Seeking critical nodes in digraphs

Massimo Bernaschi, Alessandro Celestini, Marco Cianfriglia, Stefano Guarino, Giuseppe F Italiano, Enrico Mastrostefano, Lena Rebecca Zastrow

Journal of Computational Science 69 (2023) 102012

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Seeking critical nodes in digraphs

Massimo Bernaschi [a], Alessandro Celestini [a,*], Marco Cianfriglia [a,c], Stefano Guarino [a], Giuseppe F. Italiano [b], Enrico Mastrostefano [a], Lena Rebecca Zastrow [a]

[a] Institute for Applied Computing "Mauro Picone", National Research Council, Via dei Taurini 19, 00185, Rome, Italy
[b] Luiss University, Viale Pola 12, 00198, Rome, Italy
[c] Department of Mathematics and Physics, Roma Tre University, Largo San Leonardo Murialdo 1, 00146, Rome, Italy

ARTICLE INFO

Keywords:
Critical nodes
Networks connectivity
Centrality measures
Network analysis

ABSTRACT

The Critical Node Detection Problem (CNDP) consists in finding the set of nodes, defined *critical*, whose removal maximally degrades the graph. In this work we focus on finding the set of critical nodes whose removal minimizes the pairwise connectivity of a direct graph (digraph). Such problem has been proved to be NP-hard, thus we need efficient heuristics to detect critical nodes in real-world applications. We aim at understanding which is the best heuristic we can apply to identify critical nodes in practice, i.e., taking into account time constrains and real-world networks. We present an in-depth analysis of several heuristics we ran on both real-world and on synthetic graphs. We define and evaluate two different strategies for each heuristic: *standard* and *iterative*. Our main findings show that an algorithm recently proposed to solve the CNDP and that can be used as heuristic for the general case provides the best results in real-world graphs, and it is also the fastest. However, there are few exceptions that are thoroughly analyzed and discussed. We show that among the heuristics we analyzed, few of them cannot be applied to very large graphs, when the iterative strategy is used, due to their time complexity. Finally, we suggest possible directions to further improve the heuristic providing the best results.

1. Introduction

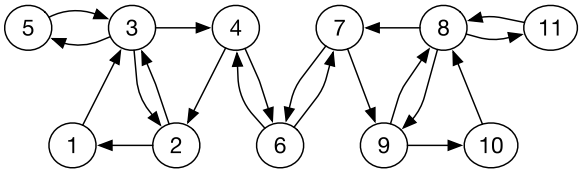
The problem of identifying the most *important* nodes in a graph is a widely studied subject and an active research area [1,2], with implications in several application fields, including, among others, economy [3], biology [4], viral marketing [5,6], misinformation [7,8] and epidemiology [9–11]. The importance of a node is not uniquely defined: it depends on the application of interest and on the specific task that we need to solve. In particular, we say that a node is *critical* when it plays a paramount role in keeping the network connected. The research of the set of nodes whose removal maximally degrades the graph connectivity is known in the literature as the Critical Node Detection Problem (CNDP) [12]. Different measures of connectivity may be considered, giving rise to different variants of the CNDP. In this paper, we focus on directed graphs and we consider the formulation of the CNDP that aims at identifying those nodes whose removal results in the minimization of the pairwise connectivity [13], i.e., of the number of vertex pairs that are strongly connected in the considered graph.

Formally, let G be a directed graph, and let $C_1, C_2, \ldots, C_N$ be its strongly connected components. The CNDP with parameter k is the problem of finding a set of nodes $S = \{n_1, n_2, \ldots, n_k\}$ of size k whose removal minimizes the connectivity $f(G)$ of G, with $f(G) = \sum_{i=1}^{N} \binom{|C_i|}{2}$. This formulation of the CNDP is known to be a NP-hard problem [13, 14]. Recently, Paudel et al. [15] proposed a linear-time algorithm, based on the framework of Georgiadis et al. [16], that optimally solves the CNDP for $k = 1$ in a directed graph. The design of the algorithm has been motivated by the study of Ventresca and Aleman [17] that addresses the same problem for undirected graphs. The algorithm proposed in [15] finds all *articulation points* in the graph – i.e., all nodes whose removal increases the number of strongly connected components – and it selects the best such nodes with respect to the objective function f. The algorithm by Paudel et al. cannot be directly used for the case $k > 1$ for two main reasons: the number of articulation points in a graph may be less than k and, in any case, the algorithm does not consider the combined effect of concurrently removing more than 1 vertex. These limitations, together with the lack of a publicly available and efficient implementation, limited the diffusion of the algorithm and fostered the use of greedy algorithms for the CNDP that rely on general-purpose measures of centrality [12].

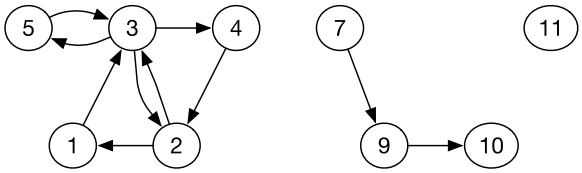
The underlying idea of any greedy heuristic for the CNDP is to rank the nodes of the graph based on a suitable centrality measure,

* Corresponding author.
E-mail addresses: massimo.bernaschi@cnr.it (M. Bernaschi), alessandro.celestini@cnr.it (A. Celestini), stefano.guarino@cnr.it (S. Guarino), gitaliano@luiss.it (G.F. Italiano), e.mastrostefano@iac.cnr.it (E. Mastrostefano).

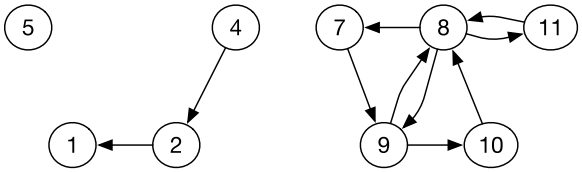
https://doi.org/10.1016/j.jocs.2023.102012
Received 12 August 2022; Received in revised form 15 February 2023; Accepted 21 March 2023
Available online 31 March 2023



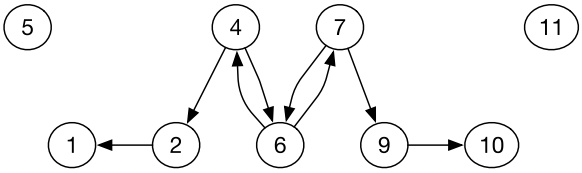
(a) A toy graph with 11 vertices, whose initial connectivity is 55.



(b) A sub-optimal solution to the CNDP for $k = 2$. The heuristic removed vertices 6 and 8. The residual connectivity is 10.



(c) A sub-optimal solution to the CNDP for $k = 2$. The heuristic removed vertices 6 and 3. The residual connectivity is 10.



(d) The optimal solution to the CNDP for $k = 2$. Vertices 3 and 8 have been removed. The residual connectivity is 3

Fig. 1. Suppose that we want to solve the CNDP for $k = 2$ for the toy graph with 11 vertices depicted in panel (a). Optimally solving the CNDP for $k = 1$ twice leads to removing node 6 at the first step and nodes 3 or 8 at the second step, thus obtaining the sub-optimal solutions depicted in panel (b) and (c). The optimal solution, shown in panel (d), instead consists in removing nodes 3 and 8, which, in this example, are the two vertices having the greatest degree centrality.

assuming that the removal of the most central nodes has the potential of disrupting the network. Given a metric X, we have two main options. On the one hand, we may run X just once, select and remove the k top-ranking nodes based on X — we refer to this heuristic as the *standard* heuristic based on X, also denoted X-S in the following. On the other hand, we may define the sequence of graphs $G = G_0, G_1, \ldots, G_k$, where G_{i+1} is obtained from G_i removing the node of G_i that has the greatest X score — this approach requires computing X on each of the k graphs $G_0, \ldots, G_{k-1}$ and we refer to it as the *iterative* heuristic based on X, also denoted X-I in the following. Of course, we may define an iterative heuristic for the general CNDP on top of the algorithm by Paudel et al.: the graph G_{i+1} in the sequence is obtained removing from G_i the node that optimally solves the CNDP with $k = 1$ for G_i. Since the algorithm by Paudel et al. is designed to identify critical nodes, this heuristic may be expected to provide near-optimal solutions. Unfortunately, the obtained solution is, in general, sub-optimal, and selecting the k nodes to remove based on some other centrality measure might indeed be a better choice, as shown with a toy example in Fig. 1. The main goal of our study is to evaluate how well this and other heuristics perform in solving the CNDP in practice, taking into account time constrains and real-world networks.

The main contributions provided by the present work are the following. First, we reviewed and implemented the algorithm proposed in [15], which we release as open source software in a repository[1] that includes all code and data used in this paper. Secondly, we present an in depth analysis of several CNDP heuristics on both synthetic and real-world graphs, that shows how the heuristic based on the algorithm by Paudel et al. [15] is, in most cases, the best and the fastest choice for identifying critical nodes in digraphs. The analysis allowed to identify a few open issues of the algorithm proposed in [15] and of its use as the base for an iterative heuristic. We suggest possible directions to overcome these drawbacks, especially in relation to the possibility that no articulation points are found at some stage of the process. Finally, we identify and thoroughly discuss in the paper a few scenarios in which the heuristics based on other centrality metrics, mostly betweenness, PageRank and degree, work better than or comparably to the one based on the algorithm by Paudel et al. In particular, the last two metrics provide a reasonable trade-off between computational time and effectiveness.

The rest of the paper is organized as follows: in Section 2 we describe which graphs and heuristics we selected for our study, and how the iterative and standard strategies work; in Section 3 we evaluate the goodness of each heuristic based on the ability of disconnecting the graph for an equal number of removed nodes; in Section 4 we evaluate the heuristics taking into account their running time, this factor can be critical in case of time constraints; finally in Section 5 we draw some conclusions and provide some directions for future works.

2. Methodology

To evaluate which is the best practical approach to solve the CNDP, we compared several heuristics against different graphs, both synthetic and originated from real-world networks. In particular, we analyzed a heuristic based on the algorithm proposed by Paudel et al. [15], that hereinafter we refer to as the Critical Node Heuristic (CNH), and a set of heuristics built on top of the following centrality measures: total degree (DG), undirected closeness (CL), PageRank [18] (PR) and betweenness (BC). The choice to consider the total degree – i.e. the sum of the out- and in-degree of a given vertex – and the undirected closeness – i.e. the closeness computed on the undirected version of the input graph, obtained ignoring edge direction – follows from preliminary evidence that these metrics provide similar or better results than, respectively, the out-/in-degree and the closeness from/to a given vertex.

The chosen metrics are inspired by different notions of centrality and are commonly used in the analysis of empirical networks. To some extent, they all tend to reward nodes having a clear impact on the connectivity of the graph. As a matter of fact, they select: (i) the nodes with the greatest number of neighbors; (ii) the nodes whose average distance to all other nodes is minimal; (iii) the nodes that are most likely visited by a random walk on the graph and; (iv) the nodes that appear the most in any shortest path of the graphs, respectively. Moreover, these metrics have different computational costs, a factor that must be taken into consideration when the CNDP must be addressed under time and/or computational constraints, or when the input graph must be quickly disconnected, with the number k of vertices to remove not being strictly bounded.

Formally, the CNH is defined as follows. We start with the input graph $G_0 = G$ and we iteratively run the algorithm by Paudel et al. [15] on graph G_i to identify a single node u_i, which we remove from G_i (together with all its incident edges) to obtain a new graph G_{i+1}. We stop when k nodes have been removed, with G_k thus being the output graph. At each iteration, three different outcomes are possible: (i) if the

[1] https://github.com/iac-cranic/seeking_critical_nodes_digraphs

Table 1
Time complexity of different heuristics. N is the number of vertices, M the number of edges. k is the number of iterations.

Heuristic	Complexity
Degree	$O(N+M)$
Closeness	$O(NM)$
PageRank	$O(kM)$
Betweenness [19]	$O(NM)$
CNH [15]	$O(N+M)$

algorithm returns a single critical node, we take u_i to be that node; (ii) if the algorithm returns multiple nodes that are equally critical, we select u_i looking first at the size of the Strongly Connected Component (SCC) to which it belongs and then at its degree (in both cases, the larger the better, with ties broken randomly); (iii) if the algorithm returns an empty list of nodes, meaning that the graph G_i has no articulation points, we select a node uniformly at random in G_i.

For each centrality measure, we considered two different strategies/modes to define a heuristic for the CNDP, that we call *standard* and *iterative*, respectively. In the *standard* strategy, centrality values are computed only once in the input graph G. Those values are then used to select the top k nodes that are considered critical, and thus removed from the graph. The standard heuristic for metric X is also denoted X-S in the following. In the *iterative* strategy, centrality values are computed each time we need to remove a new node from the graph. Similarly to the CNH, we start with the input graph $G_0 = G$ and we iteratively compute the selected metric on G_i to identify the single most central node u_i, which we remove from G_i to obtain the new graph G_{i+1} (again, ties are broken randomly). If we want to remove k nodes, we thus have to compute centrality values k times. The iterative heuristic for metric X is also denoted X-I in the following. As a matter of fact, the CNH only runs in iterative mode. Differently from the other centrality measures, in fact, the algorithm by Paudel et al. [15] only ranks the articulation points of the graph, and these nodes may be fewer than the desired k — possibly 0, as already discussed above.

It is apparent that the *standard* and *iterative* strategies have different computational costs. In the iterative mode, we apply k-times the algorithm to compute centrality values, whereas in the standard mode that is done only once in the original graph. Table 1 shows the time complexity of each algorithm we use in the paper. We study the iterative strategy to understand if the application of the metric to the updated graph can be beneficial for the solution of the problem and to which extent it can be worth to pay the extra running time cost. For very small graphs, in the following called *tiny* graphs, we also compute the k optimal critical nodes, i.e., the exact solution of CNDP. This algorithm, denoted brute force (BF) approach in the rest of the paper, is applied only to tiny graphs due to its time complexity. Finally, we use a *random* heuristic (RND) as baseline to compare against all heuristics. As the name suggests, the random heuristic randomly selects the set of k critical nodes to remove.

We ran each heuristics on a selection of real-world and synthetic graphs of different sizes and types. The aim was to analyze graphs with different topological structures, emerging from theoretical models or from real-world data. We decided to include also synthetic graphs to better understand the impact of specific topological structures, and to compare the optimal solution against all heuristics. To evaluate the performance of each heuristic, we analyze the trend of graph connectivity with respect to the number of removed nodes. Specifically, we study how the residual connectivity of the graph decreases as the percentage of removed nodes increases, where the residual connectivity is expressed as the percentage of the initial connectivity of the graph. For an equal percentage of removed nodes, the heuristic with the lowest residual connectivity is considered the best. To take into account time constraints, we analyze also the trend of graph connectivity with respect to running time, that is the time required by the heuristic to select and remove critical nodes. Also in this case, for an equal running time, the heuristic with the lowest residual connectivity is considered the best. In the last case, we aim at understanding if it is possible that, in a given time frame, faster heuristics could select and remove larger sets of nodes and achieve better connectivity results than slower heuristics, that select a smaller set of nodes with better characteristics.

2.1. Networks

We evaluate the CNDP heuristics against both *real-world* and *synthetic* graphs. *Real-world* graphs are obtained from real-world networks of different sizes and types. Following the approach of [15], we selected graphs of five different types: Road Network (RN), Peer to Peer (P2P), Web Graph (WG), Social Network (SN), and Product-Co-Purchase (PCP). The networks have been collected from the following sources: the 9th DIMACS implementation challenge [20], the Stanford Large Network Dataset Collection [21] and the Ref. [22] containing the graph academia-SN. Table 2 summarizes the characteristics of the graphs in our dataset, including a few topological properties.

Synthetic graphs are generated using two well-known network models, Erdős–Rényi (ER) and Square Grid (SG) graphs, chosen in order to test the heuristics against specific network topologies. More precisely, we use rectangular grids, not squared grids, in case of SG graphs. ER graphs are the most general null model to which all real graphs can be compared and are supposedly *hard to disconnect*. SG graphs, on the other hand, represent a stylized version of road networks, the only type of networks for which the CNH is sometimes outperformed by other heuristics (see Section 3.1). We consider synthetic graphs of 5 K, 50 K and 100 K vertices, and *tiny* synthetic graphs of 30 nodes. Given their small size, tiny ER graphs are not guaranteed to present the topological properties that typically emerge in large ER graphs. However, these small ER and SG graphs enable us to qualitatively evaluate how close each heuristic can get to the optimal solution, at least at this tiny scale.

To generate directed ER graphs, other than the number of vertices, we set the average out-degree to 7, in line with the average of all real-world networks in our dataset. To generate directed SG graphs, we proceed as follows: (i) we generate the undirected version of the graph; (ii) we create the directed version of the graph by duplicating edges; (iii) we randomly remove a fraction (2%) of the edges. We decide to remove only 2% of the edges because the real-word RN graphs in our dataset have reciprocity 1, 1 and 0.91, respectively (see Table 2). Note that it is not possible to increase the average degree of SG graphs without modifying the structure of the graph and thus the model. For each model and each graph size, we generated 10 independent graph instances – 100 for tiny graphs – and we applied each heuristic to the giant SCC of all instances.

Table 3 reports the main features of the giant SCC of our synthetic and tiny graphs, with properties averaged over the different instances.

2.2. Experimental setup

We ran all the experiments on a 24-core AMD Ryzen Threadripper 3960X@3.80 GHz system equipped with 256 GB of RAM. Few heuristics, due to their time complexity, require very long execution times. To deal with those cases we set an upper bound timeout of 172800 s (2 days) for each run. When the timeout expires, it executes a graceful shutdown procedure. Concerning the software, we developed a C implementation of CNH and bruteforce, whereas we leveraged existing solutions for the other centrality measures. We are aware that for some metrics there exist in the literature highly parallel and efficient implementations [23,24]. However, only few of them are freely available, and an efficient implementation is not available for all metrics. Thus, for seeking fairness in the comparison, we decided

Table 2
Real-World Graphs — For each graph in the dataset we show: the name, the type, the number of nodes and edges, the density, the average total degree (avg. DG), the total degree centralization (DG cen.), the total degree relative standard deviation computed as ratio between the standard deviation and the mean (DG rsd), the average local clustering coefficient (LC), the reciprocity (R), the assortativity (A) and the diameter (D). The average local clustering coefficient is computed on the undirected version of each graph. We remind the reader that with DG we denote the total degree, i.e. the sum of the out- and in-degree of a given vertex.

Name	Type	Nodes	Edges	Density	avg. DG	DG cen.	DG rsd	LC	R	A	D
amazon0601	PCP	395234	3301092	0.00002	16.70	0.01	0.97	0.426	0.557	−0.0415	52
amazon0302	PCP	241761	1131217	0.00002	9.36	0.00	0.64	0.419	0.544	0.0024	88
p2p-Gnutella31	P2P	14149	50916	0.00025	7.20	0.00	0.60	0.012	0.000	0.1498	30
p2p-Gnutella25	P2P	5153	17695	0.00067	6.87	0.01	0.49	0.010	0.000	0.0247	21
academia-SN	SN	147526	1231158	0.00006	16.69	0.07	2.86	0.274	0.605	−0.0115	20
soc-Epinions1	SN	32223	443506	0.00043	27.53	0.09	2.67	0.279	0.458	−0.0401	16
web-NotreDame	WG	53968	304685	0.00010	10.98	0.14	4.85	0.588	0.389	−0.0615	93
web-BerkStan	WG	334857	4523232	0.00004	27.02	0.23	13.32	0.657	0.226	−0.1747	679
web-Google	WG	434818	3419124	0.00002	15.73	0.01	2.57	0.657	0.377	−0.0634	51
USA-road-d.NY	RN	264346	733846	0.00001	5.55	0.00	0.35	0.025	1.0	0.1814	720
USA-road-d.BAY	RN	321270	800172	0.00001	4.98	0.00	0.40	0.021	1.0	0.0539	837
rome99b	RN	3353	8870	0.00079	5.29	0.00	0.34	0.035	0.909	0.1918	57

Table 3
Synthetic and Tiny Graphs — The numbers in subscript in each graph's name indicate the parameters used to create it — either the number of vertices or the average out-degree for ER graphs, the dimensions of the grid for SG graphs. We generated 100 independent instances of each tiny graph, and 10 independent instances of each other graph. We only considered the giant SCC of each instance. For each graph, the table reports: the name, the type, and the mean over the different instances of the number of nodes, the number of edges, the average total degree (avg. DG), the total degree centralization (DG cen.), the total degree relative standard deviation computed as ratio between the standard deviation and the mean (DG rsd), the average local clustering coefficient (LC), the reciprocity (R), the assortativity (A) and the diameter (D). The average local clustering coefficient is computed on the undirected version of each graph. We remind that DG denotes the total degree, i.e. the sum of the out- and in-degree of a given vertex.

Name	Type	Nodes	Edges	Density	avg. DG	DG cen.	DG rsd	LC	R	A	D
ER_7	tiny ER	30.0	211.2	0.24279	14.08	0.23	0.22	0.427	0.245	−0.0444	3.55
ER_5K	ER	4992.2	35057.6	0.00141	14.04	0.00	0.27	0.003	0.001	−0.0016	8.2
ER_50K	ER	49909.6	349283.7	0.00014	14.00	0.00	0.27	0.000	0.000	−0.0003	10.9
ER_100K	ER	99821.9	698513.5	0.00007	14.00	0.00	0.27	0.000	0.000	0.0002	11.0
GRID_6x5	tiny SG	30.0	96.3	0.11064	6.42	0.06	0.21	0.000	0.982	0.4006	9.01
GRID_50x100	SG	5000.0	19305.1	0.00077	7.72	0.00	0.08	0.000	0.979	0.2802	148.0
GRID_250x200	SG	50000.0	195118.3	0.00008	7.80	0.00	0.06	0.000	0.980	0.1199	448.0
GRID_500x200	SG	100000.0	390579.7	0.00004	7.81	0.00	0.06	0.000	0.979	0.0961	698.0

to rely on igraph,[2] a well-known and widely used, by researchers or analysts dealing with graphs, C-library.

All source codes are available from the paper repository[3] [25], along with the graphs datasets and results. For the generation of synthetic graphs we relied on the NetworkX python library.[4] For each type of synthetic graph we generated a set of networks and we ran the heuristics on all of them, the results are averaged over all runs. The caption of each figure specifies the size of the set of graph instances considered for the experiment.

3. Connectivity VS removed nodes

In this section, we analyze how fast each heuristic disconnects the graph with respect to the percentage of removed nodes. We evaluate the standard (S) and the iterative (I) strategies separately. For each centrality metrics X, the two heuristics based on X are denoted respectively with X-S (standard) and X-I (iterative) in the charts.

3.1. Real-world graphs

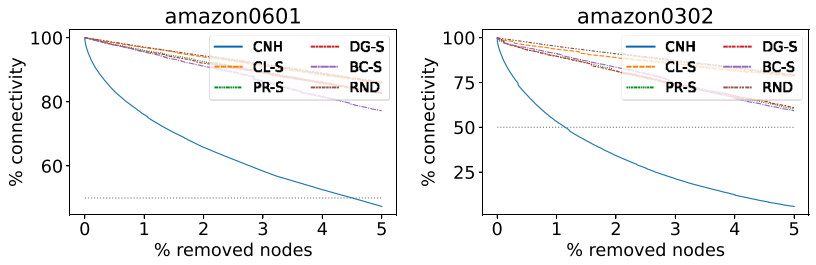
We start discussing the application of the standard strategy to real-world graphs. Figs. 2–6 show the results for each combination of heuristic and graph. In each chart we show only the section that is more relevant for the discussion, by using a different limit for the x-axis. The horizontal gray dotted lines in the charts denote the 50% of residual connectivity.

Generally speaking, we observe that, by picking the right heuristic, we can reach the 50% of residual network connectivity by removing just a small subset of nodes — less than 3% of the initial number of nodes in almost all networks, about 5%–6% in amazon0601, p2p-Gnutella25 and rome99b. CNH is, overall, the best heuristic for identifying critical nodes, i.e. for an equal percentage of removed nodes, CNH almost always reaches the lowest residual connectivity. The only two exceptions are both road networks: in USA-road-d.NY CNH is outperformed by BC-S after roughly 2% of the nodes have been removed; in rome99b CNH is outperformed by BC-I (right after 1.5% of removed nodes), by CL-I (at 2.5%), by CL-S (only between 6% and 7%) and by BC-S (at 8%). The case of the rome99b network is especially noteworthy because both BC-I and CL-I perform remarkably better than CNH.

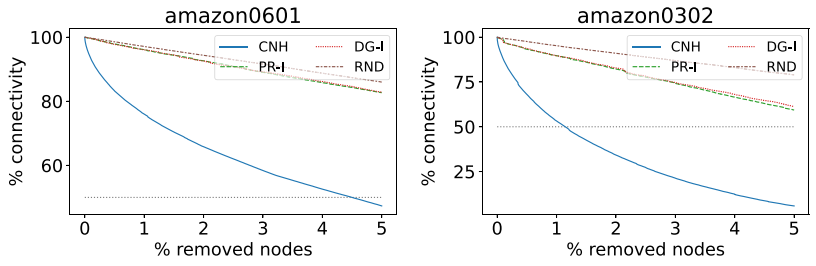
Comparing different types of networks, we observe that

- In PCP and P2P networks (Figs. 2 and 3) centrality-based heuristics generally provide a slow linear drop in the connectivity as k increases, thus performing very poorly when compared with CNH. This happens despite these two types of networks have significant topological differences (see Table 2) and may be related to their limited centralization — even if PCP networks are scale-free, while P2P networks are not. The fact that P2P networks have

[2] https://igraph.org
[3] https://github.com/iac-cranic/seeking_critical_nodes_digraphs
[4] https://networkx.org

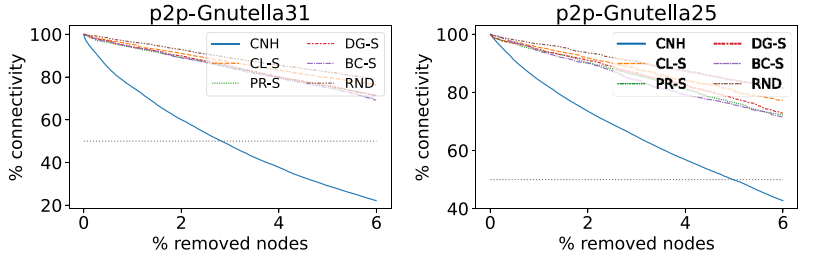


(a) Standard heuristics.

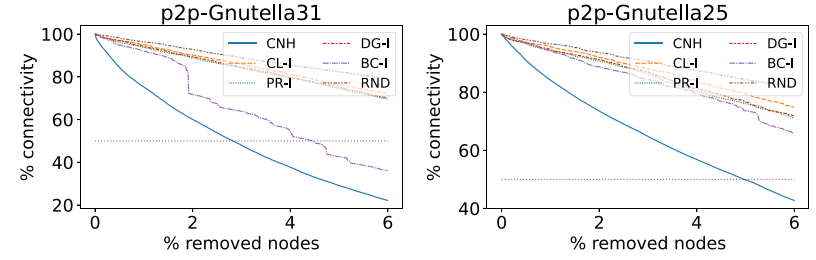


(b) Iterative heuristics.

Fig. 2. Real-world product-co-purchase (PCP). Trend of graph connectivity with respect to the number of removed nodes for real-world PCP graphs using the standard and iterative strategies. The connectivity is the percentage of the initial connectivity of the graph, whereas the number of removed nodes is the percentage of the initial nodes in the graph. The horizontal gray dotted line denotes the 50% of residual connectivity.



(a) Standard heuristics.



(b) Iterative heuristics.

Fig. 3. Real-world peer-to-peer (P2P). Trend of graph connectivity with respect to the number of removed nodes for real-world P2P graphs using the standard and iterative strategies. The connectivity is the percentage of the initial connectivity of the graph, whereas the number of removed nodes is the percentage of the initial nodes in the graph. The horizontal gray dotted line denotes the 50% of residual connectivity.

almost 0 reciprocity may explain why BC-I works slightly better in these networks, especially in p2p-Gnutella31.

- On the other hand, in SN and WG networks (Figs. 4 and 5) some centrality-based heuristics work quite well, possibly due to the pronounced scale-free nature of these networks. In web-NotreDame and web-BerkStan networks, whose DG distribution follows a power-law distribution with exponent <2.5, it is generally easy to find nodes that quickly disrupt the network, even relying on simple metrics such as DG and PR. A similar result has already been discussed in [26], in which the authors study the robustness of social and web graphs to node removal.
- RN networks make a case of their own. In this case, BC- and CL-based heuristics generally outperform the other centrality-based heuristics, and sometimes even CNH. Intuitively, the highly regular structure of road networks (e.g. see the DG centralization and relative standard deviation in Table 2) makes the hubs not so different from all other vertices, and allows to find articulation points only at the periphery of the network. To better understand how the heuristics behave in RN graphs, we analyze a set of synthetic networks that mimic their topological structure, see Section 3.2.

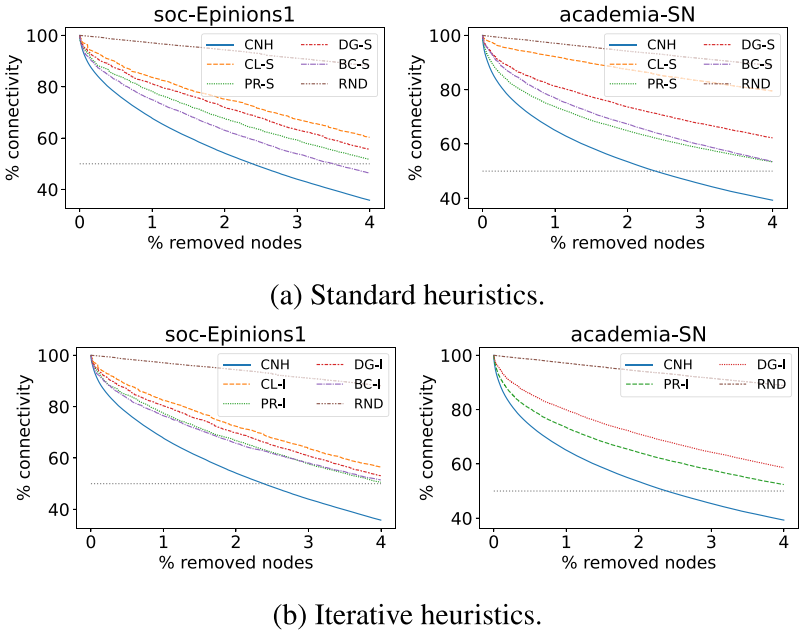


Fig. 4. Real-world social networks (SN). Trend of graph connectivity with respect to the number of removed nodes for real-world SN graphs using the standard and iterative strategies. The connectivity is the percentage of the initial connectivity of the graph, whereas the number of removed nodes is the percentage of the initial nodes in the graph. The horizontal gray dotted line denotes the 50% of residual connectivity.

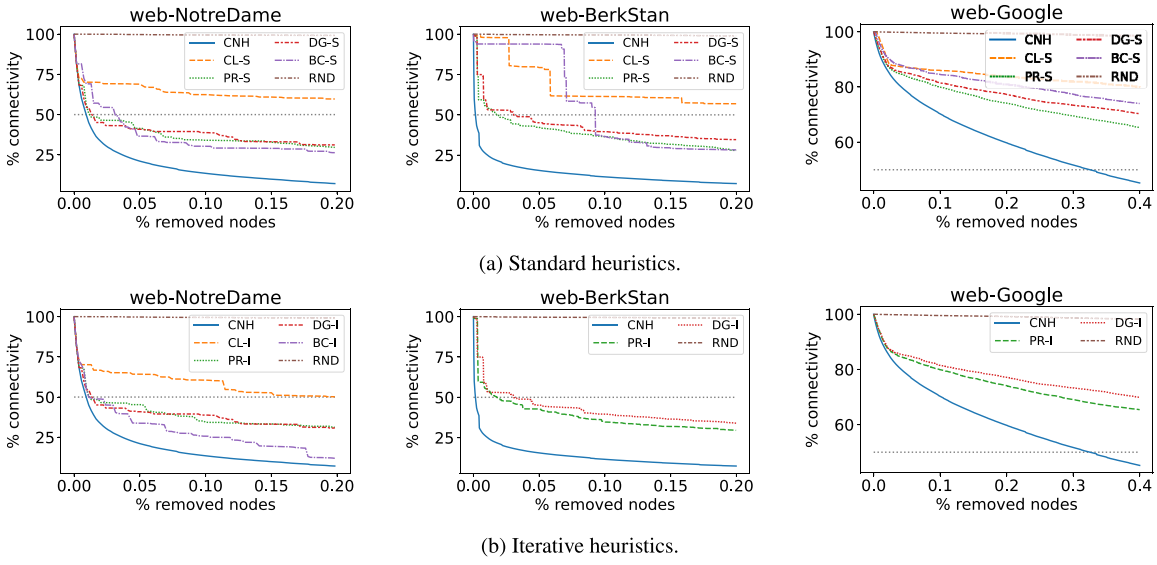
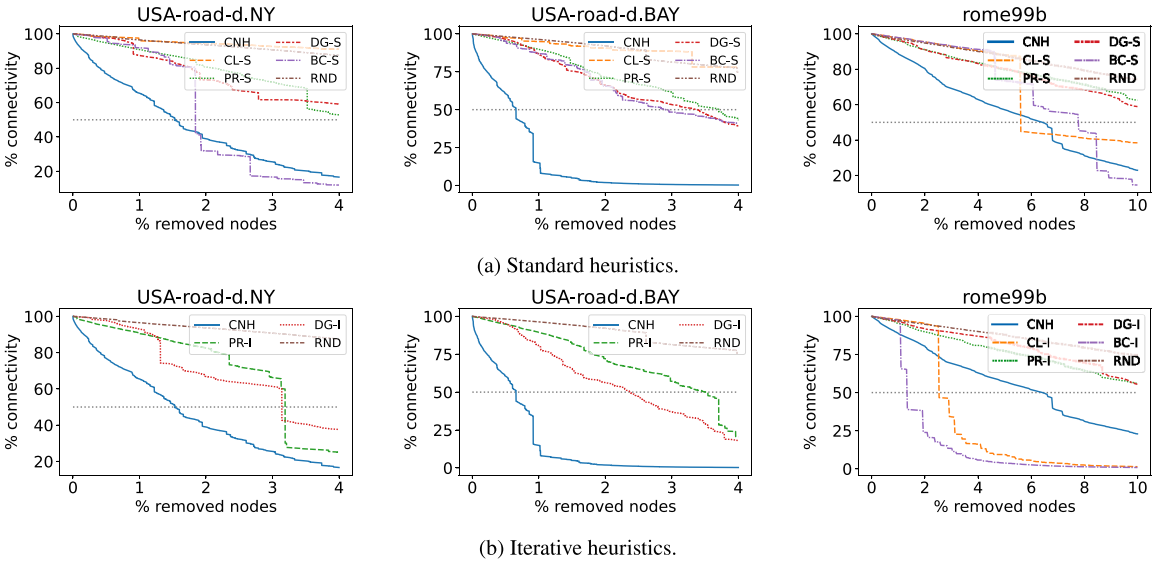


Fig. 5. Real-world web graphs (WG). Trend of graph connectivity with respect to the number of removed nodes for real-world WG graphs using the standard and iterative strategies. The connectivity is the percentage of the initial connectivity of the graph, whereas the number of removed nodes is the percentage of the initial nodes in the graph. The horizontal gray dotted line denotes the 50% of residual connectivity.

Comparing the results of executing each heuristic in standard and iterative mode, we observe that in most cases there is not a notable difference between the two approaches. Most of the time, the iterative and standard strategies achieve very close results with just a slight improvement when the iterative strategy is applied. There are few exceptions of graph and heuristic combinations, that show big improvements when we switch from the standard to the iterative mode and also a couple of cases, for the closeness, in which the standard mode provides even better results than the iterative mode. The iterative strategy requires a longer running time than the standard mode, and this extra cost does not seem to be compensated by the quality of the results. For the iterative strategy, it is important to notice that BC and CL are usually not viable solutions for disconnecting the graphs of our dataset, due to their time complexity. We recall that we set a running time constraint of 2 days for each heuristic, and our results show that only the smallest graphs of the dataset can be processed with these two heuristics. Indeed, we see that BC and CL can just remove a small subset of the nodes needed to disconnect the largest graphs, with a minimal or negligible impact on their connectivity.

3.2. Synthetic graphs

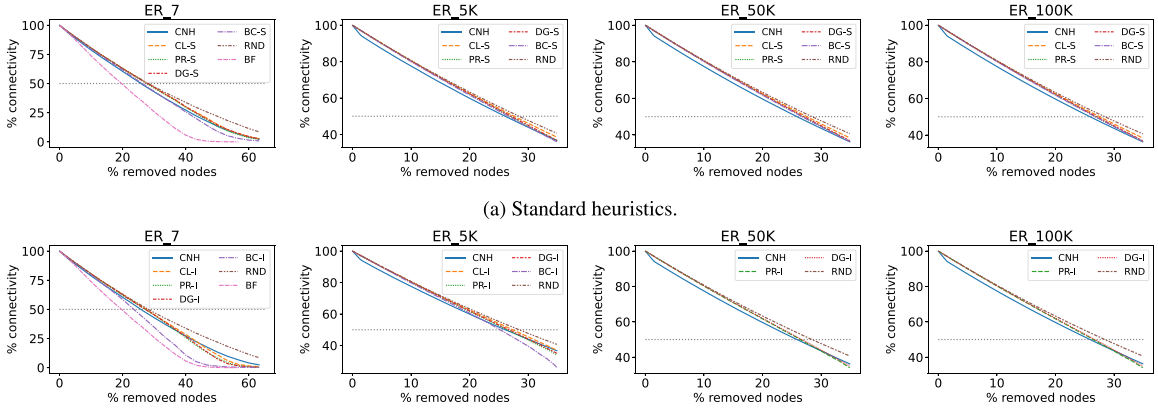
In this section we analyze ER and SG synthetic graphs. We chose to include SG graphs in an attempt to mimic the topological structure of RN graphs, whereas ER graphs have been included because they are the classical model used to generate random graphs and they are supposed to be *hard to disconnect*, as confirmed by our results. Figs. 7 and 8 show the connectivity as a function of the percentage of removed



(a) Standard heuristics.

(b) Iterative heuristics.

Fig. 6. Real-world road networks (RN). Trend of graph connectivity with respect to the number of removed nodes for real-world RN graphs using the standard and iterative strategies. The connectivity is the percentage of the initial connectivity of the graph, whereas the number of removed nodes is the percentage of the initial nodes in the graph. The horizontal gray dotted line denotes the 50% of residual connectivity.



(a) Standard heuristics.

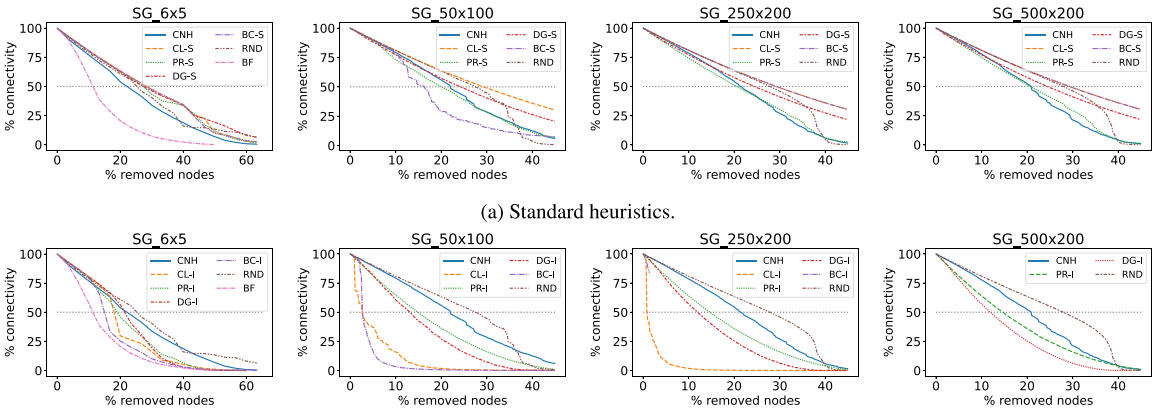
(b) Iterative heuristics.

Fig. 7. Synthetic Erdös–Rényi (ER). Trend of graph connectivity with respect to the number of removed nodes for synthetic ER graphs using the standard and iterative strategies. The connectivity is the percentage of the initial connectivity of the graph, whereas the number of removed nodes is the percentage of the initial nodes in the graph. The horizontal gray dotted line denotes the 50% of residual connectivity. For each model we generated 10 instances – 100 for tiny graphs – to which we applied all heuristics. In the charts we report the average over all runs.

nodes for ER and SG graphs, respectively. The lines in the charts are the average computed over 100 different graph instances, for tiny graphs of 30 nodes, and over 10 different graph instances, for graphs of size 5 K, 50 K and 100 K.

Fig. 7 confirms the intuition that the homogeneity of ER graphs make them, in general, hard to disconnect. In the tiny ER_7 graph the results obtained with BC-I are remarkably close to those given by a brute-force optimal solution, whereas all other heuristics are far from optimal. This result, however, is not replicated in larger graphs, where we need to remove more than 25% of the nodes to halve the connectivity of the graph regardless of the chosen algorithm — compared to the 6% needed, at worst, in real-world graphs. We observe no significant differences between the results obtained by different centrality-based heuristics in large ER graphs, despite these graphs having very variable density. Suggesting that other topological features have a major impact in solving the CNDP and density alone does not give us directions on how much a network is hard to disconnect. CNH always outperforms the other heuristics for small k, but the advantage vanishes as k increases. It is worth noting that, at a high level, the graph p2p-Gnutella25 may seem quite similar to ER_5K (see Tables 2 and 3). All centrality-based metrics indeed perform similarly for these two graphs. However, CNH works remarkably better in p2p-Gnutella25 than ER_5K, suggesting that CNH benefits from some non-evident topological feature of p2p-Gnutella25.

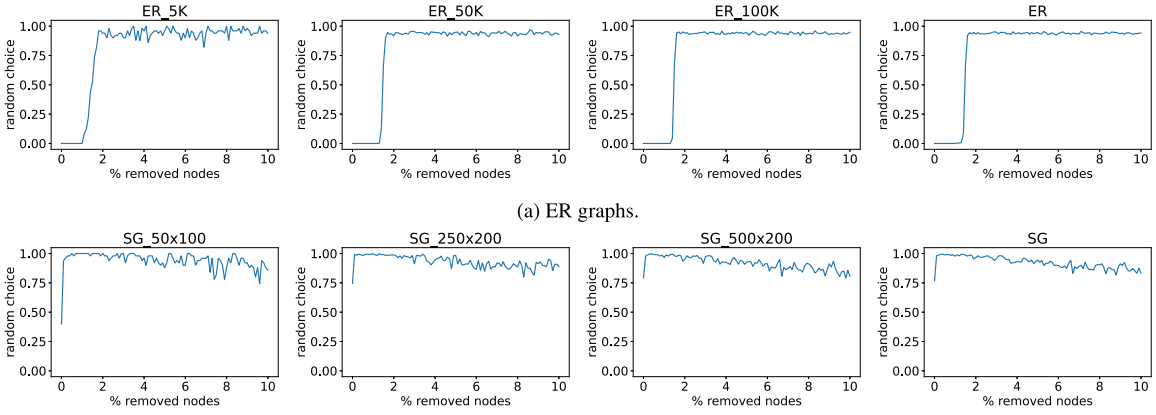
SG graphs provide a different and more varied scenario. In Fig. 8(a), we see that PR-S is the best standard heuristic for large SG graphs, whereas BC-S works especially well in SG_50x100, where it significantly outperforms CNH. More generally, CNH performs quite poorly in these graphs. If we look at Fig. 8(b), we realize that all iterative heuristics disconnect the network much faster than CNH. BC-I and CL-I, in particular, provide near-optimal solutions for the tiny SG_6x5 graph, and they completely disconnect the larger SG graphs by removing a very small percentage of the nodes. A possible explanation is that a perfectly regular SG only has articulation points at the extreme



(a) Standard heuristics.

(b) Iterative heuristics.

Fig. 8. Synthetic square grid (SG). Trend of graph connectivity with respect to the number of removed nodes for synthetic SG graphs using the standard and iterative strategies. The connectivity is the percentage of the initial connectivity of the graph, whereas the number of removed nodes is the percentage of the initial nodes in the graph. The horizontal gray dotted line denotes the 50% of residual connectivity. For each model we generated 10 instances – 100 for tiny graphs – to which we applied all heuristics. In the charts we report the average over all runs.



(a) ER graphs.

(b) SG graphs.

Fig. 9. Probability to find no articulation points in synthetic graphs. The y-value associated to a certain percentage of removed nodes is the fraction of runs in which, at that step of the algorithm, CNH made a "random choice" because it did not find any articulation point. The y-values range from 0, meaning that an articulation point was found at that step in all runs, to 1, meaning that a random node was selected at that step in all runs. Charts 'ER' and 'SG' show how many times a "random choice" has been made in ER and SG graphs respectively, values are averaged over all runs.

periphery of the network – so that CNH behaves as a very greedy algorithm – or has no articulation points at all — so that CNH works like a random algorithm. Unfortunately, BC-I and CL-I are the heuristics with the highest time complexity, and might not be viable solutions for large graphs (e.g. both heuristics reach the timeout for very small k in SG_500x200). Considering time constraints, PR-S, PR-I and DG-I guarantee a good trade-off between efficacy and efficiency. The limited agreement between the results obtained for synthetic SG graphs and for real-world RN graphs (see Fig. 6) suggests that real-world road networks are sufficiently irregular to bring out the advantages of a specific algorithm like CNH.

In general, CNH performed significantly worse in our synthetic graphs than it did in the real-world graphs analyzed in Section 3.1. To better understand why, we looked more closely at what happened at each iteration of the algorithm. CNH selects a random vertex at every iteration in which it does not find any articulation point. In Fig. 9, the y-value associated to a certain step of the algorithm (expressed as a percentage of removed nodes) is the fraction of runs in which, at that step, the algorithm made a "random choice". The y-values range from 0, meaning that an articulation point was selected at that step in all runs, to 1, meaning that a random node was chosen at that step in all runs. In ER graphs, CNH manages to identify actual critical nodes in the very first steps, but, after 1% of the nodes has been removed, it starts to mostly work as a random heuristic, meaning that there are no more articulation points in the graph. The fact that CNH behaves no worse than the other heuristics in ER graphs (see Fig. 7), however, suggests that, after a few critical nodes have been removed, one cannot do much better than this. In SG graphs, CNH basically chooses random nodes since the beginning, because a regular grid has essentially no articulation points. In this case, the comparison with other heuristics clearly shows that selecting the nodes based on their network centrality works a lot better than choosing them at random. We thus argue that CNH could be modified to operate a better choice when there are no articulation points in the graph.

The absence of articulation points at different steps of the heuristic – and the consequent removal of random nodes – does not explain, however, the results obtained for real graphs. Fig. 10 is the analogous of Fig. 9 for real-world networks: for each percentage of removed nodes, the plots show if CNH selected the node to remove randomly or if it selected an articulation point. In USA-road-d.NY, the algorithm starts to randomly select nodes only after 6% of nodes have been removed, i.e. only once the residual connectivity is below 20%. In rome99b, the algorithm always selects an articulation point — at least up to 10% of the network is removed. Nonetheless, as already discussed, in these two networks CNH is not the best heuristic for identifying critical nodes.

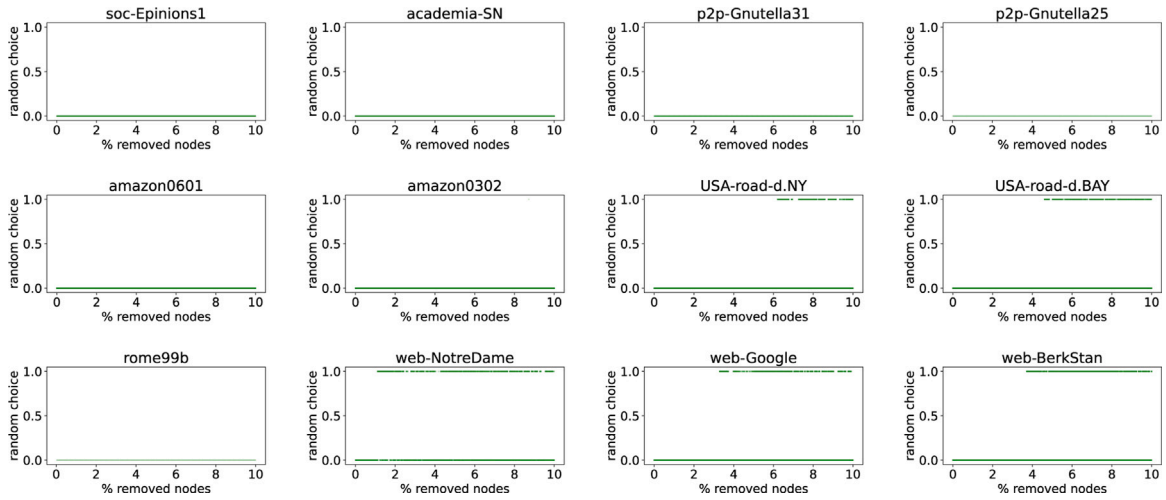


Fig. 10. CNH random choice — real-world. For each step of CNH, the charts show how the heuristic chooses the node to remove. 0 means that an articulation point was found and removed at that step. 1 means that a random node was selected and removed at that step.

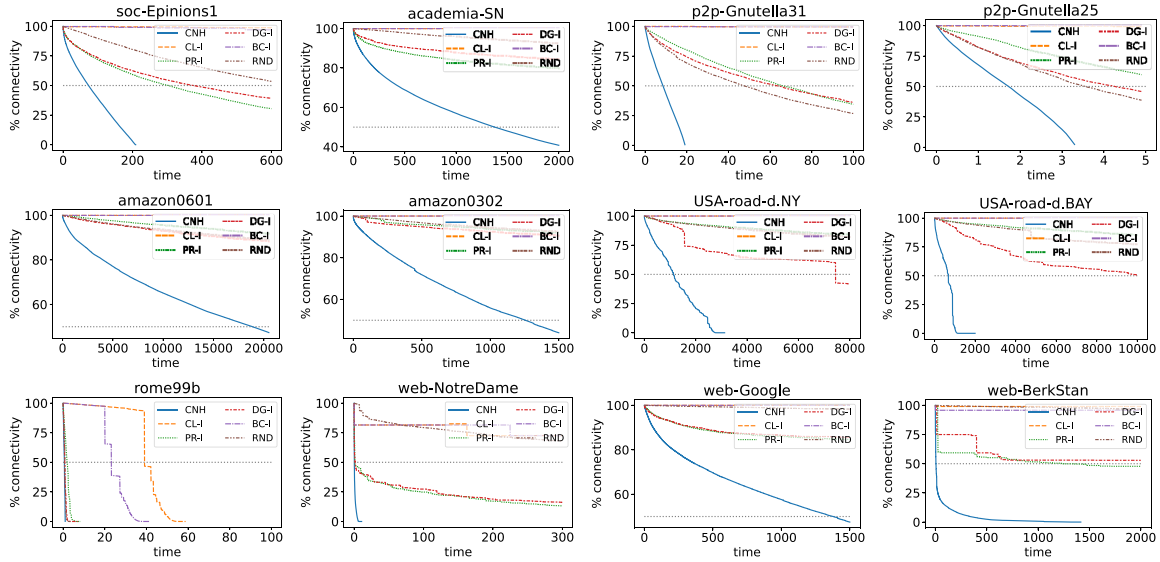


Fig. 11. Iterative — real-world. Trend of graph connectivity with respect to running time for real-world graphs using the *iterative* strategy. The connectivity is the percentage of the initial connectivity of the graph, the running time is expressed in seconds. Each heuristic is limited to remove maximum 10% of the nodes. The horizontal gray dotted line denotes the 50% of residual connectivity.

4. Connectivity VS running time

While identifying the best critical nodes is important, the time required by each heuristic to identify such critical nodes is another determining factor that we must take into account. This factor is more important as the number of nodes in the graph increases, indeed for some heuristics it could become impossible to identify critical nodes in the given time frame in large graphs, as shown in the previous sections. In many application settings, the best heuristic is thus the one that guarantees an optimal trade-off between the residual connectivity and the execution time. Only for comparable connectivity in comparable time, the heuristic that removes the minimum number of nodes is actually to be preferred.

In this section, we consider a variant of the CNDP, denoted CNDP(α), that is not explicitly parametrized by the number k of nodes to remove. We select a target $\alpha \in [0, 1)$ and we aim at finding the fastest possible heuristic that degrades the connectivity of the network to some $\gamma \leq \alpha\gamma_0$, where γ_0 is the initial connectivity of the graph. In theory, this variant of the CNDP does not pose any constraint to the number k of removed nodes. In practice, however, we do limit the number of nodes that each heuristic can remove to $k = 10\%$ of the network, knowing from Section 3.1 that 6% is enough to halve the connectivity of any graph in our dataset when the best heuristic for the graph is used. This is necessary for, at least, two reasons: (i) the maximum number of nodes that we can remove is often a constraint of the application, (ii) with no limits to k, removing all nodes of the network would always be the optimal solution to the problem. To address the CNDP(α), we study how fast each heuristic disconnects the graph analyzing the trend of graph connectivity with respect to running time. We remind the reader that we included the random heuristic as baseline.

Fig. 11 shows the connectivity obtained by each iterative heuristic as a function of the running time. To compute the running time for a given algorithm, at each step we sum up the time needed to: (i) select the node to remove; (ii) remove the node; (iii) compute the new connectivity, which includes decomposing the graph into SCCs. Step (iii) is necessary to define a stopping criterion for the CNDP(α), for which the number of nodes to remove is not known. CNH is, by far, the fastest algorithm. Even heuristics based on very fast metrics such as DG and PR, in fact, need to repeat steps (ii) and (iii) at each iteration. These two steps are very efficient in our implementation of CNH because

Table 4
For each heuristic X we show the time t_X required to compute the nodes ranking, and the minimum α_X that can be reached by removing 10% of the nodes. t_{CNH} is the time required by CNH to reach $\alpha = 0$ or $k = 10\%$. Nodes are removed based on the ranking computed by each heuristic. For each network the values of $\alpha_X > 0.5$ are in bold and the minimum α_X is underlined.

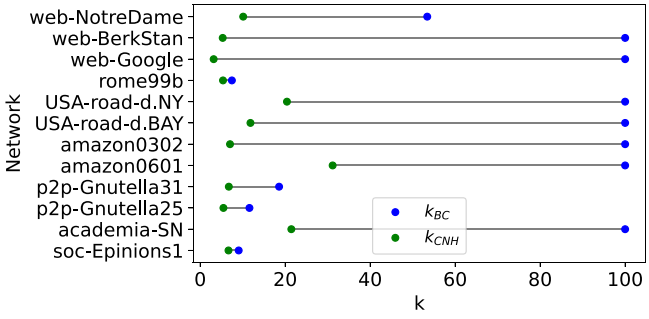
Type	Name	α_{DG}	α_{PR}	α_{CL}	α_{BC}	α_{CNH}	t_{DG}	t_{PR}	t_{CL}	t_{BC}	t_{CNH}
SN	soc-Epinions1	0.23	0.19	0.28	0.16	0.08	0.003	0.09	1e+02	2e+02	2e+02
	academia-SN	0.38	0.3	**0.62**	0.29	0.16	0.02	0.5	5e+03	9e+03	4e+03
P2P	p2p-Gnutella25	**0.56**	**0.55**	**0.62**	**0.53**	0.2	0.0002	0.007	1e+00	2e+00	3e+00
	p2p-Gnutella31	**0.54**	**0.53**	**0.63**	**0.51**	0.03	0.0008	0.03	1e+01	2e+01	2e+01
PCP	amazon0601	**0.68**	**0.66**	**0.73**	**0.63**	0.28	0.06	1e+00	4e+04	7e+04	3e+04
	amazon0302	0.34	0.31	**0.61**	0.25	0.0	0.01	0.4	1e+04	2e+04	4e+03
RN	USA-road-d.BAY	0.007	0.021	**0.72**	0.23	9.3e−05	0.01	0.4	1e+04	2e+04	1e+03
	rome99b	**0.59**	**0.63**	0.38	0.15	0.23	0.0001	0.004	0.5	0.6	0.9
	USA-road-d.NY	0.24	0.036	**0.8**	0.008	0.0022	0.009	0.3	9e+03	2e+04	3e+03
WG	web-NotreDame	0.0014	0.0018	0.11	0.0022	5.2e−05	0.002	0.1	2e+02	2e+02	8e+00
	web-Google	**0.51**	0.47	**0.69**	**0.54**	0.0	0.04	1e+00	3e+04	6e+04	5e+03
	web-BerkStan	0.21	0.12	0.38	0.11	0.0	0.04	1e+00	1e+04	2e+04	1e+03

they exploit the data structures used to find the articulation points of the graph. In particular, in step (ii) we manage nodes relabeling at SCC level, so when we remove a node, we re-assign labels only to the nodes belonging to the same SCC. In step (iii) we compute the new connectivity as the difference between the connectivity before node x removal and the CNH score of node x. The case of the rome99b network is especially noteworthy: even if BC-I and CL-I can disconnect the graph by removing fewer nodes than CNH (see Fig. 6), CNH can disconnect the graph a lot faster than both BC-I and CL-I. Summing up, if the goal is degrading the connectivity of the network to some fraction α of the initial connectivity, CNH is the fastest of all iterative heuristics considered in this paper. Additionally, in all considered real-world graphs, except rome99b, it does so by removing fewer nodes than the other heuristics.

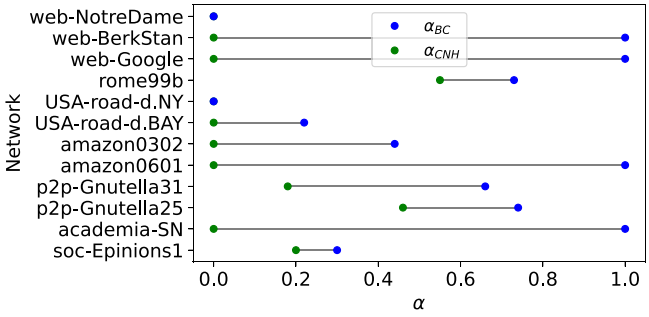
When we consider standard heuristics, most of the competitive advantage of CNH is lost. The time required by any standard heuristic X-S to address the CNDP(α) is, in fact, the sum of the time needed to: (a) compute the metric X once; (b) find the minimum k such that removing the k top-ranking nodes decreases the connectivity by a factor α. Step (b) can be implemented with a binary search method, identical for all metrics, which requires to remove a bunch of nodes and compute the obtained connectivity just a logarithmic number of times — log of one tenth of the network size, if k is bounded. In the following, we make the conservative assumption that the time for step (b) is negligible, and we approximate the running time of heuristic X-S by the time required to compute the metric X in the original graph.

The asymptotic running time of CL and BC is roughly $O(N)$ times greater than that of DG and PR (see Table 1). The values reported in Table 4 confirm that this is true, in practice, for all of our real-world graphs. In Table 4, t_X is the running time, in seconds, of metric X, whereas α_X is the relative connectivity obtained by the X-S heuristic by removing the top 10% of the network — in bold, all cases where $\alpha_X > 0.5$. When CNH completely disrupts the network by removing less than 10% of the network, t_{CNH} is the time required by CNH to reach $\alpha = 0$. Table 4 shows that CL-S is hardly useful in practice: it is the least effective of all standard heuristics, while being very time consuming. CNH and BC-S generally outperform, in terms of α, both DG-S and PR-S. CNH, however, runs in time comparable to BC-S, and both these heuristics are orders of magnitude slower than DG-S and PR-S. Whether the extra running time of CNH and BC-S is affordable depends on the specific application setting. Generally speaking, PR-S offers a reasonable trade-off between running time and efficacy.

While the time to remove 10% of the network with CNH and BC-S is comparable, CNH is always preferable to BS-S, as clarified by Fig. 12. If we bound CNH to the running time t_{BC}, we may measure the pros of using CNH by respectively fixing the target connectivity or the number of removed nodes. We define k_{CNH} and α_{CNH} as, respectively, the



(a) k_{CNH} vs. k_{BC} for our real-world graphs.



(b) α_{CNH} vs. α_{BC} for our real-world graphs.

Fig. 12. Comparison of CNH and BC-S when bound to the running time of BC-S. k_{CNH} and α_{CNH} are, respectively, the percentage of the network that can be removed in the running time of BC-S using CNH, and the connectivity obtained removing those k_{CNH} nodes; k_{BC} is the percentage of the network that has to be removed with BC-S to obtain the connectivity α_{CNH}; α_{BC} is the connectivity obtained with BC-S removing the k_{CNH} top-ranking nodes.

number of nodes that CNH removes in time t_{BC} and the α reached by CNH by removing such k_{CNH} nodes. In Fig. 12(a), we compare k_{CNH} with the number of nodes that BC-S must remove to reach α_{CNH}, denoted k_{BC}. In Fig. 12(b), we compare α_{CNH} with the α reached by BC-S removing k_{CNH} nodes, denoted α_{BC}. In many cases $k_{CNH} \ll k_{BC}$ and $\alpha_{CNH} \ll \alpha_{BC}$, meaning that CNH is clearly better than BC-S from all points of view. In some relatively small networks, such as rome99b and soc-Epinions1, $k_{CNH} \approx k_{BC}$, but $\alpha_{CNH} < \alpha_{BC}$. In web-NotreDame and USA-road-d.NY, CNH can completely disrupt the network (i.e. $\alpha_{CNH} = 0$) in time t_{BC}, removing a lot fewer nodes than what would be needed with BC-S (i.e. $k_{CNH} \ll k_{BC}$).

5. Conclusions and future work

Understanding which nodes are critical in keeping a network connected is a widely-studied problem, usually known as the Critical Node Detection Problem (CNDP). The CNDP is generally parametrized by an integer k and it is formulated as the problem of finding the set of k nodes whose deletion results in the minimum pairwise connectivity among the remaining nodes. In directed graphs, the pairwise connectivity is the number of vertex pairs that are strongly connected in the graph. The CNDP is NP-hard [13], so that heuristic algorithms are needed to solve it in practice.

A linear-time optimal algorithm for the case $k = 1$ in digraphs has been proposed by Paudel et al. in [15]. In this paper, we investigated whether iterating Paudel's algorithm is a viable solution to address the CNDP for a generic k. The main idea of our iterative Critical Node Heuristic (CNH) – identifying and removing the most critical node in the network, to obtain the input graph of the following iteration – can be generalized to work with any centrality metrics. We defined four other iterative heuristics, DG-I, PR-I, CL-I and BC-I, respectively based on the total degree, pagerank, closeness, and betweenness centralities, and their "standard" counterparts, DG-S, PR-S, CL-S and BC-S, where the metrics are used to rank all nodes in the graph and the k top-ranking nodes are removed altogether. We studied the performance of our CNH and all other iterative and standard heuristics, on both real-world and synthetic graphs, measuring their effectiveness and efficiency, also taking into account time constraints.

Our analysis shows that, in general, CNH is a very good option to identify critical nodes in real-world digraphs: it is quite fast and allows to quickly disrupt the network by removing a relatively small number of nodes. The running time of CNH is comparable to the time needed to compute the closeness or the betweenness centrality just once on the input graph, but it provides much better results than both CL-S and BC-S heuristics. Generally speaking, the longer running time required by the iterative heuristics are not always compensated by the quality of the results. The centrality-based heuristic that benefits more from an iterative strategy is BC-I, but iterating the computation of the betweenness centrality is not a viable solution for large graphs. In cases where the running time of CNH is not affordable, PR-S and PR-I provide a good trade-off between time and effectiveness.

The topological structure of the input graph has a remarkable impact on both the average performance of the considered heuristics and the gain provided by CNH with respect to other heuristics. In the very homogeneous Erdős–Rényi random graphs, all metrics perform equally poorly, as partially known. In peer-to-peer and product-co-purchase networks, CNH is the only heuristic that guarantees a super-linear decrease of the connectivity as a function of k. In networks with a pronounced scale-free structure, such as social networks and web graphs, it is generally easier to disconnect the network with any of the considered heuristics, and especially easy with CNH. Finally, a few centrality-based metrics perform comparably to, or even better than, CNH in road networks. This trend is even more visible in regular square grid networks – which can be seen as a stylized and rigid version of road networks – where CNH rarely finds any articulation points and is thus outperformed by all iterative centrality-based heuristics.

Our results suggest possible directions that we plan to explore in the future, in order to improve the performance of our CNH algorithm. In particular, we identified graphs without articulation points as a possible main issue for CNH. At any iteration in which no articulation points are found, in fact, CNH removes a node chosen uniformly at random in the network. This sometimes results in a considerable degradation of its effectiveness. A hybridization of CNH with other heuristics, in which the random choice is replaced by an informed choice that considers other forms of network centrality, might be a suitable correction. Such a change in the algorithm must be done taking into account the time complexity needed to make the choice: we either compute the centrality just once, at the very beginning of the process, or we need a metric that has a linear-time complexity, such as the degree or pagerank, to prevent from losing one of the main advantages of CNH.

Finally, we believe that revising the algorithm by Paudel et al. and optimizing the implementation of CNH is an important contribution towards making these algorithms usable in practice. To this end, all the source code used in this paper is freely available on the paper repository [25], together with the datasets and the experimental results.

CRediT authorship contribution statement

Massimo Bernaschi: Conceptualization, Writing – review & editing. **Alessandro Celestini:** Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing, Visualization, Data analysis, Data curation. **Marco Cianfriglia:** Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing, Data analysis, Data curation. **Stefano Guarino:** Conceptualization, Writing – review & editing, Data analysis, Visualization. **Giuseppe F. Italiano:** Conceptualization, Writing – review & editing. **Enrico Mastrostefano:** Conceptualization, Software, Writing – review & editing. **Lena Rebecca Zastrow:** Conceptualization, Software, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgment

The authors would like to thank Loukas Georgiadis for useful discussions and comments. This work was partially supported by TAILOR Project EU HORIZON 2020 Research and Innovation Programme GA No 952215 https://tailor-network.eu; and MIUR, the Italian Ministry for Education, University and Research, under PRIN Project AHeAD (Efficient Algorithms for HArnessing Networked Data).

References

[1] S. Borgatti, Identifying sets of key players in a social network, Comput. Math. Organ. Theory 12 (1) (2006) 21–34.
[2] K. Das, S. Samanta, M. Pal, Study on centrality measures in social networks: a survey, Soc. Netw. Anal. Min. 8 (1) (2018) 1–11.
[3] Y. Zenou, Key players, in: The Oxford Handbook of the Economics of Networks, Vol. 11, 2016.
[4] A. Celestini, M. Cianfriglia, E. Mastrostefano, A. Palma, F. Castiglione, P. Tieri, Critical nodes reveal peculiar features of human essential genes and protein interactome, in: 2019 IEEE International Conference on Bioinformatics and Biomedicine, BIBM, IEEE, 2019, pp. 2121–2128.
[5] D. Kempe, J. Kleinberg, É. Tardos, Maximizing the spread of influence through a social network, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, pp. 137–146.
[6] E. Bakshy, J. Hofman, W. Mason, D. Watts, Everyone's an influencer: Quantifying influence on Twitter, in: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, WSDM '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 65–74, http://dx.doi.org/10.1145/1935826.1935845.
[7] S. Guarino, N. Trino, A. Celestini, A. Chessa, G. Riotta, Characterizing networks of propaganda on Twitter: a case study, Appl. Netw. Sci. 5 (1) (2020) 1–22.
[8] K. Shu, A. Sliva, S. Wang, J. Tang, H. Liu, Fake news detection on social media: A data mining perspective, ACM SIGKDD Explor. Newsl. 19 (1) (2017) 22–36.
[9] Y. Chen, G. Paul, S. Havlin, F. Liljeros, H. Stanley, Finding a better immunization strategy, Phys. Rev. Lett. 101 (5) (2008) 058701.
[10] F. Bauer, J. Lizier, Identifying influential spreaders and efficiently estimating infection numbers in epidemic models: A walk counting approach, Europhys. Lett. 99 (6) (2012) 68007.

[11] R. Cohen, S. Havlin, D. Ben-Avraham, Efficient immunization strategies for computer networks and populations, Phys. Rev. Lett. 91 (24) (2003) 247901.
[12] M. Lalou, M. Tahraoui, H. Kheddouci, The critical node detection problem in networks: a survey, Comp. Sci. Rev. 28 (2018) 92–117.
[13] A. Arulselvan, C. Commander, L. Elefteriadou, P. Pardalos, Detecting critical nodes in sparse graphs, Comput. Oper. Res. 36 (7) (2009) 2193–2200.
[14] T. Dinh, Y. Xuan, M. Thai, P. Pardalos, T. Znati, On new approaches of assessing network vulnerability: Hardness and approximation, IEEE/ACM Trans. Netw. 20 (2) (2012) 609–619.
[15] N. Paudel, L. Georgiadis, G. Italiano, Computing critical nodes in directed graphs, J. Exp. Algorithmics 23 (2) (2018) 2.
[16] L. Georgiadis, G. Italiano, N. Parotsidis, Strong connectivity in directed graphs under failures, with applications, SIAM J. Comput. 49 (5) (2020) 865–926.
[17] M. Ventresca, D. Aleman, Efficiently identifying critical nodes in large complex networks, Comput. Soc. Netw. 2 (1) (2015) 6.
[18] L. Page, S. Brin, R. Motwani, T. Winograd, The Pagerank Citation Ranking: Bringing Order to the Web, Tech. Rep., Stanford InfoLab, 1999.
[19] U. Brandes, A faster algorithm for betweenness centrality, J. Math. Sociol. 25 (2) (2001) 163–177.
[20] C. Demetrescu, A. Goldberg, D. Johnson, 9th Dimacs Implementation Challenge–Shortest Paths, American Mathematical Society, 2006.
[21] J. Leskovec, A. Krevl, Snap Datasets: Stanford Large Network Dataset Collection, 2014.
[22] M. Fire, L. Tenenboim-Chekina, R. Puzis, O. Lesser, L. Rokach, Y. Elovici, Computationally efficient link prediction in a variety of social networks, ACM Trans. Intell. Syst. Technol. 5 (1) (2014) http://dx.doi.org/10.1145/2542182.2542192.
[23] M. Bernaschi, G. Carbone, F. Vella, Scalable betweenness centrality on multi-gpu systems, in: Proceedings of the ACM International Conference on Computing Frontiers, 2016, pp. 29–36.
[24] A. Rungsawang, B. Manaskasemsak, Fast pagerank computation on a gpu cluster, in: 2012 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, IEEE, 2012, pp. 450–456.
[25] M. Bernaschi, A. Celestini, M. Cianfriglia, S. Guarino, G. Italiano, E. Mastrostefano, L. Zastrow, Seeking critical nodes in digraphs, 2022, http://dx.doi.org/10.5281/zenodo.1234, URL https://github.com/github/iac-cranic/seeking_critical_nodes_digraphs.
[26] P. Boldi, M. Rosa, S. Vigna, Robustness of social and web graphs to node removal, Soc. Netw. Anal. Min. 3 (4) (2013) 829–842.

Massimo Bernaschi has been 10 years with IBM working in High Performance Computing. Currently he is with the National Research Council of Italy (CNR) as Chief Technology Officer of the Institute for Applied Computing "M. Picone". He is an adjunct professor at "LUISS" University in Rome and has been named CUDA-Fellow in 2012.

Alessandro Celestini is a researcher at the Institute for Applied Computing "M. Picone" of the National Research Council of Italy. He holds both a Bachelor and Master's degree in Computer Science, from the University of Rome "La Sapienza", and a Ph.D. In Computer Science and Engineering, from the School for Advanced Studies IMT Lucca. His research interests include complex network analysis, graph algorithms and high performance computing.

Marco Cianfriglia is a researcher at Department of Mathematics and Physics of Roma Tre University. He holds both a Bachelor and Master's degree in Computer Science, from the University of Rome "La Sapienza", and a Ph.D in Mathematics from Roma Tre University. His main interests are high performance computing, digital forensics, and security.

Stefano Guarino earned his Ph.D. in Mathematics from the University of Roma Tre, and he is now a researcher at the Institute for Applied Computing "M. Picone" of the National Research Council of Italy. His research activity focuses on data analysis and security, social network modeling and analysis, and the study of diffusion processes on networks.

Giuseppe F. Italiano is Professor of Computer Science at Luiss University. He has been Visiting Professor at Columbia University, Université Paris-Sud, Max–Planck–Institut fur Informatik and Hong Kong University of Science & Technology. Most of his research is centered around algorithms, and in 2016 he was nominated Fellow of the European Association for Theoretical Computer Science for "fundamental contributions to the design and analysis of algorithms for solving theoretical and applied problems in graphs and massive datasets, and for his role in establishing the field of algorithm engineering".

Enrico Mastrostefano has received his degree in Physics and a Ph.D. in Computer Science from "Sapienza University of Rome". Currently he is a research fellow at the Institute for applied mathematics "Mauro Picone" (IAC) - CNR. His research interests include complex networks, mathematical modeling of biological systems and high performance computing.

Lena Rebecca Zastrow received her Ph.D. in Physics at the University of Rome "Roma Tre" and her degree in Physics at the University of Rome "Sapienza". Her research interests lie in the area of computational modeling of complex networks and biological systems.