Socievole and Pizzuti
Applied Network Science (2023) 8:43
https://doi.org/10.1007/s41109-023-00569-0

Applied Network Science

REVIEW Open Access

Comparative evaluation of strategies for improving the robustness of complex networks

Annalisa Socievole[1*] and Clara Pizzuti[1]

*Correspondence: annalisa.socievole@icar.cnr.it

[1] Institute for High Performance Computing and Networking (ICAR), National Research Council of Italy (CNR), Via P. Bucci, 8-9C, 87036 Rende, CS, Italy

Abstract

Designing network systems able to sustain functionality after random failures or targeted attacks is a crucial aspect of networks. This paper investigates several strategies of link selection aiming at enhancing the robustness of a network by optimizing the effective graph resistance. In particular, we study the problem of optimizing this measure through two different strategies: the addition of a non-existing link to the network and the protection of an existing link whose removal would result in a severe network compromise. For each strategy, we exploit a genetic algorithm as optimization technique, and a computationally efficient technique based on the Moore–Penrose pseudoinverse matrix of the Laplacian of a graph for approximating the effective graph resistance. We compare these strategies to other state-of-the art methods over both real-world and synthetic networks finding that our proposals provide a higher speedup, especially on large networks, and results closer to those provided by the exhaustive search.

Keywords: Effective graph resistance, Robustness, optimisation, Moore–Penrose pseudoinverse, Graph spectrum, Genetic algorithms

Introduction

In the last 2 decades, the study of the organization of complex systems with concepts and methods of network science has been receiving considerable interest because of the ability of networks to well represent different kinds of real-world, natural, and technological systems (Barabási and Pósfai 2016). The Internet, the World Wide Web, transportation and power grid networks, biological, ecological, and social networks are just a few examples of networks deeply studied to understand their underlying arrangement (Girvan and Newman 2002; Lu et al. 2016; Battiston et al. 2020).

A critical aspect of networks is their ability to react to attacks, either targeted or due to random failures (Albert et al. 2000) because of the costs and impairments that damage to the structure could provoke. The research on approaches to enhance the robustness of networks is an active field that has been investigated in several application domains. Existing methods for improving robustness rely on some defense mechanisms that modify network topology, such as edge rewiring, edge addition, and edge protection

(Freitas et al. 2022; Wang et al. 2014). Each mechanism has a cost. Edge rewiring, for instance, that changes the connection between two nodes by using several strategies, has a lower cost than adding a new edge (Beygelzimer et al. 2005).

Several measures have been proposed for assessing the modifications to the network topology based on graph properties, such as the connectivity or the average betweenness of nodes or edges, and the spectrum of adjacency or Laplacian matrix (Freitas et al. 2022). However, the detection of the set of links or the single link that enhances the robustness of a network by optimizing the chosen criterion is a combinatorial search problem whose exact solution is computationally intractable. Thus greedy methods that obtain a suboptimal solution have been proposed.

In this paper, an investigation of several strategies of link selection for improving the robustness of a network G, by optimizing the well-known *effective graph resistance* measure, in the following denoted R_G, proposed by Ellens et al. (2011) is presented. This measure is based on the analogy between graphs and electrical circuits and has been shown to decrease when links are added to the graph and increase when links are removed (Ellens et al. 2011).

We study the problem of optimizing the effective graph resistance from two points of view: (1) adding the link that maximally decreases the effective graph resistance and (2) protecting the link whose removal would cause the maximum network damage, i.e. the link that maximally increases the effective graph resistance.

In Pizzuti and Socievole (2018, 2019), we proposed two methods based on genetic algorithms (Goldberg 1989), named *RobGA* and *RobLPGA* respectively, to find the best link in the network to either add or protect in order to optimize R_G. As outlined in Pizzuti and Socievole (2023), the main drawback of these methods is that the computation of the effective graph resistance must be repeated each time a new candidate solution is evaluated, thus making the approaches computationally inefficient. A modification to *RobGA* that improves the computation of R_G has been presented in Pizzuti and Socievole (2023) by introducing an incremental computation of R_G. The modified approach, named $RobGA\{L^+\}$, provides a good trade-off between the slightly increased error of the effective graph resistance value obtained with the approximated approach and the simulation time.

In the following, we first recall the concepts introduced in Pizzuti and Socievole (2023) needed to perform the incremental computation of R_G when adding a link and the results obtained by $RobGA\{L^+\}$. Then we present the method based on the Moore–Penrose pseudoinverse of the Laplacian matrix for the incremental computation of R_G when a link is deleted and extend *RobLPGA* to $RobLPGA\{L^+\}$, that performs the incremental computation of the effective graph resistance in the case of edge removal.

A comparative analysis with four single-link addition/removal strategies on both real and synthetically generated networks shows that the evolutionary approaches outperform the other strategies, and that the incremental computation of the effective graph resistance provides a good balancing between the increased percentage error value introduced with the approximate computation and the lower running times.

The paper is organized as follows. In "Related work" section the main works in this research area are described. In "Effective graph resistance R_G" and "Incremental computation of R_G" sections, the concept of effective graph resistance and of

incremental computation of the pseudo-inverse of Laplacian are recalled, respectively. In "Optimising the effective graph resistance through link addition or link protection" section the problems we tackle in the paper are defined. In "RobGA{L^+}" and "RobLPGA{L^+}" sections, the *RobGA* and *RobLPGA* methods are briefly described, and the efficient computation of R_G is introduced. In "Experimental setup" section, the real-world and synthetic networks used in the experimentation and the strategies adopted for the comparison are described. In "Results" section the results of the comparative analysis are reported. Finally, "Conclusion and further works" section concludes the paper and discusses the future directions.

Related work

The problem of robustness and how to quantify this measure in complex interconnected systems are extensively discussed by Freitas et al. (2022). In this survey, classical and more recent graph robustness measures are reviewed along with the types of attacks that can affect network robutness, and the defense techniques required to mitigate such attacks. Other works surveying graph measures for network robustness can be found in Oehlers and Fabian (2021) and Ellens and Kooij (2013). Since in this work we are interested in a particular robustness measure, the effective graph resistance, which will be recalled in the next section, in the following subsections we focus on the state of the art in (1) robustness management through link perturbations and (2) evolutionary methods improving robustness.

Robustness management through link perturbations

The improvement of network robustness is usually achieved by inducing a *perturbation* on its topology. A perturbation is an event occurring on the network which can be decomposed in a temporal sequence of elementary changes affecting its topology (Mieghem et al. 2010). An *elementary change* is any change occurring at time t that alters the network in terms of the corresponding graph matrix, such as the adjacency or the Laplacian matrix (Van Mieghem 2011). Elementary changes includes: (1) node addition, (2) node removal, (3) link addition, (4) link removal, (5) link rewiring (i.e. changing one of the two end nodes of a link with another node),(6) node weight change and (7) link weight change.

Through targeted perturbations, the robustness of a network can be enhanced or preserved. Focusing on link perturbations and effective graph resistance as indicator of robustness, Wang et al. (2014) demonstrate both experimentally and theoretically that network robustness can be improved by: (1) adding a new link that minimizes the effective graph resistance and (2) protecting a link (i.e. labelling this link as one of the most vulnerable) whose removal would maximize the effective graph resistance. Four methods that select a particular link to add or remove are evaluated on different types of networks, both real and modelled, showing the measurable consequences that the topology changes have on network robustness.

The strategies that add a link for mitigating degree-based targeted attacks appear promising also on interdependent networks and multilayer networks as shown in the work by Kazawa and Tsugawa (2020). Here, the authors analyze the performance of different link-addition strategies for single-layer networks like low degree (LD) and

random addition (RA), for interdependent networks (i.e., RIDD, random inter degree–degree difference and LIDD, low inter degree–degree difference), and extensions of the aforementioned strategies referred to as low-degree IDD (LD_IDD), low-degree-product IDD (LDP_IDD), and low-degree-sum IDD (LDS_IDD). The results suggest that methods for interdependent networks are also suitable for multiplex networks.

Schneider et al. (2011) use iterative ad-hoc rewirings that randomly choose couple of links and make link swaps only if robustness increases. This ensure a good robustness while preserving the number of links. In addition, focusing on the design of robust scale-free networks, an onion-like structure, where at the core there are high-degree nodes and at the adjacent layers there are nodes with decreasing degrees, is proposed.

Buesser et al. (2011) use the simulated annealing optimization heuristic on scale-free networks subject to malicious attacks to hub nodes (i.e. highly connected nodes). The network is properly rewired through the elimination of some existing links and the addition of new links by preserving the degree distribution and node connectivity. As robustness measure, the R value defined by Herrmann et al. (2011) is optimized.

Carchiolo et al. (2019) add a small number of new connections for enhancing the robustness of scale-free networks. Differently from most of the works in this direction that usually target hub nodes, the new links are added between nodes with a secondary role with respect to the most central ones. The aim is to set up *long-range* connections as backup paths in case of hub failures.

Louzada et al. (2013), present a rewiring method based on a small number of perturbations, which is suitable for real-time actions under budget constrains. The method is based on the evolution of the network largest component during a sequence of targeted attacks. Differently from random rewirings (Schneider et al. 2011), this smart strategy drives the formation of a *modular onion-like structure* characterized by layers of nodes grouped by degree.

Evolutionary methods improving robustness

Evolutionary computation is a type of optimization technique inspired by biological evolution (Bäck et al. 1997), successfully used for the resolution of many challenging real world complex optimization problems, including robustness optimization. Evolutionary methods initialize a population of individuals/solutions, and evolve it by applying the genetic operators (mainly *crossover*, *mutation*, *selection*) to improve the value of a *fitness function,* which is the function to optimize, while exploring the solution space. In principle, evolutionary methods are highly flexible since they can be applied to any kind of optimization problem.

Zhou and Liu (2014) propose a memetic algorithm for enhancing the robustness of scale-free networks through the optimization of the R value (Herrmann et al. 2011). Similarly to Buesser et al. (2011), this work focuses on attacks to hub nodes by exploiting a proper link rewiring able to preserve the degree distribution of the network. The method applies a customized crossover operator performing both a global and a local search. In such a way, the algorithm is able to search the network structure which optimizes the robustness.

In another work, Wang and Liu (2017) study how to improve the robustness of Erdős–Rényi networks and scale-free networks subject to attacks to links causing cascading

failures. First, a new robustness measure, namely R_{ce}, based on the R value (Herrmann et al. 2011) and adapted to cascading failures is proposed. Then, a memetic algorithm labeled as $MA - R_{ce}$ exploiting the same genetic operators used in Zhou and Liu (2014) and a local search operator employing simulated annealing as in Buesser et al. (2011) is proposed. Also $MA - R_{ce}$ preserves the degree distribution.

In Pizzuti and Socievole (2018), we proposed *RobGA*, a genetic algorithm that improves network robustness by adding a link that minimizes the effective graph resistance of the network R_G as robustness indicator. Similarly, in another work (Pizzuti and Socievole 2019), we focused on improving robustness through link protection by proposing *RobLPGA*, a genetic algorithm finding the link whose removal would maximally augment R_G. In our last work (Pizzuti and Socievole 2023), we focused on the computational effort necessary to improve robustness through *RobGA*, by proposing a fast computation of R_G through an approximation based on an incremental computation of the Moore–Penrose pseudoinverse matrix L^+ of the Laplacian L of the network graph. Termed as $RobGA\{L^+\}$, this method provides a good speedup with a low percentage error in the computation of the effective graph resistance.

Effective graph resistance R_G

The *effective graph resistance* R_G is a measure derived from the field of electric circuit analysis (Ellens et al. 2011) and based on the analogy that exists between graphs and resistive electrical circuits measure, which can be used to characterize the overall robustness of a graph G. Intuitively, R_G can be regarded as the overall difficulty of transport in a graph G. More specifically, given an undirected and connected graph G, an equivalent electrical network EEN can be composed by setting an edge e_{ij} with weight w_{ij} corresponding to an electrical resistance $\omega_{ij} = w_{ij}^{-1}$ Ohm. The *effective resistance* R_{ij}, is thus defined as the voltage developed between two nodes i and j when a unit current flows from i to j. A notable feature of effective resistance is that its square root $\sqrt{R_{ij}}$ is an Euclidean measure.

Since the current from a node i to a node j can spread over multiple paths, Klein and Randić (1993) defined R_G as a measure characterized by the "multiple-route distance diminishment", differently from the classical distance measures using a single path from i to j. In the context of a graph, if there exists more than one path between two nodes i and j, link failures can be easily managed by selecting alternative paths bypassing the unavailable edge. The smaller R_G, the higher the multiple routes and hence, the more robust the network is.

To compute R_G, we consider a network described by the undirected and connected graph without self-loops $G = (V, E)$, where V is the set of n nodes and E is the set of m links between node pairs.

The adjacency matrix A of the graph G is an $n \times n$ symmetric matrix where an element a_{ij} is w_{ij} or 0 depending on whether an edge between nodes i and j is present or not, and w_{ij} is the weight of the edge representing the affinity between nodes i and j. The *Laplacian* L of G is defined as the $n \times n$ symmetric matrix $L = \Delta - A$ where $\Delta = diag(d_i)$ represents the $n \times n$ diagonal matrix containing the nodes' degrees and $d_i = \sum_{j=1}^{n} a_{ij}$. Specifically, $L_{ij} = d_i$ if $i = j$, $L_{ij} = -1$ if $(i, j) \in E$, and $L_{ij} = 0$ otherwise.

For an undirected and connected graph, its Laplacian L is positive semi-definite, with eigenvalues that are all real and non-negative since the eigenvalues of the symmetric matrices Δ and A are real. In particular, the set of eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, named spectrum of L, has a unique smallest eigenvalue $\lambda_1 = 0$ with the rest of $n-1$ eigenvalues that are all positives ($0 = \lambda_1 \leq \lambda_2 \leq \ldots \lambda_n$). Fiedler coined the second lowest eigenvalue λ_2 as *algebraic connectivity*, for this reason the associated eigenvector is also referred to as the Fielder vector (Fiedler 1973).

The inverse matrix of L can not be computed due to the zero eigenvalue which makes L rank deficient with $rank(L) = n - 1 < n$. However, it is possible to obtain a matrix which can act as the inverse of L through the *Moore–Penrose pseudoinverse* matrix of L, denoted with L^+. L^+ shares with L the property of being positive semi-definite. In addition, the eigendecomposition of the pseudoinverse $L^+ = \Psi\Lambda^+\Psi'$ has the same set of orthogonal eigenvectors of L. The eigenvalues of L^+, contained in the diagonal matrix Λ^+, include $\lambda_1^+ = 0$ and the reciprocals of the positive eigenvalues of L, i.e. $\lambda_i^+ = \frac{1}{\lambda_i}$, $i = 2, \ldots, n$.

Ranjan et al. (2014) showed that R_{ij} can be computed through the elements of the Moore–Penrose pseudoinverse as:

$$R_{ij} = l_{ii}^+ + l_{jj}^+ - l_{ij}^+ - l_{ji}^+ \tag{1}$$

Having the effective resistances between pairs of nodes, the formal definition of *effective graph resistance* is the sum of the effective resistances between all pairs of vertices in the graph. Klein and Randić (1993) have proved that this measure satisfies the following spectral expression:

$$R_G = n \sum_{k=2}^{n} \frac{1}{\lambda_k} \tag{2}$$

Several studies consider the effective graph resistance a highly valuable robustness measure. In the work by Ghosh et al. (2008), for example, R_G is seen as a measure of the closeness between nodes indicating how well G is connected. The analogy between effective graph resistance and random walks is shown in Tizghadam and Leon-Garcia (2008) and Ellens et al. (2011): the pairwise effective resistance is proportional to the time duration of a random walk between the two nodes. Translated into effective graph resistance, R_G is proportional to the expected commute time averaged on all node pairs. Another interesting property is that R_G strictly decreases when edges are added or weights are increased (Ellens et al. 2011). Intuitively, the complete graph is more robust than a star, for example, due to the existence of more alternative paths.

Incremental computation of R_G

Despite the versatility offered by the Moore–Penrose pseudoinverse matrix of the Laplacian to practically compute the effective graph resistance, the computation of this pseudoinverse matrix is expensive, incurring an $O(n^3)$ computational time. In large networks, like online social networks composed of millions of nodes that change their connections over time, the dynamic evolution of the network topology obstacles the utility of this matrix. As the friendships change or user profiles are added or removed,

like in Facebook networks for example, the topology of such networks changes and this would require regular and clearly expensive re-computations of the matrix. Similarly, even if a network is relatively small but an algorithm would require regular updates of the Moore–Penrose pseudoinverse matrix, incremental updates in the computations of the matrix would be desirable considering that most of the topology changes happens locally.

In the work by Ranjan et al. (2014), a method for the incremental computation of the Moore–Penrose pseudoinverse of the Laplacian in undirected graphs is proposed. First, they show that L^+ can be efficiently computed through a *rank*(1) perturbation matrix, thus making L invertible, as

$$L^+ = \left(L + \frac{1}{n}J\right)^{-1} - \frac{1}{n}J \tag{3}$$

where $J \in R^{n\times n}$ is a matrix of ones. Even if Eq. (3) incurs also $O(n^3)$ computational time, on Erdős–Rényi graphs this L^+ computation is much faster than the one exploiting the MATLAB *pinv* standard command, which uses the singular value decomposition (SVD) to compute the pseudoinverse of matrices. Then, based on a divide-and-conquer approach, Ranjan et al. provide scalar forms for the computation of the elements of the pseudoinverse matrix through a two-stage and incremental process, both in case of topology modification through (a) edge addition and (b) edge deletion. In this last case, since the graph breaks up into disjoint components, the approach provides the submatrices pseudoinverse elements as well.

In the following, we recall the equations provided in Ranjan et al. (2014) for computing incrementally the Moore–Penrose pseudoinverse matrix. We will use these expressions for the incremental computation of R_G. This will help our algorithm proposal, both in case of link addition and link removal, to efficiently compute R_G when the topology is affected by an edge addition or removal without having to recompute L^+ from scratch.

Given the graph G, we indicate with $G + \{e_{ij}\}$ the modified graph when an edge $e(i, j)$ is added to G and with $G - \{e_{ij}\}$ the modified graph when a link is removed. The elements of the Moore–Penrose pseudoinverse matrix of the Laplacian of the modified graph can be computed incrementally as follows.

- Edge addition

Let l^+_{uv} and $l^{+(1)}_{uv}$ denote the general entries of the Moore–Penrose pseudoinverses L^+ of the Laplacian of G, and $L^{+(1)}$ of the Laplacian of the modified graph $G + \{e_{ij}\}$, respectively. According to Theorem 3 in Ranjan et al. (2014)

$$l^{+(1)}_{uv} = l^+_{uv} - \frac{\left(l^+_{ui} - l^+_{uj}\right)\left(l^+_{iv} - l^+_{jv}\right)}{\omega_{ij} + R_{ij}} \tag{4}$$

where ω_{ij} is the resistance of the edge between i and j (i.e. the inverse of its weight w_{ij}) and R_{ij} is their effective resistance in G. The general term of $L^{+(1)}$ is thus a linear combination of the elements of L^+, requiring $O(1)$ computations per element in $L^{+(1)}$ if L^+ is known.

- Edge removal

Let l_{uv}^{+} and $l_{uv}^{+(1)}$ denote the elements of the Moore–Penrose pseudoinverses L^{+} of the Laplacian of G, and $L^{+(1)}$ of the Laplacian of the updated graph $G - \{e_{ij}\}$ when en edge is removed from G, respectively. In this case, there are two cases to address: the removal of a *non-bridge edge* which does not split into components the graph, and the removal of a *bridge-edge* whose removal yields to graph disconnection into two partitions.

According to Theorem 4 in Ranjan et al. (2014), when deleting a non-bridge edge:

$$l_{uv}^{+(1)} = l_{uv}^{+} + \frac{\left(l_{ui}^{+} - l_{uj}^{+}\right)\left(l_{iv}^{+} - l_{jv}^{+}\right)}{\omega_{ij} - R_{ij}} \tag{5}$$

According to Theorem 5 in Ranjan et al. (2014), upon deleting a bridge-edge e_{ij}, we obtain two disjoint sub-graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with pseudoinverses of Laplacians $L^{+(1)}$ and $L^{+(2)}$, respectively. The orders of these two subgraphs of the original graph G are n_1 and n_2, respectively. The expressions of the elements of the two submatrices are:

$$l_{uv}^{+(1)} = l_{uv}^{+} - \frac{n_1 \sum_{z \in V_1(G_1)} \left(l_{uz}^{+} + l_{zv}^{+}\right) - \sum_{u \in V_1(G_1)} \sum_{v \in V_1(G_1)} l_{uv}^{+}}{n_1^2} \tag{6}$$

$$l_{xy}^{+(2)} = l_{xy}^{+} - \frac{n_2 \sum_{w \in V_2(G_2)} \left(l_{xw}^{+} + l_{wy}^{+}\right) - \sum_{x \in V_2(G_2)} \sum_{y \in V_2(G_2)} l_{xy}^{+}}{n_2^2} \tag{7}$$

Optimising the effective graph resistance through link addition or link protection

According to Theorem 2.7 in Ellens et al. (2011), "*the effective graph resistance strictly decreases when edges are added or weights are increased*". Conversely, this robustness measure increases upon link removal. Robustness could be thus optimised if the network graph is expanded with a new edge (i.e. minimizing R_G) or also reinforcing the link whose attack would maximize R_G. As outlined in Wang et al. (2014), these strategies are suitable when network efficiency needs to be increased with new infrastructural connections.

In this paper, we thus consider the problem of optimizing the effective graph resistance for improving the network robustness from two points of view: (1) adding a link and (2) protecting the link whose removal would cause the maximum network damage. Given a graph G with E_c the set of m_c new links not included in E, we formally define the two distinct problems of enhancing robustness as follows.

Problem 1 (*link addition*). Find an edge $e_{ij} \in E_c$ such that

$$R_{G+\{e_{ij}\}} \leq R_{G+\{e_{kl}\}} \tag{8}$$

for any other possible new edge $e_{kl} \in E_c$.

Problem 2 (*link protection*). Find an edge $e_{ij} \in E$ such that

$$R_{G-\{e_{ij}\}} \geq R_{G-\{e_{kl}\}} \tag{9}$$

for any other existing edge $e_{kl} \in E$.

To solve the first optimization problem, in Pizzuti and Socievole (2018) we proposed a genetic algorithm, namely *RobGA*, a method able to optimize the effective graph resistance. In *RobGA*, each possible individual of a population P, that is a link, is represented through a vector of 2 elements, where each element represents the ID of the end node of the link. Through this simple representation and ad-hoc defined crossover and mutation genetics operators, we showed that *RobGA* is able to provide solutions that in most of the cases match the ones found by the exhaustive search both on real-world and synthetically generated networks. It is worth noting that exhaustive search checks all the solutions space to find which one offers the best effective graph resistance. Moreover, *RobGA* also showed a very good performance compared to a set of heuristics investigated in Wang et al. (2014). With an analogous methodology, in Pizzuti and Socievole (2019) we solved the problem of link protection through *RobLPGA*, a genetic algorithm looking for the link to protect for optimizing the effective graph resistance. In this case, we used the same individual representation and fitness function used in *RobGA*, but crossover and mutation operators were redefined in order to adapt them to existing links.

However, both methods require the computation of the effective graph resistance *every time* a possible solution, a new link to add or an existing link to protect, is evaluated. In other words, when the genetic algorithm tests an individual e_{ij}, the effective graph resistance needs to be recomputed on $G + \{e_{ij}\}$ or $G - \{e_{ij}\}$, for Problem 1 and Problem 2 respectively. This means recomputing the Laplacian of a graph at each generation T for each element e of the population P, requiring $T \times P$ times. Over these schemes, R_G is computed through Eq. 2 and hence, through the eigenvalues of the Laplacian of the modified graph, which has a complexity order of $O(n^3)$. The overall complexity of the methods is thus $O(T \times P \times n^3)$, i.e. $O(n^3)$. It is worth noting that executing the methods over large networks leads to a notable increase of the computational time especially when using large population sizes. In this case, it would be preferable avoiding recomputing R_G all over again.

To overcome this drawback, the incremental computation of R_G, detailed in the previous section, can be a viable alternative to improve the performance of the two methods. In the following subsections, we describe how we solve the two aforementioned robustness optimization problems by using a more computationally efficient implementation of the effective graph resistance, by leveraging on the approach proposed by Ranjan et al. (2014) and the previous schemes *RobGA* and *RobLPGA*. We call these two approaches *RobGA*$\{L^+\}$ and *RobLPGA*$\{L^+\}$.

RobGA$\{L^+\}$

RobGA$\{L^+\}$ is an improved version of *RobGA*, initially proposed in Pizzuti and Socievole (2023). Like *RobGA*, this genetic algorithm creates a population of P individuals where

The $RobGA\{L^+\}$ **Method:**
Input: A graph $G = (V, E)$ representing a network, maximum number of generations T, populations size P, crossover fraction cf, mutation rate mr
Output: A link $e = (i, j) \in E_c$

```
Initialize a population of random individuals choosing edges from the set of non-existing link E_c
compute the pseudoinverse L^+ of the Laplacian of G with Equation 3
while termination condition is not satisfied do
  for each individual e = (i, j) of the population,
    compute the pseudoinverse L^+(1) of the Laplacian of G + {e} using the incremental Equation 4
  evaluate the objective function R_{G+{e}}
  create a new population of individuals by applying the variation operators
end while
Return the individual e that added to G gives the lowest value of the effective graph resistance
```

Fig. 1 The pseudo-code of the *RobGA*{L^+} algorithm

each individual represents a non-existing edge. Each individual/chromosome that is an edge $e_{ij} \notin E$ is represented as a vector of 2 genes where each element contains the value i and j, respectively. The crossover operator, given two parents $e_1 = e_{i,j} = (i, j)$ and $e_2 = e_{kl} = (k, l)$, combines them to generate a child e_3 such that the corresponding edge obtained by rewiring the parent nodes does not exist. Finally, given an individual e_{ij}, the mutation operator disconnects i from j and connects it to one of its neighbors chosen at random.

Differently from *RobGA*, the method exploits Eq. 1 and the Moore–Penrose pseudoinverse for computing the effective resistances of the links and then sums them for obtaining R_G. Each effective resistance R_{ij} is computed in terms of the Moore–Penrose pseudoinverse matrix elements through Eq. 4. In this way, the general Laplacian element of the augmented graph, $l_{uv}^{+(1)}$, is computed using the corresponding Laplacian element previously computed over the graph without the edge (i, j). It is worth noting that when initially computing R_G on the input graph, we do not use the MATLAB *pinv* function but the approximate formula given by Eq. 3 to further speed up the computation.

In Fig. 1, the pseudo-code of the algorithm is presented. *RobGA*{L^+} receives in input a graph G, a maximum number of generations T, the populations size P and the genetic parameters crossover fraction cf and mutation rate mr. It initializes the population by randomly choosing P chromosomes that are non-existing links. Initially, the Laplacian L^+ of G is computed. The elements of this matrix will be later used in the incremental computation of the Laplacian. Then, for each of the T generation, for each link e of P, the pseudoinverse of the Laplacian of $G + \{e\}$ is computed and the fitness function (i.e. $R_{G+\{e\}}$ computed with Eq. 1) is evaluated. After this, the algorithm creates a new population of individuals by applying crossover and mutation. At the end, the individual with the lowest value of $R_{G+\{e\}}$ is returned.

RobLPGA{L^+}

With a similar logic, we solve Problem 2 by proposing an improvement of *RobLPGA* based on the incremental computation of R_G. We call this new method *RobLPGA*{L^+}. *RobLPGA*{L^+} is a genetic algorithm sharing with *RobLPGA* individuals representation, ad-hoc genetic operators and effective graph resistance as the fitness function to maximize. Even when removing a link, the computational complexity depends by the fitness

The $\boldsymbol{RobLPGA\{L^+\}}$ **Method:**
Input: A graph $G = (V, E)$ representing a network, maximum number of generations T, populations size P, crossover fraction cf, mutation rate mr
Output: A link $e = (i, j) \in E$

1 **Initialize** a population of random individuals choosing edges from E
2 **compute** the pseudoinverse L^+ of the Laplacian of G with Equation 3
3 **while** termination condition is not satisfied do
4 **for each** individual $e = (i, j)$ of the population with $d_i \geq 2$ and $d_j \geq 2$
5 **if** $e = (i, j)$ is a non bridge-edge
6 **compute** the pseudoinverse $L^{+(1)}$ of the Laplacian of $G - \{e\}$ using the incremental Equation 5
7 **else** // $e = (i, j)$ is a bridge-edge splitting the graph into $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$
8 **compute** the pseudoinverses $L^{+(1)}$ and $L^{+(2)}$
9 **take** as $G - \{e\}$ its subgraph with the highest network size
10 **evaluate** the objective function $R_{G-\{e\}}$
11 **create** a new population of individuals by applying the variation operators
12 **end while**
13 **Return** the individual e that removed from G gives the highest value of the effective graph resistance

Fig. 2 The pseudo-code of the *RobLPGA*{L^+} algorithm

computation. In the case of *RobLPGA*, it required the computation of the eigenvalues of the Laplacian matrix every time a link to remove was tested and for each generation, that is $O(T \times P \times n^3)$. In *RobLPGA*{L^+}, R_G is computed through Eqs. 5, 6 and 7, if the removed link is a non-bridge edge or a bridge-edge, respectively. In the first case, the elimination of a link does not disconnect the graph into components. In the second case, when a link removal splits the graph into components, the two components have two separate Moore–Penrose pseudoinverse matrices. In this case, for simplifying the search, we consider an infinite value for the effective graph resistance and hence a bridge-edge as the optimal solution. In Fig. 2, the various steps of the algorithm are reported. It is worth noting that from the set of the existing edges, the algorithm only considers the links where nodes *i* and *j* have a degree greater or equal to 2 (step 4). This ensures that the evaluated link does not contain leaf nodes and as a result, the network traffic can flow toward other nodes. Protecting a link whose removal would disconnect just one node from the entire network, even if the resulting effective graph resistance is high, does not significantly impact network robustness if we consider that the rest of the network with $n - 1$ nodes would still continue working.

Experimental setup

In this section, we describe the experimental setting used for the comparative evaluation of the robustness strategies. As simulation environment to implement and evaluate the strategies, we used MATLAB R2020a. In particular, for *RobGA*, *RobGA*{L^+}, *RobLPGA* and *RobLPGA*{L^+} we used the Genetic Algorithm solver implemented in the Global Optimization Toolbox by using as genetic parameters: $P = 100$, $T = 300$, $cf = 0.9$ and $mr = 0.2$. In the following subsections, we describe the datasets, the other robustness strategies in comparison, and the performance indexes used.

Datasets

We consider both real and synthetic networks. The topological characteristics of such networks, alias (ID), number of nodes (n), number of links (m), average degree (<*k*>),

Table 1 Real-world networks topological features

Network	ID	*n*	*m*	<*k*>	<*C*>	*D*
Bell South	BS	51	66	1.294	0.081	0.052
ASNET-AM	AA	65	77	1.184	0.063	0.037
ITC Deltacom	ITC	113	161	1.425	0.053	0.025
ION	ION	125	146	1.168	0.006	0.019
US Carrier	USC	158	189	1.196	0.002	0.015
Ego 3980	3980	44	138	3.136	0.227	0.072
Ego 686	686	168	1656	9.8572	0.266	0.059
Ego 3437	3437	532	4812	9.045	0.272	0.017
US Power Grid	USPG	4941	6594	2.669	0.103	5.403e−04

Table 2 Synthetically generated networks topological features. The number of links and the density values are averaged over 10 network samples

Network type	Network ID	*n*	*m*	<*k*>	<*C*>	*D*
Erdős–Rényi	ER_128	128	627.3	5.23	0.054	0.041
	ER_256	256	1423.2	11.117	0.064	0.043
	ER_512	512	3240.2	12.64	0.01	0.024
	ER_1024	1024	6310.2	12.222	0.004	0.011
Watts–Strogatz	WS_128	128	384	6	0.109	0.047
	WS_256	256	768	6	0.104	0.023
	WS_512	512	2560	10	0.036	0.019
	WS_1024	1024	5120	10	0.039	0.009
Bárabasi–Albert	BA_128	128	253.4	3.954	0.132	0.031
	BA_256	256	510.2	3.984	0.129	0.015
	BA_512	512	1529.7	5.996	0.017	0.011
	BA_1024	1024	3065.2	5.986	0.012	0.005

average clustering coefficient (<*C*>), and density (D) are shown in Tables 1 and 2. Real-world networks are as follows.

Internet backbones:[1] we selected 5 graphs from the Internet Topology Zoo repository, where each node represents a BGP (Border Gateway Protocol) router and the edges between them their physical connections. Such networks often experience network attacks such as traffic reroute or blackholing.

Facebook ego networks: these are three friendship networks of Facebook users, where the Facebook users are the nodes and two users are considered connected if they are friends. Typically, Facebook graphs are composed of several ego networks (i.e. the user with its one-hop friends connected to other egos through common friends). We only take into account the *largest connected component* (LLC) of each network and not the entire topology since there are isolated nodes/small components (i.e. 8 nodes in 3980, 2

[1] http://www.topology-zoo.org.

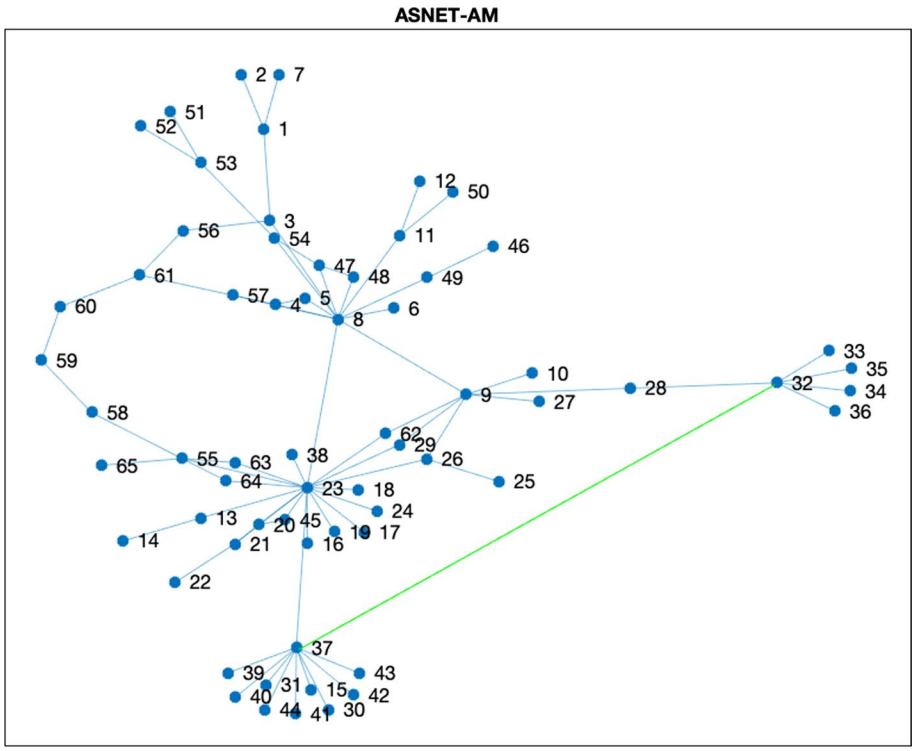


Fig. 3 *ASNET-AM* real-world network. The green link is the link added by the exhaustive search for having the optimal effective graph resistance

nodes in 686 and 2 nodes in 3437). These online social networks are vulnerable to fake news propagation and profile hacking.

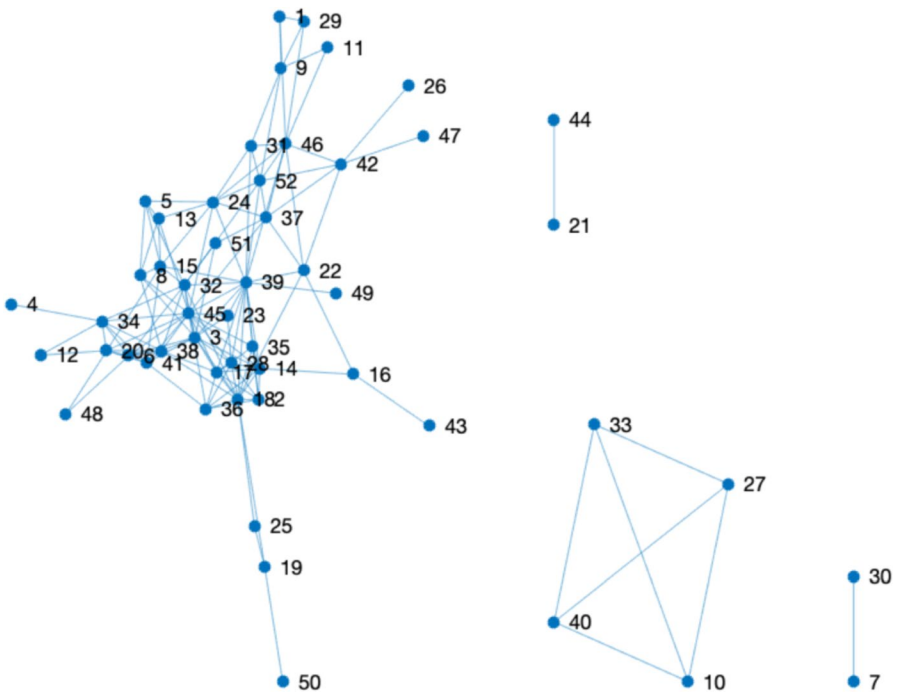
US power grid:[2] the nodes represent transformers, substations or generators of the Western States Power Grid while the links are the high-voltage transmission lines. Cascading failures and blackouts are very common on these complex networks.

Figure 3 shows one of the 5 Internet backbones, the ASNET-AM, a network composed by 65 nodes and 77 links. The best link to add, in this case, is the link highlighted in green ([32 37]). This link ensures a minimum effective graph resistance of $R_G + e(32, 37) = 5.38 \times 10^3$. Without the addition of this link, $R_G = 6.005 \times 10^3$. Figure 4 shows the Facebook Ego 3980: this network has originally 52 nodes (Fig. 4a) distributed over 4 components. We cut the components with few nodes and consider only the largest connected component depicted in Fig. 4b composed by 44 nodes and 138 links. The colored links are examples of links to protect whose removal would maximally increase the effective graph resistance.

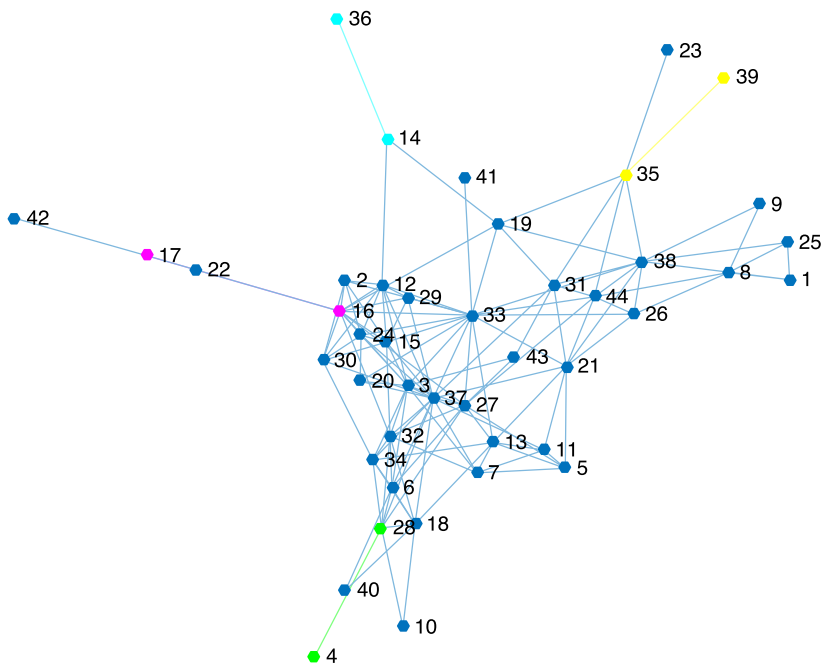
In this paper, we also synthetically generated the following graphs.

Erdős–Rényi random graph given n nodes, this graph is created by randomly assigning a link to two nodes with probability or *link density* p_c. The graph is connected if the density is greater than the critical threshold $p_c \approx \ln(n)/n$.

[2] http://konect.uni-koblenz.de/networks/opsahl-powergrid.



(a)



(b)

Fig. 4 Facebook Ego 3980 real world-network. **a** The whole topology and **b** the largest connected component. The colored links are those whose removal would maximally increase the effective graph resistance

Watts–Strogatz small-world graph this graph is created through a two-steps process. First, a ring lattice with n nodes and mean degree $2k$ is generated: at this step each node is connected to k neighbors on either side. Then, each edge is rewired at random with probability p. Here, we use the following generation parameters: for $n = 128$ and $n = 256$, $k = 6$ and $p = 0.5$, for $n = 512$ and $n = 1024$, and $k = 10$ and $p = 0.5$.

Bárabasi–Albert power law graph starting from n_0 nodes, this graph is generated by connecting at every time step t a new node j with with $n_k \leq n_0$ links to k neighbors. This is done with a probability $p = d_j/2m_t$, where d_j is the degree of node j and m_t are the number of edges at time t. For 128-nodes and 256-nodes, we use $n_0 = 5$ and $n_k = 2$, $n_0 = 10$ and $n_k = 3$ for the other networks.

All these graph models have features that can be found in real-world networks. Erdős–Rényi graphs, for example, can model ad-hoc networks, collaboration networks and peer-to-peer networks. Social networks, contact networks built upon Wi-Fi or Bluetooth encounters are often connected as small-world Watts–Strogatz graphs. The degree distribution in the World Wide Web, to provide another example, obeys approximately a power-law.

Strategies for selecting a link

For analyzing our strategies, we compare them to other state-of-the art methods. We consider 4 strategies of link addition or removal (Wang et al. 2014) that optimize the effective graph resistance and take into account the topological or the spectral features of the graph: S_1 *Semi-random*, S_2 *Degree Product*, S_3 *Fiedler vector*, S_4 *Effective resistance*. In addition to the above strategies, the *exhaustive search, that finds the optimal solution by checking all the possible links.* is also evaluated. This can be also considered the worst-case scenario since the optimal link search analyzes all the possible new links, in case of link addition, or all the existing links in case of link removal. The main drawback of this strategy, however, is its complexity order which dramatically increases with n, having $O(n^5)$. In this subsection, we briefly describe the strategies adapted to the link addition case. It is worth noting that these strategies can be applied to the link removal as well, just considering as edge set E and not E_c, and similarly, m instead of m_c as number of links. Let be $e(i, j)$ the link to select. The contestant strategies work as follows.

- S_1: node i has the lowest degree and node j is picked at random. The computational cost is $O(n^2 - n + m_c + 1)$, with $O(n(n-1))$ the cost for computing the node degrees, $O(m_c)$ the cost for searching the node with the minimum degree and $O(1)$ is required for selecting a random node.
- S_2: the product of the degrees $d_i d_j$ between two nodes is the minimum. The complexity for S_2 is $O(n^2 - n + 2m_c)$, with $O(n(n-1))$ for computing node degrees, $O(m_c)$ for their product and $O(m_c)$ for searching the minimum value of degree product.
- S_3: differently from the previous two strategies based on graph topology, this strategy analyzes the graph spectrum. Nodes i and j have the maximum difference $|y_i - y_j|$, where y_i and y_j are the i-th and j-th elements of the Fiedler vector y. The complexity of S_3 is $O(n^3 + 2m_c)$, where $O(n^3)$ is required for the Fiedler vector computation,

$O(m_c)$ for the difference $|y_i - y_j|$ of the m_c links, and $O(m_c)$ for searching the maximum of the difference.

- S_4: the nodes i and j have the maximum effective resistance R_{ij}, computed as in Eq. (1). The complexity of S_4 is $O(n^3 + 4m_c)$, where $O(n^3)$ is needed for computing the Moore–Penrose pseudoinverse L^+, $O(3m_c)$ for R_{ij} for the m_c links, and $O(m_c)$ for finding the highest effective resistance value.

Performance indexes

The strategies are compared by measuring the following performance indexes.

- *Percentage Error* the percentage relative error between strategy S_x and the exhaustive search (*) in terms of effective graph resistance measured on the graph G modified with the addition or the removal of a link.

$$\Delta R_G = \left| \frac{R_G^{S_x} - R_G^*}{R_G^*} \right| * 100 \tag{10}$$

- *Speedup* the ratio between the time required to run the strategy S_x and the strategy S_y.

$$Speedup = \frac{t_{sim}^{S_x}}{t_{sim}^{S_y}} \tag{11}$$

If the speedup has a value greater than 1, this means that S_y is faster than S_x. More precisely, if the speedup value is n, strategy S_y is characterized by an *n-fold* speedup.

Results

In this section, we report the findings of the comparative evaluation of the different strategies. In the first subsection, we first briefly recall the results of $RobGA\{L^+\}$ obtained in our previous work (Pizzuti and Socievole 2023) by extending the discussion highlighting the strong points of the proposed scheme. Then, we discuss new results of further simulations carried on the algorithm which investigate the relationship between population size and speedup, and population size and percentage error. In the second subsection, we describe the results obtained with $RobLPGA\{L^+\}$.

Link addition

Table 3 shows the results of the performance comparison between the 4 strategies, *RobGA* and $RobGA\{L^+\}$, over the real-world networks in terms of percentage error ΔR_G between a strategy and the exhaustive search. For *RobGA* and $RobGA\{L^+\}$, the results are averaged over 10 runs of the algorithm. We do not report the values of standard deviations since they are negligible. For $RobGA\{L^+\}$ we report two values between braces, first the minimum value of ΔR_G and then its average value.

Table 3 Comparison of percentage error ΔR_G between effective graph resistances in the augmented network for S_1, S_2, S_3, S_4 heuristics, *RobGA* and *RobGA*$\{L^+\}$ over real-world networks. For *RobGA*$\{L^+\}$, the minimum and the average $\Delta R_{G+\{e\}}$ values are reported

ID	$\Delta R^{S_1}_{G+\{e\}}$	$\Delta R^{S_2}_{G+\{e\}}$	$\Delta R^{S_3}_{G+\{e\}}$	$\Delta R^{S_4}_{G+\{e\}}$	$\Delta R^{RobGA}_{G+\{e\}}$	$\Delta R^{RobGA}_{G+\{e\}}\{L^+\}$
BS	8.918	4.493	0.76	0	0	{0, 0.401}
AA	7.472	6.97	1.914	2.565	0	{0, 1.152}
ITC	15.062	10.591	1.235	1.231	0.184	{0, 0.985}
ION	9.484	7.669	1.749	3.834	0	{0.255, 0.292}
USC	19.869	15.765	5.692	10.453	0.159	{0.396, 0.7}
3980	8.938	3.018	0.386	0	0	{0, 0.77}
686	7.26	0.847	0.435	0.826	0	{0.021, 0.282}
3437	2.186	1.567	0.635	0.522	0.077	{0, 0.419}

Table 4 Links added by the several strategies over real-world networks for the best ΔR_G value

ID	e^*	e^{S_1}	e^{S_2}	e^{S_3}	e^{S_4}	e^{RobGA}	$e^{RobGA\{L^+\}}$
BS	[43 47]	[32 25]	[14 24]	[3 6]	[3 6]	[43 47]	[43 47]
AA	[32 37]	[23 3]	[27 52]	[15 33]	[7 36]	[32 37]	[32 37]
ITC	[34 59]	[48 19]	[40 80]	[40 109]	[40 109]	[34 59]	[34 59]
ION	[4 55]	[30 20]	[7 72]	[5 55]	[54 103]	[4 55]	[104 55]
USC	[80 93]	[78 67]	[56 72]	[116 148]	[41 48]	[79 77]	[79 78]
3980	[36 42]	[37 5]	[4 39]	[23 42]	[36 42]	[36 42]	[36 42]
686	[26 62]	[140 145]	[62 89]	[62 164]	[88 153]	[26 62]	[7 62]
3437	[243 440]	[388 165]	[410 434]	[366 477]	[440 430]	[440 503]	[440 503]
USPG	–	[2847 3401]	[1853 3148]	[799 4463]	–	[3925 1776]	[4432 2747]

Table 5 Comparison of effective graph resistance in the original network (R_G) and in the augmented network resulting from the various strategies over USPG

ID	R_G	$R^*_{G+\{e\}}$	$R^{S_1}_{G+\{e\}}$	$R^{S_2}_{G+\{e\}}$	$R^{S_3}_{G+\{e\}}$	$R^{S_4}_{G+\{e\}}$	$R^{RobGA}_{G+\{e\}}$	$R^{RobGA\{L^+\}}_{G+\{e\}}$
USPG	6.377e+07	–	6.242e+07	6.314 e+07	6.173e+07	–	6.105e+07	6.107e+07

The percentage error of *RobGA* over the Internet backbones BS, AA, ITC, ION and USC is the lowest compared to the other strategies. For the networks BS, AA and ION, in particular, the genetic algorithm has an average ΔR_G value equal to 0 meaning that it is able to find the same optimal link provided by the exhaustive search (see Table 4). The approximation introduced by *RobGA*$\{L^+\}$ in the computation of R_G definitely results in a good performance: the average error is always less than any other strategy and in the best case, it matches the exhaustive search performance (UTC, AA, ITC).

A similar behavior has been found over Facebook networks: on network 3980, for example, the exhaustive search finds link [36 42], the same holds for S_4, *RobGA* and *RobGA*$\{L^+\}$. Also on networks 686 and 3437 the approximation of *RobGA*$\{L^+\}$ works well being the second best in the strategies ranking.

The values of R_G on the US Power Grid are reported in Table 5. Considering that the number of non-existing edges is 1.2×10^7, the strategies are compared just using the effective graph resistance value since the computational time required by the exhaustive search is prohibitive. Also, S_4 requires high computational time, for this reason, its R_G is not computed. *RobGA* is again the top performing method followed by $RobGA\{L^+\}$.

The results for the synthetic networks are shown in Tables 6, 7 and 8. For each network type, 10 network samples have been generated and the genetic algorithms have been executed 10 times. Table 6 compares the strategies in terms of average percentage error over the 128-nodes networks. The best performance is achieved by *RobGA*, while its approximation, $RobGA\{L^+\}$ has an error a bit higher if compared to the other strategies. However, despite the lower performance of $RobGA\{L^+\}$ over ER_128 and WS_128 networks in terms of error, its computational time is lower, much more lower when compared to S_4. Table 7 shows the average speedup as the number of nodes increases: $RobGA\{L^+\}$ is faster than its contestant strategies, especially over large networks. On the Erdős–Rényi networks with 1024 nodes, for example, $RobGA\{L^+\}$ is around ten times faster than *RobGA* and even 1097 times faster than S_4. As the network size increases, S_4 would not be a suitable choice due to its computational cost, $RobGA\{L^+\}$, on the contrary, would be the best compromise between the percentage error value of effective graph resistance and execution time.

Table 6 Comparison of average percentage error ΔR_G for $RobGA\{L^+\}$ over synthetic networks with 128 nodes

ID	$\Delta R^{S_1}_{G+\{e\}}$	$\Delta R^{S_2}_{G+\{e\}}$	$\Delta R^{S_3}_{G+\{e\}}$	$\Delta R^{S_4}_{G+\{e\}}$	$\Delta R^{RobGA}_{G+\{e\}}$	$\Delta R^{RobGA\{L^+\}}_{G+\{e\}}$
ER_128	1.028	0.029	0.044	0.093	0.004	0.103
WS_128	0.694	0.055	0.083	0.038	0.027	0.122
BA_128	0.607	0.219	0.064	0.012	0	0.0426

Table 7 Average speedup of $RobGA\{L^+\}$ over *RobGA* and S_4 on synthetic networks

Network type	Strategy	$n = 128$	$n = 256$	$n = 512$	$n = 1024$
Erdős–Rényi	*RobGA*	1.984	5.381	7.612	10.261
	S_4	1.985	101.993	401.451	1097.735
Watts–Strogatz	*RobGA*	1.805	3.53	5.076	9.708
	S_4	1.311	97.859	216.135	885.474
Bárabasi–Albert	*RobGA*	1.781	2.535	7.347	7.141
	S_4	2.933	59.4267	379.817	795.38

Table 8 Average speedup of $RobGA\{L^+\}$ over *RobGA* and average percentage error ΔR_G on synthetic networks with 128 nodes as the population size p varies

Network type	$p = 300$	$p = 500$
ER_128	{2.726, 0.085}	{3.015, 0.071}
WS_128	{2.747, 0.071}	{4.121, 0.058 }
BA_128	{2.154, 0.027}	{2.998, 0.011}

In a new experiment (Table 8), we analyzed more in deep the behavior of the speedup of $RobGA\{L^+\}$ over $RobGA$. In the experiment of Table 7, one can observe that the advantage of the approximation is more noticeable on the largest networks and with respect to S_4. For this reason, we compared $RobGA\{L^+\}$ and $RobGA$ measuring the speedup and the average percentage error on synthetic networks with 128 nodes as the population size varies. Generally, as the number of individuals of the genetic algorithm increases, the space of the solutions increases thus leading to better results. As expected, on the Erdős–Rényi networks, for example, with a population of 300 individuals, we found that the speedup of $RobGA\{L^+\}$ over $RobGA$ improves, achieving 2.726 while with a population of 100 the value was 1.984. Moreover, the error improves as well with a value of 0.085, in contrast to the previous value of 0.103. As the population size increases to 500, the speedup and the error improve accordingly. The advantage of increasing the population size is also visible on Watts–Strogatz and Bárabasi–Albert networks. We can thus conclude that the population size parameter has an important impact on the performance of the approximation and can help to improve $RobGA\{L^+\}$ behavior.

Link protection

Table 9 compares the various strategies in terms of percentage error ΔR_G when robustness is achieved through link protection. In this case, we analyze $RobLPGA$ and $RobLPGA\{L^+\}$ versus S_1, S_2, S_3, and S_4. For this experiment, we focused only on synthetic networks since the more sparse structure of the real-world networks with many leaf nodes leads the various strategies to often select links that disconnect the network into a main giant component and an isolated node. Theoretically, R_G is maximized since the network is disconnected, but in practice, protecting a link whose removal would result in isolating a node is less meaningful than situations in which the removed links

Table 9 Comparison of average percentage error ΔR_G for $RobLPGA\{L^+\}$ over synthetic networks with 128 nodes

ID	$\Delta R^{S_1}_{G-\{e\}}$	$\Delta R^{S_2}_{G-\{e\}}$	$\Delta R^{S_3}_{G-\{e\}}$	$\Delta R^{S_4}_{G-\{e\}}$	$\Delta R^{RobLPGA}_{G-\{e\}}$	$\Delta R^{RobLPGA\{L^+\}}_{G-\{e\}}$
ER_128	0.049	0.049	0.049	0	0	0.782
WS_128	0.67	0.507	0	0	0.003	1.452
BA_128	1.008	0.365	0.0013	0.004	0	1.953

Table 10 Average speedup of $RobLPGA\{L^+\}$ and S_4 over $RobLPGA$ on synthetic networks

Network type	Strategy	$n = 128$	$n = 256$	$n = 512$	$n = 1024$
Erdős–Rényi	*RobLPGA*	5.409	4.155	6.244	5.543
	S_4	1.532	4.974	10.517	19.174
Watts–Strogatz	*RobLPGA*	4.125	4.506	3.504	5.344
	S_4	0.934	3.096	5.596	11.323
Bárabasi–Albert	*RobLPGA*	3.473	3.4257	5.848	5.27
	S_4	0.698	1.58	5.409	11.924

Table 11 Average speedup of *RobLPGA*$\{L^+\}$ over *RobLPGA* and average percentage error ΔR_G on synthetic networks with 128 nodes as the population size p varies

Network type	$p = 300$	$p = 500$
ER_128	{4.592, 0.701}	{6.388, 0.682}
WS_128	{4.915, 1.377}	{7.943, 1.356}
BA_128	{4.943, 1.93}	{8.299, 1.892}

disconnect dense network areas. Note in Tables 1 and 2 that the considered real-world networks have overall a low average node degree around 1 and an average density lower than the real-world networks. ION, for example, has 125 nodes, average degree 1.168 and density 0.019, while 128-nodes Erdős–Rényi networks have average density 5.23 and density 0.041.

With the aforementioned network topologies, *RobLPGA* selected always a link disconnecting the network, resulting in infinite values of the effective graph resistance, as shown in Pizzuti and Socievole (2019). In this case, we can not compute an average percentage error of R_G. As such, here we consider only synthetic networks also for this further reason. Differently from the link addition case, the error introduced by the approximation is lower on the Erdős–Rényi networks but sensibly higher on the Bárabasi–Albert networks as it can be observed in Table 9. *RobLPGA* is always the best performing, with S_4 the second best. However, when looking at the speedup of Table 10, *RobLPGA*$\{L^+\}$ is faster than *RobLPGA* and much faster than S_4 as in the link addition case. Differently from this last case, the entity of the decrease in simulation time varies.

Compared to *RobLPGA*, *RobLPGA*$\{L^+\}$ is faster but this does not seem to scale with the increase in the number of nodes. This happens because the population size is always 100 even if n increases, and this lets the algorithm work always on a subset of 100 existing links. In the link addition case, the 100 links of the population were non-existing and, thus, extracted from a wider solution space. As the number of nodes increases, there is more variability in the population and hence, a higher probability of selecting a link giving a better effective graph resistance value.

Compared to S_4, *RobLPGA*$\{L^+\}$ is much faster and the speedup increases with the number of nodes because S_4 does not work only on a subset of 100 existing links and this requires more simulation time as n increases.

Table 11 makes a comparison between *RobLPGA* and *RobLPGA*$\{L^+\}$ in terms of speedup and ΔR_G as the population size varies from 100 to 300 and 500. The approximations benefit from the increment in population size providing faster solutions with lower errors. As the population size increases, both the two performance indexes improve.

Conclusion and further works

We introduced two robustness optimization methods based on the effective graph resistance measure of a graph. The two methods, namely *RobGA*$\{L^+\}$ and *RobLPGA*$\{L^+\}$, improve this robustness measure by either adding a link or protecting a link in a computationally efficient way through the Moore–Penrose pseudoinverse matrix of the Laplacian of the graph. We presented a comparative analysis of 6 different single-link

addition/removal strategies on both real-world and synthetically generated networks with the goal of evaluating the more efficient strategy.

The results we obtained are promising: indeed, the proposed methodology of optimizing the effective graph resistance through a genetic algorithm outperforms the other strategies and in some cases equals the exhaustive search. Moreover, computing the effective graph resistance through an approximation based on the Moore–Penrose matrix provides a higher speedup, especially on large networks, and results closer to those provided by the exhaustive search. Reducing computational costs by exploiting the approximate computation of the effective graph resistance value thus provides energy-efficient approaches while preserving good performance results. In the context of the very recent research activity of Green Artificial Intelligence that aims to design new methods that must take into account the saving of computation resources, our algorithms can be effectively extended to deal with dynamic networks that change their topology over time.

Further work will be devoted to the study of $RobLPGA\{L^+\}$ on network topologies where the candidate link to be removed is a bridge-edge. This aspect needs further investigation since an open question is how to compare the value of effective graph resistance of two bridge-edge solutions, since a bridge-edge solution splits the graph into two components providing a graph resistance value for each subgraph. Currently, we randomly choose one of these as the best solution but a more appropriate strategy would be maybe that one combining the values of the robustness of the subgraphs providing a cumulative measure.

Acknowledgements
We acknowledge the support of the PNRR project FAIR—Future AI Research (PE00000013), Spoke 9—Green-aware AI, under the NRRP MUR program funded by the NextGenerationEU.

Author contributions
All authors contributed to the paper, read and approved the manuscript.

Funding
Not applicable.

Availability of data and materials
The real-world networks are all available online. The synthetic networks can be generated through the reference models with the parameters specified in the text.

Declarations

Ethics approval and consent to participate
Not applicable.

Competing interests
Both authors declare that they have no competing interests.

Received: 27 February 2023 Accepted: 30 June 2023
Published online: 18 July 2023

References

Albert R, Jeong H, Barabási A-L (2000) Error and attack tolerance of complex networks. Nature 406:378–381
Bäck T, Fogel DB, Michalewicz Z (1997) Handbook of evolutionary computation. Release 97(1):1
Barabási A-L, Pósfai M (2016) Network science. Cambridge University Press, Cambridge
Battiston F, Cencetti G, Iacopini I, Latora V, Lucas M, Patania A, Young J, Petri G (2020) Networks beyond pairwise interactions: structure and dynamics. Phys Rep 874:1–92
Beygelzimer A, Grinstein G, Linsker R, Rish I (2005) Improving network robustness by edge modification. Physica A Stat Mech Appl 357(3–4):593–612

Buesser P, Daolio F, Tomassini M (2011) Optimizing the robustness of scale-free networks with simulated annealing. In: International conference on adaptive and natural computing algorithms. Springer, pp 167–176
Carchiolo V, Grassia M, Longheu A, Malgeri M, Mangioni G (2019) Network robustness improvement via long-range links. Comput Soc Netw 6:1–16
Ellens W, Kooij RE (2013) Graph measures and network robustness. arXiv preprint arXiv:1311.5064
Ellens W, Spieksm FM, Mieghem PV, Jamakovic A, Kooij RE (2011) Effective graph resistance. Linear Algebra Appl 435(10):2491–2506
Fiedler M (1973) Algebraic connectivity of graphs. Czechoslov Math J 23(2):298–305
Freitas S, Yang D, Kumar S, Tong H, Chau DH (2022) Graph vulnerability and robustness: a survey. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE.2022.3163672
Ghosh A, Boyd S, Saberi A (2008) Minimizing effective resistance of a graph. SIAM Rev 50(1):37–66
Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proc Natl Acad Sci USA 99:7821–7826
Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley Longman Publishing Co., Inc., Boston
Herrmann HJ, Schneider CM, Moreira AA, Andrade JS Jr, Havlin S (2011) Onion-like network topology enhances robustness against malicious attacks. J Stat Mech Theory Exp 2011(01):01027
Kazawa Y, Tsugawa S (2020) Effectiveness of link-addition strategies for improving the robustness of both multiplex and interdependent networks. Physica A Stat Mech Appl 545:123586
Klein DJ, Randić M (1993) Resistance distance. J Math Chem 12:81–95
Louzada VH, Daolio F, Herrmann HJ, Tomassini M (2013) Smart rewiring for network robustness. J Complex Netw 1(2):150–159
Lu L, Chen D, Ren X-L, Zhang Q-M, Zhang Y-C, Zhou T (2016) Vital nodes identification in complex networks. Phys Rep 650:1–63
Mieghem PV, Doerr C, Wang H, Hernandez JM, Hutchison D, Karaliopoulos M, Kooijt R (2010) A framework for computing topological network robustness. Delft University of Technology, Report 20101218
Oehlers M, Fabian B (2021) Graph metrics for network robustness—a survey. Mathematics 9(8):895
Pizzuti C, Socievole A (2018) A genetic algorithm for improving robustness of complex networks. In: Tsoukalas LH, Grégoire É, Alamaniotis M (eds) IEEE 30th international conference on tools with artificial intelligence, ICTAI 2018, 5–7 November 2018. Greece, Volos, pp 514–521
Pizzuti C, Socievole A (2019) A genetic algorithm for enhancing the robustness of complex networks through link protection. In: Complex networks and their applications VII: Volume 1 proceedings the 7th international conference on complex networks and their applications COMPLEX NETWORKS 2018 7. Springer, pp 807–819
Pizzuti C, Socievole A (2023) Incremental computation of effective graph resistance for improving robustness of complex networks: a comparative study. In: Complex networks and their applications XI: proceedings of the eleventh international conference on complex networks and their applications: COMPLEX NETWORKS 2022, vol 2. Springer, pp 419–431
Ranjan G, Zhang Z, Boley D (2014) Incremental computation of pseudo-inverse of Laplacian. In: Combinatorial optimization and applications. COCOA. Springer, Switzerland, pp 730–749
Schneider CM, Moreira AA, Andrade JS Jr, Havlin S, Herrmann HJ (2011) Mitigation of malicious attacks on networks. Proc Natl Acad Sci 108(10):3838–3841
Tizghadam A, Leon-Garcia A (2008) On robust traffic engineering in core networks. In: IEEE global telecommunications conference,GLOBECOM 2008, pp 1–6
Van Mieghem P (2011) Graph spectra for complex networks. Cambridge University Press, New York
Wang S, Liu J (2017) Enhancing the robustness of complex networks against edge-based-attack cascading failures. In: 2017 IEEE congress on evolutionary computation (CEC). IEEE, pp 23–28
Wang X, Pournaras E, Kooij RE, Van Mieghem P (2014) Improving robustness of complex networks via the effective graph resistance. Eur Phys J B 87(9):221
Zhou M, Liu J (2014) A memetic algorithm for enhancing the robustness of scale-free networks against malicious attacks. Physica A Stat Mech Appl 410:131–143

Publisher's Note