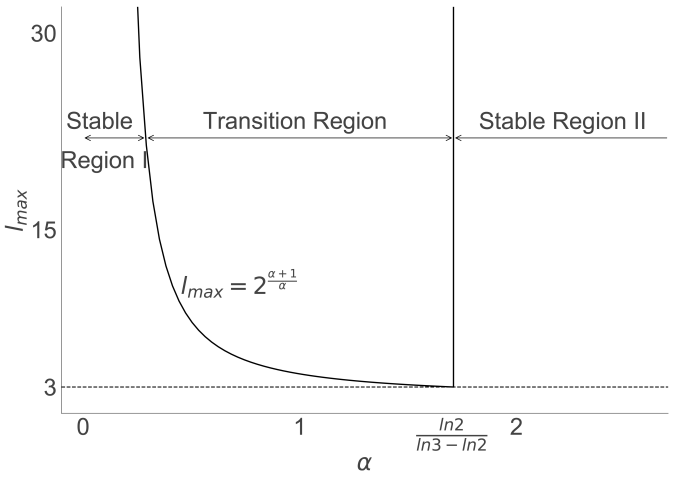


Fig. 3: Three cases of adding edges. The boundary curve is $l_{max} = 2^{\frac{\alpha+1}{\alpha}}$ and the boundary line is $\alpha = \frac{ln2}{ln3-ln2} \approx 1.7$. We connect the endpoints of all paths in the Stable Region I, add edges of length two in the Stable Region II, and add different edges in the Transition Region.

3.4 Theoretical results

We decompose the network and take the GCC as a sub-network to decrease the average shortest path length so that the budget is enough to biconnect the sub-network using the Alg. 2. We add edges to cover all APs in some normalized paths, denoted as a set SP. Therefore, the added edges are determined by the edges added in each path. These edges are influenced by the maximum length of these paths $l_{max} = max\{|p| \mid p \in SP\}$ and α in the cost function.

If the length l_1 of the normalized path $p = n_1\text{-}\ldots\text{-}n_{N_1}$ is equal to two, the path is n_1-n_2-n_3 with one AP n_2. It can be covered by the edge (n_1, n_3) of length two that does not depend on α. If the length l_1 of the normalized path p is not less than three, $l_1 \geq 3$, suppose that we cover all APs by adding one edge $\{e_1\}$ or two edges $\{e_2, e_3\}$. In this normalized path, n_2 and n_{N_1-1} are two real-APs, therefore, e_1 is the edge (n_1, n_{N_1}) with length l_1. e_2 and e_3 connect node n_1 and node n_{N_1} with another two nodes and their length is set to l_2 and l_3. We do not add edges between connected nodes, so l_2 and l_3 are not less than two, $l_2, l_3 \geq 2$. We need to cover all APs without redundant cost. Therefore, if these two nodes are non-APs, we have $l_1 \geq l_2 + l_3$. If at least one of these two nodes is an AP, we need to cover this AP by the other edge. So we have $l_1 = l_2 + l_3 - 1$. In summary, we have

$$l_1 \geq l_2 + l_3 - 1 \tag{4}$$

The costs of these two solutions are l_1^α and $l_2^\alpha + l_3^\alpha$. When α is equal to zero, we have $l_1^\alpha = l_2^\alpha = l_3^\alpha = 1$. So

$$l_1^\alpha < l_2^\alpha + l_3^\alpha \tag{5}$$

always holds. Increasing the number of added edges will increase the total cost. By induction, we conclude that the edge connecting two endpoints of each path $p \in SP$ is the optimal solution.

When α is greater than zero, if we still hope to connect the endpoints of all paths, we need to make Eq. (5) hold for all paths. Then l_1^α should be smaller than the minimum value of $l_2^\alpha + l_3^\alpha$, equal to $2^\alpha + 2^\alpha = 2^{\alpha+1}$ when $l_2 = l_3 = 2$. Therefore, we have $l_1^\alpha < 2^{\alpha+1}$, i.e., $l_1 < 2^{\frac{\alpha+1}{\alpha}}$. For all paths in SP, we should have $l_{max} < 2^{\frac{\alpha+1}{\alpha}}$. l_{max} is not less than three and equal to three when α is equal to $\frac{ln2}{ln3-ln2}$. Thus, the range of α is $(0, \frac{ln2}{ln3-ln2})$. It indicates that we connect the endpoints of all paths when $l_{max} < 2^{\frac{\alpha+1}{\alpha}}$ and $\alpha < \frac{ln2}{ln3-ln2}$ in the Stable Region I in Fig. 3.

When α is not less than $\frac{ln2}{ln3-ln2}$, we set a function as

$$f(l_2, l_3) = (l_2 + l_3 - 1)^\alpha - l_2^\alpha - l_3^\alpha \tag{6}$$

Take the partial derivatives of f as

$$\partial f_{l_2} = \alpha(l_2 + l_3 - 1)^{\alpha-1} - \alpha l_2^{\alpha-1} \tag{7}$$
$$\partial f_{l_3} = \alpha(l_2 + l_3 - 1)^{\alpha-1} - \alpha l_3^{\alpha-1} \tag{8}$$

Because $l_2 + l_3 - 1 > l_2, l_3$ and $\alpha - 1 > 0$, we have $\partial f_{l_2}, \partial f_{l_3} > 0$. Therefore, $f(l_2, l_3)$ is an increasing function whose minimum value is equal to $3^\alpha - 2^\alpha - 2^\alpha = 3^\alpha - 2^{\alpha+1}$ when $l_1 = 3$, $l_2 = l_3 = 2$. Let $3^\alpha - 2^{\alpha+1} \geq 0$, then $\alpha \geq \frac{ln2}{ln3-ln2}$. In this case, we have

$$(l_2 + l_3 - 1)^\alpha - l_2^\alpha - l_3^\alpha \geq 0 \tag{9}$$

Combining Eq. (4) and Eq. (9), we get

$$l_1^\alpha \geq l_2^\alpha + l_3^\alpha \tag{10}$$

Therefore, increasing the number of added edges will reduce the total cost. By induction, we should add edges with the minimum length, equal to two to cover APs when $\alpha \geq \frac{ln2}{ln3-ln2}$ in the Stable Region II in Fig. 3.

However, when $\alpha \in (0, \frac{ln2}{ln3-ln2})$ and $l_{max} \geq 2^{\frac{\alpha+1}{\alpha}}$, it is difficult to predict the added edges because they are determined by different l_1 and α. So, there is a transition region between two stable regions in Fig. 3.

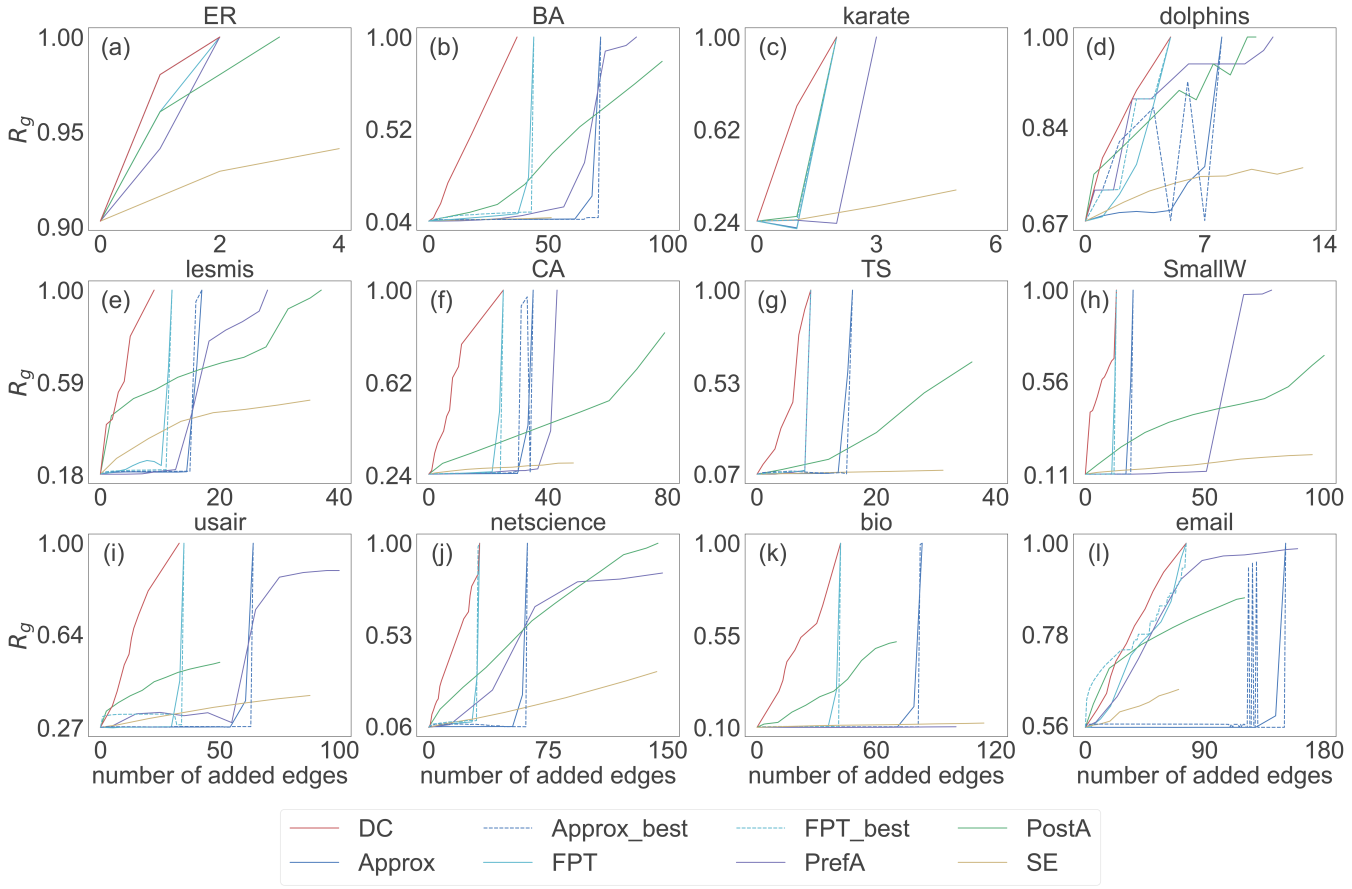


Fig. 4: Comparisons on two random networks and ten real-world networks with the unit cost. DC is the best-performing algorithm that increases the network robustness R_g most when adding the same number of edges.

4 Results

4.1 Evaluation with unit cost

We compare DC algorithm with state-of-the-art network strengthening methods (PrefA, PostA, and SE) and augmentation algorithms (Approx and FPT) on both random networks and real-world networks. We generate an ER network and a BA network randomly. The Tokyo Subway (TS) network and the Air China (CA) network are collected by ourselves, and other real-world networks can be downloaded from the UCI Network Data Repository [30]. These networks have extensively been studied in the literature and they cover a variety of network structures, as indicated by the number of APs, average degree, and diameter in TABLE 1. For these networks, we can get satisfactory results by setting $T = 1000$ in Alg. 2.

TABLE 1: Overview of two random networks, ten real-world networks and the network properties.

Network	$\|N\|$	$\|E\|$	$\|AP\|$	Avg. deg	Diam.
ER	100	244	3	4.88	6
BA	100	99	27	1.98	10
karate	34	78	1	4.59	5
dolphins	62	159	7	5.13	8
lesmis	77	254	8	6.60	5
CA (Air China)	107	469	7	8.77	4
TS (Tokyo Subway)	215	260	91	2.42	32
SmallW	233	994	5	8.53	4
usair	332	2126	27	12.80	6
netscience	379	914	57	4.82	17
bio (bio-CE-GT)	878	3181	46	7.24	10
email (email-univ)	1133	5451	132	9.62	8

We first set α to zero to recover the experimental setup used in the literature, where the total cost is measured by the number of added edges. Augmentation algorithms biconnect the network by adding all edges at once. To consider the process of adding edges, we provide two schemes: (i) We add one edge chosen from these edges randomly at each step and calculate the average results of one hundred independent experiments. The algorithms are denoted as Approx and FPT. (ii) We add the best edge that makes the robustness maximum when added at each step. We call the corresponding algorithms as Approx_best and FPT_best. Fig. 4 shows that DC (highlighted in red) is the best-performing algorithm that increases the robustness

most when adding the same number of edges. The worst method is SE. It does not perform any guided search and does not increase the number of edges, so it is hard to biconnect the network. The results of Approx, Approx_best, FPT, FPT_best, PrefA and PostA are between those of DC and SE and are rather unstable. If the budget is enough, they can biconnect the network. But if the budget is limited, they can not strengthen the robustness effectively. Approx and FPT suddenly increase R_g from a small value to one for some networks such as BA, lesmis, etc. The reason is that removing an AP may lead to the emergence of new APs, so more nodes will be removed from the network. But if we add edges to cover this AP, it will not be removed and no more APs will emerge. Therefore, covering the last AP may increase R_g a lot. Approx_best and FPT_best do not perform better than Approx and FPT algorithms, and the robustness could decrease because adding one edge does not necessarily increase the network robustness against APTA.

4.2 Evaluation with length-dependent cost

We further study the effect of making the cost length-dependent, i.e., $\alpha > 0$. We know from our theoretical result that when α is greater than or equal to two, DC algorithm connects the endpoints of all paths. For convenience, we only consider the case that α belongs to $[0, 2]$. We compare DC algorithm with other methods on two real-world networks. First, in the CA network, for which the increase of cruise time leads to a decrease of the unit distance cost, we set α to 0.0, 0.5, and 1.0. Second, in the TS network, for which the unit distance cost increases with the length for encountering more practical problems when building, we set α to 1.0, 1.5, and 2.0. Fig. 5 shows that DC algorithm strengthens the robustness significantly using less cost compared with other methods. It is worth mentioning that Approx biconnects the TS network using the same cost as DC algorithm when α is equal to 1.0, 1.5, and 2.0. But DC algorithm strengthens the robustness more when the cost is limited. More comparisons on random networks and more real-world networks are shown in Appendix E. The experimental results show that DC algorithm strengthens the robustness more effectively than other methods for different α. Approx_best and FPT_best still do not perform better than Approx and FPT algorithms, and their complexity is high because of choosing the best edge at each step. So we do not consider these two algorithms in the following experiments.

4.3 Properties of the strengthened networks

To understand the reasons underpinning the performance of DC algorithm, we calculate the length distributions of the added edges biconnecting the TS network in Fig. 6 (a)-(c). When α is equal to zero, DC algorithm adds nine edges connecting the endpoints of all paths that we choose. The length of added edges could be small or large, depending on the length of these paths. When α is equal to two, DC algorithm adds ninety-three edges of length two. When α is equal to one, DC algorithm adds eighteen edges, and the distribution is a compromise between the above two situations. The corresponding added edges are visualized

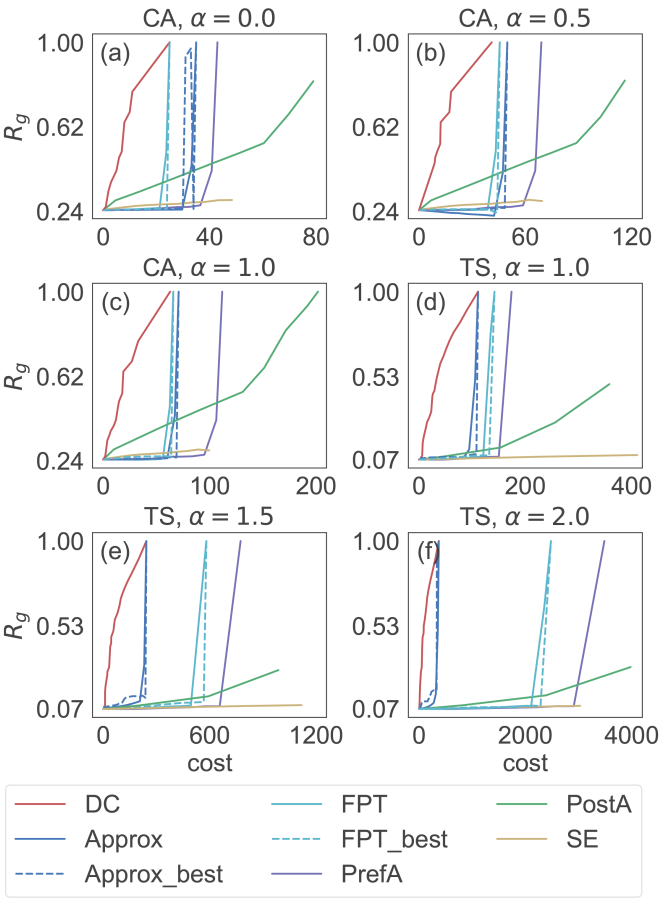


Fig. 5: Comparisons on the CA network and the TS network with length-dependent cost. α equals 0.0, 0.5, 1.0 in (a)-(c) and 1.0, 1.5, 2.0 in (d)-(f). DC algorithm performs better than other methods to strengthen the robustness for different α.

in red in Fig. 6 (a)-(c). Furthermore, we show that the number of added edges n increases but their average length l decreases when α increases in $[0, \frac{ln2}{ln3-ln2}]$ in Fig. 6 (d). This is, nicely, consistent with the theoretical result in Fig. 3. With the increase of α, adding long-distance edges costs more, so DC algorithm avoids connecting long-distance nodes but adds more edges to reduce the total cost. When α is greater than $\frac{ln2}{ln3-ln2}$, DC algorithm adds the same number (ninety-three) of edges of length two. Approx algorithm also adds edges with different lengths as α changes in Fig. 6 (e)-(g). When α is equal to two, it adds ninety-three edges of length two. That is the reason that it performs as well as DC algorithm to biconnect the TS network in Fig. 5 (f). The length of the edges added by FPT algorithm does not change significantly when α is different in Fig. 6 (i)-(k). When α is equal to zero, the number of added edges is nine and equal to the result of DC algorithm, as shown in Fig. 4 (g). However, with the increase of α, FPT algorithm still adds a few long-distance edges, leading to high costs. PrefA algorithm and PostA algorithm do not consider the length-dependent cost and add the same sets of edges for different α in Fig. 6 (h) and (l).

We show the degree distributions of the original TS network, the biconnected networks when α is equal to zero, one, and two in Fig. 7 (a), and the strengthened networks

TABLE 2: The change of ACC and ASPL. ACC increases but ASPL decreases as the network is strengthened.

Index	Network	ACC	ASPL
1	TS (original)	0.0242	10.33
2	TS, $R_g = 1$, $\alpha = 0$	0.0299	9.79
3	TS, $R_g = 1$, $\alpha = 1$	0.0407	9.15
4	TS, $R_g = 0.416$, $\alpha = 2$	0.0840	10.07
5	TS, $R_g = 0.732$, $\alpha = 2$	0.1861	9.58
6	TS, $R_g = 1$, $\alpha = 2$	0.3144	8.35

during the strengthening process when α is equal to two in Fig. 7 (b). We find that the node with degree one does not exist in the biconnected network because it would connect with an AP and needs to be strengthened. Besides, high degree nodes do not change significantly, implying that the edge addition does not change the network backbone. We also calculate the average clustering coefficient (ACC) and the average shortest path length (ASPL) in TABLE 2. Comparing indexes 1,2,3,6, we know that ACC increases but ASPL decreases as the network is strengthened. During the strengthening process, when α is equal to two, we can confirm this via the comparison of indexes 1,4,5,6.

4.4 Strengthening network robustness against general attack strategies

We further study the network robustness against the high-degree adaptive attack (HDA) and the high-betweenness adaptive attack (HBA), two alternatives extensively studied in the literature. At each step, they remove the largest degree or betweenness node. The robustness is simplified as $R_g = \frac{1}{N+1}\sum_{q=0}^{1} s(q)$ because these two attack strategies can eventually remove all nodes from the network.

The most robust network is the fully connected network whose $s(q)$ is reduced by $\frac{1}{N}$ when a node is removed. While DC algorithm is based on the identification of APs, the idea of strengthening the robustness is generic. Specifically, APs can be substituted by any list of nodes yielded by the chosen attack strategy, and the algorithm will return a strengthened network against this attack. If the removed node is not an AP, its removal does not dismantle the network, and thus, it does not need to be strengthened. But if the removed node is an AP, $s(q)$ will be reduced by more than $\frac{1}{N}$. So this node needs to be strengthened. Based on this idea, DC algorithm is modified accordingly. At each step, we first remove $m \in [0, N-3]$ nodes from the network based on the attack strategy. If the next removed node is an AP in the current network G_c, its removal splits the component into some small components. We choose the least important node (the smallest degree or betweenness node) from the largest component and the second largest component, respectively, and find the shortest path between them in G_c. We biconnect this path using Alg. 1, add the corresponding edges to the original network and calculate $\Delta Rg/C$, where ΔRg is the increment of robustness and C is the cost of these edges. To expand the search space, we also find the least important node in each weak core, connect it with the least important node in the critical giant component, like PostA algorithm, and calculate $\Delta Rg/C$. Finally, we add the edges with the largest $\Delta Rg/C$ to the original network so that we can strengthen the robustness more with less cost. This process is repeated until a certain number of edges are added. The experimental results on the TS network are shown in Fig. 8. Our algorithm is more effective to increase the robustness than other methods for different α. Approx and FPT can not further strengthen the robustness once the network is biconnected. PrefA and PostA still can not adjust the added edges according to the cost function.

4.5 Applying DC algorithm on the realistic problem

Afore-presented results are based on the network topology. Still, real-world networks include other information, such as the geographical positions of nodes. In the TS network, two stations may be geographically close, but their topological distance can be large if they do not belong to the same line. We tackle this case by estimating the distance between pairs of nodes through the Haversine Formula, and by then taking this distance as the cost of adding edges. Alg. 2 has then to be adapted in one place: In the first stage, when we find the shortest path between two unused nodes u, v chosen from L randomly, we also consider its alternative paths. We search the nodes x and y that are not APs in the bicomponents with the minimum geographical distance from nodes u and v, respectively. Then we find the shortest paths u-...-x and v-...-y. Comparing the edges added in the alternative paths with the edges added in the original path by Alg. 1, we choose the edges with less cost. Other methods can also include the geographical information. For Approx algorithm and FPT algorithm, we use the geographical distance as the cost function. For PostA algorithm, we add the edge that strengthens the robustness most using unit distance cost. For PrefA algorithm, we add edges between the smallest degree nodes with the minimum geographic distance. For SE algorithm, we still swap edges randomly and accept the change only if the robustness is increased. Fig. 9 shows that DC algorithm performs better on strengthening the network robustness than other methods when the cost is measured by the geographical distance. The strengthened and biconnected networks by DC algorithm and Approx algorithm are shown in Fig. 10 (a) and (b). The edges added by DC algorithm are reasonable from an operational point of view because they connect the short-distance stations. As the cost of the result of Approx algorithm is smaller than twice the optimal cost, this algorithm adds some redundant edges. Comparing these results with those in Fig. 6 (a)-(c), we find that the added edges are not realistic when the cost is based on SPL in graph theory. When α is small, the added edges connect long-distance nodes. When α is large, the number of added edges is large. Therefore, to get more practical solutions, we need to take the geographical distance into consideration.

5 Conclusion

Network robustness measures the capacity of a network to keep functioning when disrupted. In this paper, we generalized the robustness measurement against any attack strategy. Then we tackled the problem of strengthening the network robustness with the minimum cost and proved it to be NP-hard. Next, we designed a novel, targeted heuristic algorithm to strengthen the network robustness against

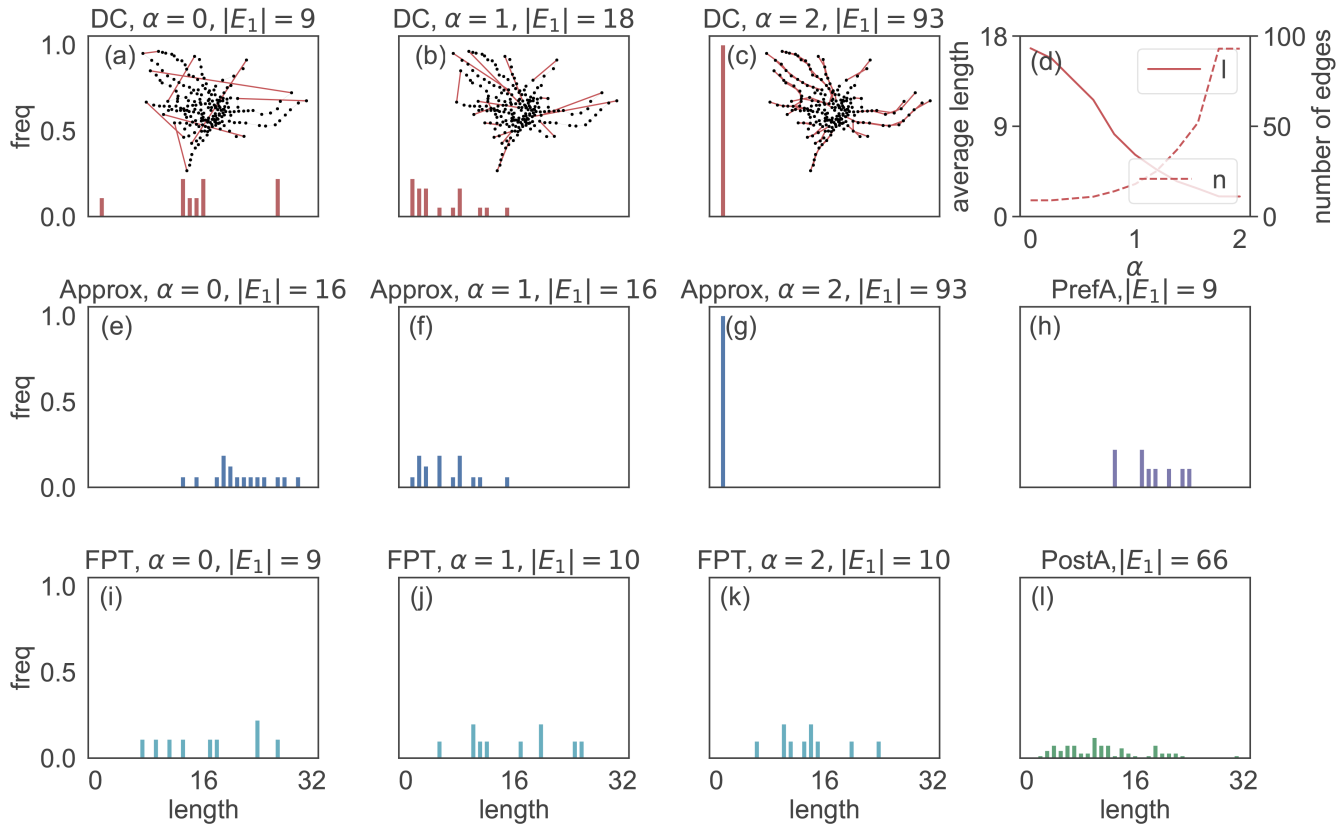


Fig. 6: Illustration of the reason that DC algorithm is more effective. (a)-(c) and (e)-(l) show the length distributions of the edges added by different methods. (a)-(c) also show the corresponding added edges in red. The transitions of number of edges n added by DC algorithm and their average length l are shown in (d). $|E_1| = n$ is the number of added edges.

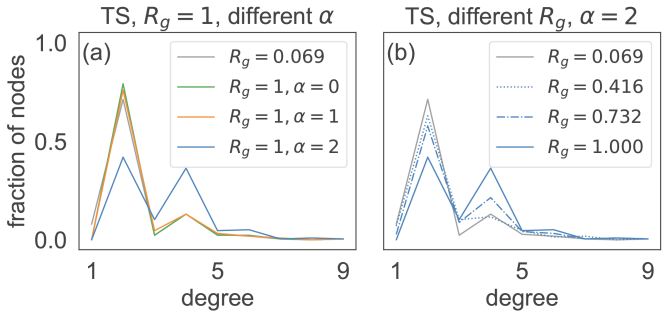


Fig. 7: The change of the degree distribution. (a) The node with degree one does not exist in the biconnected network for different α. (b) The number of nodes with degree one decreases with the increase of R_g.

articulation points-targeted attack. The network was decomposed into the right size by the attack strategy, and then we added edges to cover all APs in the specific paths in the network. For the paths, the results of our algorithm are optimal, and we provide the theoretical solutions to guide how to add edges according to different cost definitions. Experiments on both random and real-world networks show that our algorithm is better suited for the strengthening problem with unit cost as well as variable costs compared with the state-of-the-art methods. Compared to PostA, PrefA and SE algorithms, our algorithm can add edges pertinently to cover the articulation points to reduce the impact of their removal and take the cost of adding edges into consideration. Compared to FPT and Approx algorithms, our algorithm can strengthen the network robustness effectively using the limited budget. Against general attack strategies, we show that articulation point plays an important role in network strengthening. Our algorithm excels by revealing the essence of strengthening the robustness, that is, increasing the size of the giant connected component pertinently during the node removal process. Besides, our algorithm is flexible. With a minor modification, it is better suited for the realistic strengthening problem on real-world networks.

For future research, our algorithm could be adapted towards different attack strategies, such as localized attacks [31], [32]. In our study, the network robustness was measured by the size of the giant connected component; an alternative measure is the generalized k-core [33], for which different techniques might be required for effective strengthening. Moreover, there is a need for extending our algorithm to strengthen multi-layer or interdependent networks [34].

Acknowledgments

This study is supported by the National Natural Science Foundation of China (Grant No. 71731001).

References

[1] F. Radicchi, "Predicting percolation thresholds in networks," *Physical Review E*, vol. 91, no. 1, p. 010801, 2015.

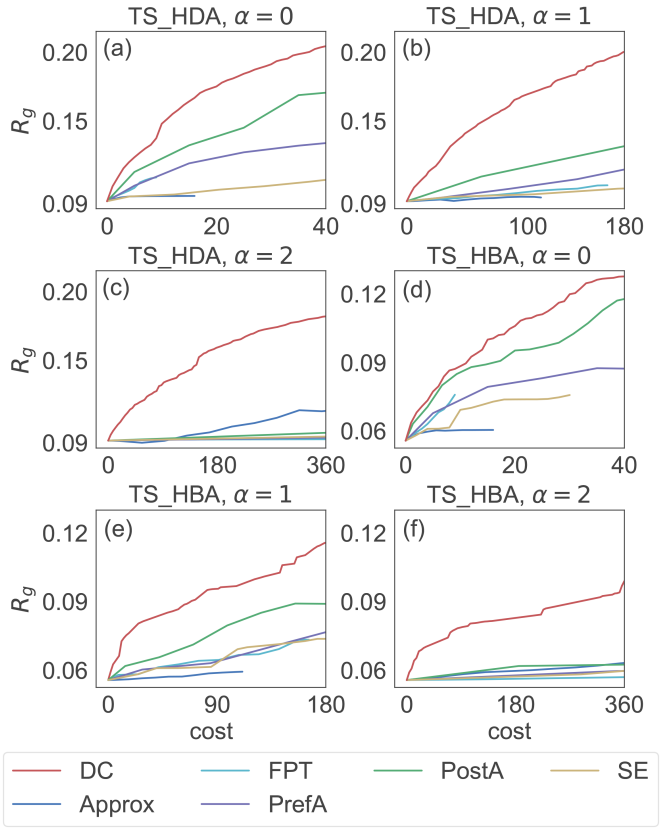


Fig. 8: Comparisons of strengthening robustness against (a)-(c) HDA and (d)-(f) HBA on the TS network. Our algorithm is more effective to strengthen the robustness than other methods.

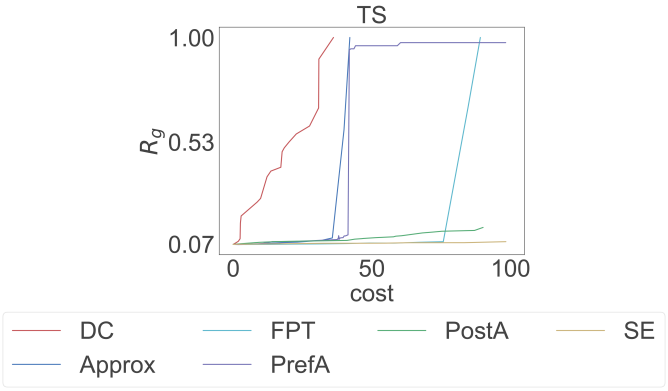


Fig. 9: Comparisons on the TS network when the cost is defined as the geographical distance. DC is the best-performing algorithm to strengthen the network robustness.

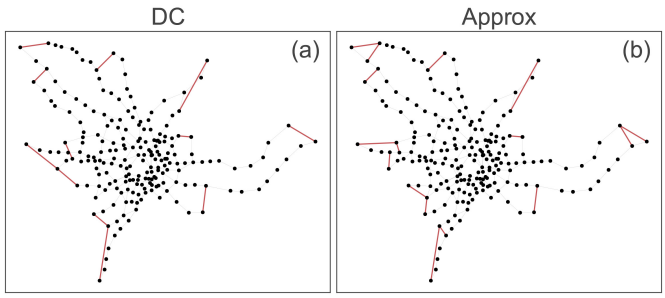


Fig. 10: Visualization of the strengthened TS networks. Red edges are added by (a) DC algorithm and (b) Approx algorithm when the cost is defined as the geographical distance.

[2] D. Lv, A. Eslami, and S. Cui, "Load-dependent cascading failures in finite-size erdös-rényi random networks," *IEEE Transactions on Network Science and Engineering*, vol. 4, no. 2, pp. 129–139, 2017.

[3] P. Crucitti, V. Latora, M. Marchiori, and A. Rapisarda, "Efficiency of scale-free networks: error and attack tolerance," *Physica A: Statistical Mechanics and its Applications*, vol. 320, pp. 622–642, 2003.

[4] D. S. Callaway, M. E. Newman, S. H. Strogatz, and D. J. Watts, "Network robustness and fragility: Percolation on random graphs," *Physical Review Letters*, vol. 85, no. 25, p. 5468, 2000.

[5] R. Cohen, K. Erez, D. ben Avraham, and S. Havlin, "Resilience of the internet to random breakdowns," *Physical Review Letters*, vol. 85, pp. 4626–4628, Nov 2000.

[6] B. Karrer, M. E. Newman, and L. Zdeborová, "Percolation on sparse networks," *Physical Review Letters*, vol. 113, no. 20, p. 208702, 2014.

[7] C. M. Schneider, A. A. Moreira, J. S. Andrade, S. Havlin, and H. J. Herrmann, "Mitigation of malicious attacks on networks," *Proceedings of the National Academy of Sciences*, vol. 108, no. 10, pp. 3838–3841, 2011.

[8] S. Iyer, T. Killingback, B. Sundaram, and Z. Wang, "Attack robustness and centrality of complex networks," *PloS One*, vol. 8, no. 4, p. e59613, 2013.

[9] S. Wandelt, X. Shi, and X. Sun, "Estimation and improvement of transportation network robustness by exploiting communities," *Reliability Engineering & System Safety*, vol. 206, p. 107307, 2021.

[10] S. Wandelt, X. Sun, M. Zanin, and S. Havlin, "QRE: quick robustness estimation for large complex networks," *Future Generation Computer Systems*, vol. 83, pp. 413–424, 2018.

[11] R.-R. Liu, C.-X. Jia, and Y.-C. Lai, "Asymmetry in interdependence makes a multilayer system more robust against cascading failures," *Physical Review E*, vol. 100, no. 5, p. 052306, 2019.

[12] J. Zhang and J. M. Moura, "Cascading edge failures: A dynamic network process," *IEEE Transactions on Network Science and Engineering*, vol. 5, no. 4, pp. 288–300, 2017.

[13] S. Wandelt, X. Sun, D. Feng, M. Zanin, and S. Havlin, "A comparative analysis of approaches to network-dismantling," *Scientific reports*, vol. 8, no. 1, pp. 1–15, 2018.

[14] S. Wandelt, X. Sun, and X. Cao, "Computationally efficient attack design for robustness analysis of air transportation networks," *Transportmetrica A: Transport Science*, vol. 11, no. 10, pp. 939–966, 2015.

[15] R. Cohen, S. Havlin, and D. Ben-Avraham, "Efficient immunization strategies for computer networks and populations," *Physical Review Letters*, vol. 91, no. 24, p. 247901, 2003.

[16] T. P. Peixoto and S. Bornholdt, "Evolution of robust network topologies: Emergence of central backbones," *Physical Review Letters*, vol. 109, no. 11, p. 118703, 2012.

[17] T. Verma, F. Russmann, N. A. Araújo, J. Nagler, and H. J. Herrmann, "Emergence of core–peripheries in networks," *Nature communications*, vol. 7, no. 1, pp. 1–7, 2016.

[18] Z.-X. Wu and P. Holme, "Onion structure and network robustness," *Physical Review E*, vol. 84, no. 2, p. 026106, 2011.

[19] Z. Jiang, M. Liang, and D. Guo, "Enhancing network performance by edge addition," *International Journal of Modern Physics C*, vol. 22, no. 11, pp. 1211–1226, 2011.

[20] A. Zeng and W. Liu, "Enhancing network robustness against malicious attacks," *Physical Review E*, vol. 85, no. 6, p. 066130, 2012.

[21] W. Li, Y. Li, Y. Tan, Y. Cao, C. Chen, Y. Cai, K. Y. Lee, and M. Pecht, "Maximizing network resilience against malicious attacks," *Scientific reports*, vol. 9, no. 1, pp. 1–9, 2019.

[22] A. Yehezkel and R. Cohen, "Degree-based attacks and defense strategies in complex networks," *Physical Review E*, vol. 86, no. 6, p. 066114, 2012.

[23] L. Tian, A. Bashan, D.-N. Shi, and Y.-Y. Liu, "Articulation points in complex networks," *Nature communications*, vol. 8, no. 1, pp. 1–9, 2017.

[24] R. Albert, H. Jeong, and A.-L. Barabási, "Error and attack tolerance of complex networks," *Nature*, vol. 406, no. 6794, pp. 378–382, 2000.

[25] F. Morone and H. A. Makse, "Influence maximization in complex networks through optimal percolation," *Nature*, vol. 524, no. 7563, pp. 65–68, 2015.
[26] K. P. Eswaran and R. E. Tarjan, "Augmentation problems," *SIAM Journal on Computing*, vol. 5, no. 4, pp. 653–665, 1976.
[27] G. N. Frederickson and J. Ja'Ja', "Approximation algorithms for several graph augmentation problems," *SIAM Journal on Computing*, vol. 10, no. 2, pp. 270–283, 1981.
[28] D. Marx and L. A. Végh, "Fixed-parameter algorithms for minimum-cost edge-connectivity augmentation," *ACM Transactions on Algorithms (TALG)*, vol. 11, no. 4, pp. 1–24, 2015.
[29] A. Patron, R. Cohen, D. Li, and S. Havlin, "Optimal cost for strengthening or destroying a given network," *Physical Review E*, vol. 95, no. 5, p. 052305, 2017.
[30] C. L. DuBois, "UCI network data repository," http://networkdata.ics.uci.edu, 2008.
[31] G. Como and F. Fagnani, "Robustness of large-scale stochastic matrices to localized perturbations," *IEEE Transactions on Network Science and Engineering*, vol. 2, no. 2, pp. 53–64, 2015.
[32] P. Basaras, G. Iosifidis, D. Katsaros, and L. Tassiulas, "Identifying influential spreaders in complex multilayer networks: A centrality perspective," *IEEE Transactions on Network Science and Engineering*, vol. 6, no. 1, pp. 31–45, 2017.
[33] Y. Shang, "Attack robustness and stability of generalized k-cores," *New Journal of Physics*, vol. 21, no. 9, p. 093013, sep 2019.
[34] Q. Cai, M. Pratama, S. Alam, C. Ma, and J. Liu, "Breakup of directed multipartite networks," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 947–960, 2019.

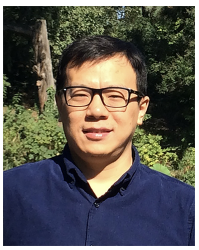
Qingnan Rong received the B.S. degree in Hua Luogeng Class in the department of Mathematics and System Sciences from Beihang University in 2017. Now he is pursuing his Ph.D. degree in the department of Electronic and Information Engineering of Beihang University. His research interests include complex network robustness and vulnerability, and network design.

Jun Zhang received the B.S., M.S. and Ph.D. degrees in communications and electronic systems from Beihang University, Beijing, China, in 1987, 1991, and 2001, respectively. He used to be a professor in Beihang University, and has served as the dean of the school of Electronic and Information Engineering, the vice president, and the secretary of the Party Committee of Beihang University. Now he is a professor in Beijing Institute of Technology, and also the president of Beijing Institute of Technology. His research interests are networked and collaborative air traffic management systems, covering signal processing, integrated and heterogeneous networks, and wireless communications. He has won the awards for science and technology in China many times, and he is a member of Chinese Academy of Engineering.

Xiaoqian Sun received the Ph.D. degree in aerospace engineering from Hamburg University of Technology in 2012. She is an Associate Professor with the School of Electronic and Information Engineering, Beihang University, and Beijing Key Laboratory for Network-Based Cooperative ATM, Beijing. Her research interests mainly include air transportation networks, multimodal transportation, and multi-criteria decision analysis.

Sebastian Wandelt received the Ph.D. degree in computer science from Hamburg University of Technology. He is a Professor with the School of Electronic and Information Engineering, Beihang University, and Beijing Key Laboratory for NetworkBased Cooperative ATM, Beijing. His research interests include transportation systems, scalable data management, and compressing/searching large collections of objects.

Massimiliano Zanin was born in Verona, Italy, in 1982. He received the Ph.D. degree in computer engineering from the Universidade Nova de Lisboa, Portugal, in 2014. He is currently a Researcher with the Institute for Cross-Disciplinary Physics and Complex Systems, Palma de Mallorca, Spain. His main research interests include complex networks and data science, from a theoretical perspective and through their application to several real-world problems. Throughout his career, he has published more than 80 articles in international journals and more than 50 contributions in conferences, reaching an H-index of 25. He has participated in several European competitive research projects. He is currently the PI of the ERC StG ARCTIC, on the analysis and modeling of delay propagation in air transport.



Liang Tian received the Ph.D. degree in physics from Nanjing University of Aeronautics and Astronautics and Hong Kong Baptist University in 2012. He is an Assistant Professor in the Department of Physics, Hong Kong Baptist University, Hong Kong, China. He performs theoretical and empirical research in the fields of Complex Systems, Statistical Physics, and System Biology by using various analytical, numerical, simulation, and data mining and machine learning techniques.