Computer Science Review 28 (2018) 92–117

Contents lists available at ScienceDirect

Computer Science Review

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

Survey

The Critical Node Detection Problem in networks: A survey

Mohammed Lalou [a,b,c,*], Mohammed Amin Tahraoui [c], Hamamache Kheddouci [c]

[a] Département d'Informatique, Faculté des Sciences Exactes, Université A. Mira Béjaia, 06000 Béjaia, Algeria
[b] Institut des Sciences et Technologie, Centre Universitaire A. Boussouf Mila, 43000 Mila, Algeria
[c] Université Claude Bernard Lyon 1-UFR-Informatique, Lab LIRIS, 43 bd du 11 Novembre 1918, F-69622 Villeurbanne Cedex, France

ARTICLE INFO

Article history:
Received 29 December 2016
Received in revised form 18 February 2018
Accepted 20 February 2018

Keywords:
Critical node
Important node
Node-deletion problem
Complex network
Combinatorial optimization
Complexity
Approximation schemes
Graph connectivity

ABSTRACT

In networks, not all nodes have the same importance, and some are more important than others. The issue of finding the most important nodes in networks has been addressed extensively, particularly for nodes whose importance is related to network connectivity. These nodes are usually known as *Critical Nodes*. The *Critical Node Detection Problem* (*CNDP*) is the optimization problem that consists in finding the set of nodes, the deletion of which maximally degrades network connectivity according to some predefined connectivity metrics. Recently, this problem has attracted much attention, and depending on the predefined metric, different variants have been developed. In this survey, we review, classify and discuss several recent advances and results obtained for each variant, including theoretical complexity, exact solving algorithms, approximation schemes and heuristic approaches. We also prove new complexity results and induce some solving algorithms through relationships established between different variants.

Contents

1. Introduction ... 93
   1.1. Problem relevance ... 94
   1.2. Aims and motivations ... 94
   1.3. Outline of the paper ... 94
2. Definitions and notations ... 95
   2.1. Graph theory ... 95
   2.2. Computational complexity ... 96
   2.3. Approximation algorithms ... 96
3. Identifying critical nodes in networks ... 96
   3.1. Problem formulation ... 96
   3.2. Connectivity metric importance ... 97
   3.3. Problem complexity ... 97
4. Solving CNDP ... 97
   4.1. A general greedy approach for CNDP ... 97
5. Variants of CNDP ... 98
   5.1. Classification ... 98
   5.2. CNP — Critical node problem ... 99
      5.2.1. Complexity results ... 99
      5.2.2. Solving CNP ... 101
   5.3. MaxNum — Maximize the number of connected components ... 104
      5.3.1. Complexity results ... 105
      5.3.2. Solving MaxNum ... 106
   5.4. MinMaxC — Minimize the largest connected component size ... 106
      5.4.1. Complexity results ... 106

* Corresponding author at: Département d'Informatique, Faculté des Sciences Exactes, Université A.Mira Béjaia, Algeria.
E-mail addresses: mohammed.lalou@univ-lyon1.fr, m.lalou@centre-univ-mila.dz (M. Lalou), mohammed-amin.tahraoui@univ-lyon1.fr (M.A. Tahraoui), hamamache.kheddouci@univ-lyon1.fr (H. Kheddouci).

https://doi.org/10.1016/j.cosrev.2018.02.002
1574-0137/

5.4.2. Solving MinMaxC 107
5.5. CC-CNP: Cardinality constrained critical node problem 107
5.5.1. Complexity results 108
5.5.2. Solving CC-CNP 108
5.6. β-vertex disruptor Problem 109
5.6.1. Complexity results 110
5.6.2. Solving the β-vertex disruptor problem 110
5.7. Other variants 110
6. Discussion and future research directions 111
7. Applications of CNDP 113
8. Conclusion 115
References 115

1. Introduction

Many mechanisms and functions in networks are greatly affected by a fraction of nodes, which are usually qualified as *important*. A node is *important* if its failure or malicious behavior significantly degrades network performance. The issue of identifying the most important nodes in networks has long been the focus of an intensive amount of research. In the literature, these nodes appear under various names, depending on their role in the network, such as: most influential nodes [1], most vital nodes [2], most k-mediator nodes [3], key-player nodes [4]. With respect to network connectivity, the most important nodes are often known as *Critical Nodes*.

Therefore, critical nodes of a network are those the removal of which significantly degrades network connectivity. The *Critical Node Detection Problem* (*CNDP*) is the optimization problem that consists in finding the set of these nodes. In other words, the *CNDP* consists in finding the set of nodes whose deletion leads to achieving a certain objective related to making the network disconnected.

As a network can take different forms after disconnection, *node criticality* depends on how the network is disconnected once the node has been deleted, which depends, in turn, on the objective of the application considered. Thus, a node that is critical for some purposes or considerations may not be critical for others. Considering the network in Fig. 1, if we ask for the set of two nodes, the removal of which maximizes the number of connected components, then the optimal solution is to delete nodes $\{a, d\}$, which generates seven components. While if we consider the case where we aim at minimizing the set of nodes, the removal of which constrains the cardinality of each connected components to four nodes at most, the optimal solution is then to delete the nodes $\{b, c\}$. In both cases, we ask for the same thing: to delete nodes from the graph, but for different objectives. These objectives depend on the application at hand. For instance, the last case is relevant when inheriting the spreading of complex contagions[1] where more than four interactions are needed to acquire the contagion. In fact, if we assume that the graph in Fig. 1 corresponds to a social network, then vaccinating critical nodes $\{b, c\}$ (corresponding to individuals) allows the network to be partitioned on communities of four susceptible individuals each, at most (see Fig. 1). Hence either all individuals in the community are infected or not, there is no spread of the contagion as there is no chance of having more than four interactions with infected individuals.

Therefore, we can define critical nodes as those the deletion of which disconnects the network according to some predefined connectivity metrics, such as: maximizing the number of connected components, minimizing pairwise connectivity in the network, minimizing the largest component size, etc.

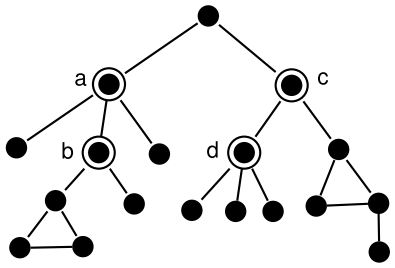

Fig. 1. Identifying critical nodes in a network with respect to the objective of the application considered.

Graph connectivity has long been considered as the measure of network robustness, and assessing how well the graph is connected or, on the contrary, stating how much effort (by deleting nodes or edges) is required to disconnect it has been extensively studied in the literature. This problem is known as the edge- or vertex-connectivity problem. For a detailed review, we refer the reader to [5]. Generally, deleting nodes (or edges) from a graph in order to obtain a disconnected graph with some specific properties is one of the oldest issues in graph theory. For problems based on the deletion of edges, a variety of variants have been defined and studied in the literature, such as the graph partitioning problem [6–8], the minimum k-cut problem [9,10], the multicut problem [11–13], the multiway cut problem [14–16], the multi-multiway cut problem [17], etc. For problems based on the deletion of nodes, which include the problem we are reviewing in this paper, we ask for deleting nodes rather than edges. Many variants have been studied in the literature, including:

- *The vertex separator problem.* Given a graph $G = (V, E)$ and an integer k, this problem aims at partitioning G into three subsets of nodes A, B and C such that $|C|$ is minimum, A and B are disconnected and $max(|A|, |B|) < k$. In other words, it seeks to find the minimum set of nodes $C \subseteq V$, the deletion of which partitions the graph into two bounded subsets A and B of at most k nodes. The problem is NP-hard even on 3-bounded degree graphs[2] [18], and it has been studied on general graphs [18], planar graphs [19,20], using polyhedral approach [21,22], metaheuristic algorithms [23], etc.
- *The multi-terminal vertex separator problem.* Given a graph $G = (V, E)$ and k terminal nodes, this problem consists in finding a subset $S \subseteq V$ of non-terminal nodes of minimum weight, the deletion of which generates k components, each

[1] This is a kind of contagion where an individual in the network acquires the contagion through interaction with $n > 1$ other affected individuals (n is an integer).

[2] A graph G is k-bounded degree if the degree of each node is less than or equal to k.

one contains exactly one terminal. This problem is also known as *the vertex multi-terminal cut problem*, and it is at least as hard as the edge version [15]. In the literature, different studies have been carried out considering this problem [24,15,25]

- *The vertex multicut problem.* This problem is the vertex version of the multicut problem [11]. It consists in, given a graph $G = (V, E)$, a set of s terminal-pairs and an integer k, finding a set of at most k nodes, the deletion of which disconnects the nodes in each terminal-pair. This problem was defined on two versions depending on whether the removal of terminal nodes is allowed (the restricted version) or not (the unrestricted version). The problem is NP-hard on bounded-degree trees [26], and different results were obtained for some classes of graphs, namely split, cobipartite and permutation graphs [27], trees and complete graphs [27], interval graphs [28] and bounded treewidth graphs [26,28]. Moreover the fixed-parameter complexity of the problem was explored in [29].
- *The minimum Vertex Cover problem (MVC).* Given a graph $G = (V, E)$, the *MVC* problem consists in finding a minimum set of nodes $A \subseteq V$ such that $G[V \setminus A]$ is an independent set. Thus, the *MVC* can be stated as follows: find the minimum subset of nodes, the deletion of which results in a set of connected components of one node each, at most. This problem is a classical NP-complete problem in graph theory, and it is one of Karp's 21 problems. It has been extensively studied in the literature [30].

All these problems, including the *CNDP*, are variants of the well-known class of problems called Node-deletion problems [31,32], and many results arisen from studying these problems exist in the literature. But only recently the *CNDP* was reconsidered by Arulselvan et al. (2009) [33] through a detailed study of the first variant called *CNP*, for *Critical Node Problem*. Since then the problem has received much attention, and several variants have been developed, with many intensive studies have been carried out to deal with each one. The purpose of this paper is to give a structured overview of recent results that mostly appeared subsequent to the work of Arulselvan et al. (2009) [33].

1.1. Problem relevance

Identifying critical nodes is an efficient way to analyze and apprehend the properties, structures, and functions of networks. Indeed, this facilitates network control whether the objective is to keep or to destroy it, since these nodes are those which maintain its cohesiveness and the removal of which significantly degrades its connectivity. Therefore, identifying critical nodes is of prime importance and has many applications in several domains, including computational biology [34,35], network vulnerability assessment [36–39], network immunization [33,40,41], etc. Section 7 gives more details about possible applications of the *CNDP* in different areas.

The three major factors that make the *CNDP* an important parameter are the following:

- Almost all network applications are usually designed to be run in a connected environment. The *CNDP* is the problem that determines the nodes whose preserving provides such an environment, and the removal of which disrupts it. Thus, finding these nodes is very useful for studying applications before and after design.
- The *CNDP* is a double-edged parameter. In fact, it can be used for offensive or defensive purposes, depending on the objective of the application at hand. For instance, on a computer network, critical nodes are those mainly immunized against virus attacks (defensive), or those primarily targeted to destroy an opponent network (offensive). Then, once identified, critical nodes may be the focus of protection and defensive monitoring for positive application purposes, or the focus of attacks and offensive attempts for negative purposes.
- The *CNDP* is useful for one of the most interesting issues, namely application robustness and security analysis. Indeed, the design of network applications (such as routing protocols) is often based on the selection of a kernel set of nodes, such as the maximum independent set or dominating set, and hence application security is proportional to the criticality of the selected kernel. Based on the hypothesis that the more critical nodes we have in the kernel, the more the application is vulnerable, and conversely, the fewer critical nodes we have, the more the application is robust, the *CNDP* is worth taking into account when designing secured network applications.

As partitioning networks can be done by deleting nodes or links, and as the link-deletion based problems have been extensively studied in the literature, we may wonder about the importance of studying problems based on node deletion. It should be noted that there are many situations where removing nodes makes more sense than removing links. This is the case, for example, when dealing with a virus spreading in the society, where we aim at stopping its propagation. In fact, nodes in the social network are people and links are social relationships, and since removing connections between individuals is usually difficult and may not be possible, we can instead vaccinate some individuals against contaminations in such a way that the spreading will be inhibited (here, the vaccination of a node is equivalent to removing it from the network).

1.2. Aims and motivations

The specific motivations behind developing such a review are summarized below. First, the importance of the problem, as it is a fundamental problem that has major applications in many areas: hence creation of a reference material with key results is essential. Second, the large number of results arising from the studies tackling this problem: thus it is extremely useful to summarize and classify these results in the same work.

To the best of our knowledge, this is the first survey conducted on the *Critical Node Detection problem*, and we are aware of any prior work. However, as this problem has become quite popular and has been studied in different fields including graph theory, network analysis, network vulnerability assessment, etc., leading to a variety of publications, we do not pretend to be able to give a comprehensive survey of all methods in all fields, but rather review recent theoretical results that mostly published further to the work of Arulselvan et al. (2009) [33]. Moreover, we are primarily focused on the combinatorial aspect of the problem. In other words, our main objective is to review, classify and discuss several recent advances and results obtained in the literature for each variant, including theoretical complexity, exact algorithms, approximation schemes and heuristic algorithms, as well as deduce and prove new results by establishing relationships between different variants.

1.3. Outline of the paper

The rest of the paper is structured as follows. We start with providing the basic concepts used in this survey in Section 2. In Section 3, we give a general formulation for the *CNDP* and study its complexity on general graphs. *Section 4* presents a detailed discussion of the general greedy approach for solving the problem.

In Section 5, we classify different variants of the *CNDP* into two main classes, and then detail each variant by reviewing different complexity analyses and solving results presented in the literature. In Section 6, we conduct a discussion about different approaches presented for solving the *CNDP* and thus present some ideas for future considerations. Different applications of the problem in several areas are reviewed in Section 7. The paper ends with a conclusion.

2. Definitions and notations

In this section, we introduce the necessary terminology used in the rest of the survey. We provide some of the basic definitions of graph theory, computational complexity and approximation algorithms. Readers not familiar with these topics are invited to refer to [42].

2.1. Graph theory

Throughout this paper, all graphs considered are finite. A graph $G = (V, E)$ consists of two finite sets: the set of nodes V, where $|V| = n$, and the set of edges $E \subseteq V \times V$, where $|E| = m$. Nodes and edges of graph G may have, respectively, general nonnegative weights and costs, and hence a weight $w_i \geq 0$ (resp. a cost $c_{vu} \geq 0$) may be associated with each node $v \in V$ (resp. edge $uv \in E$).

For each node $v \in V$, $N(v)$ denotes the neighborhood set of v, where $N(v) = \{u \in V | \{u, v\} \in E\}$. Similarly, for a subset $S \subseteq V$, $N(S) = \{v \in V | v \in N(u) \text{ for some } u \in S\}$ is the neighborhood set of S. Also, we use $G[V \setminus S] = (V \setminus S, E(V \setminus S))$, where $E(V \setminus S) = \{uv \in E | u, v \in V \setminus S\}$, to denote the subgraph of G induced by $V \setminus S$. For a connected component h in G, we denote σ_h its cardinality.

A *path* in G is a sequence of nodes $\{v_1, v_2, \ldots, v_k\}$ such that each pair (v_i, v_{i+1}) is an edge in E. G is said to be *connected* if there is a path between any two distinct nodes. Otherwise, graph G is *disconnected*. The pairwise connectivity of a graph is defined as the number of connected node pairs. For each pair $(u, v) \in V \times V$, the pairwise connectivity $p(u, v)$ is quantified as follows:

$$p(u, v) = \begin{cases} 1, & \text{if } u \text{ and } v \text{ are connected,} \\ 0, & \text{otherwise.} \end{cases}$$

In the following, we present some graph classes considered in this survey.

Interval graphs. An *interval graph* $G(V, E)$ is a graph whose nodes correspond to a set of intervals on the real line such that two nodes are adjacent in G if and only if the corresponding intervals intersect. An interval graph is said to be proper if and only if no interval is properly contained in another. These classes of graphs are very useful for modeling real-world problems and have many applications in real life [43–45].

Series–parallel graphs. A graph G is a *series–parallel graph* if it is constructed by a sequence of *series* and *parallel* compositions starting from a set of single-edge graphs [46]. Given two two-terminal graphs[3] $G_{s_1}^{t_1}$ and $G_{s_2}^{t_2}$, a *series* (resp. *parallel*) *composition* is the operation that creates a new two-terminal graph by merging t_1 and s_2 (resp. s_1 and s_2, and t_1 and t_2).

Trees. A *tree* $T(V, E)$ is a graph with no cycle, where any two nodes are connected by exactly one path. If a node is designated as root, the tree is called a rooted tree. The *tree traversal* is the process consisting in visiting all nodes of the tree. A *post-order traversal* of a tree is the order of traversal where the root node is visited last. Thus, we first traverse the left subtree, then the right subtree and finally the root node.

Regular graphs. A graph G is a *k-regular graph* if each node of G has a degree equal to k. If $k = 3$, G is also called a *cubic graph*. However, G is called a *k-degree bounded graph* if the degree of each node is less than or equal to k.

Planar graphs. A graph is *planar* if it can be drawn in a plane without edges crossing, *i.e.,* it can be drawn in such a way that no edges cross each other, and the only intersections of edges are at their endpoints. In other words, a plane graph is the graph that can be embedded in the plane.

Bipartite and split graphs. A *bipartite graph* $G(V, E)$ is a graph for which the set of nodes V can be divided into two disjoint subsets V_1 and V_2 where each one forms an independent set of G, and such that every edge $e \in E$ has one endpoint in V_1 and the other endpoint in V_2. However, a *split graph* is a graph $G(V, E)$ for which the set of nodes V can be divided into two subsets V_1 and V_2, *i.e.,* $V = V_1 \cup V_2$, such that V_1 is a clique and V_2 is an independent set.

Interdependent-power network. An interdependent-power network consists of two graphs representing, respectively, the power network and the communication network, with a set of interdependencies (edges) between them. Such network is characterized by cascading failures due to the interdependency of the two graphs.

k-hole graphs. A graph G is said to be *k-hole* if it contains k holes, where a *hole* is a set of nodes $\{v_0, \ldots, v_p\}$ such that an edge exists between v_i and v_j $(i < j)$ if and only if $i = j - 1$ or $i = 1$ and $j = p$.

Power-law graphs. A *power-law graph* is a graph for which the number of nodes of degree k is proportional to $k^{-\beta}$, where β is a fixed value $\beta > 1$.

Unit-disk graphs. A *unit-disk* graph is a graph $G(V, E)$ whose nodes correspond to a set of equal-sized circles on the plane such that two nodes are adjacent in G if and only if their corresponding circles intersect. This class of graphs has many applications and is mainly used to model the topology of ad-hoc wireless communication networks.

Given a graph $G = (V, E)$, a *tree decomposition* of G (also called clique tree or join tree) is a mapping of G into a tree, *i.e.,* it is a tree whose nodes are subsets of nodes of G formed as follows [47]: *(i)* the union of subsets of nodes forms the set of nodes of G, *(ii)* for each edge ij in G, there is a node in the tree containing both i and j, *(iii)* for each node v in G, the nodes in the tree containing v form a connected subtree. Given a tree decomposition of a graph G, the *treewidth* of G, denoted T_w, is the minimum integer c such that there exists a tree decomposition of G where the size of the largest node set is $c + 1$. Note that the tree decomposition of a graph is generally not unique. G is called a *graph with bounded treewidth* if its treewidth c is a fixed constant. A detailed list of graphs with bounded treewidth is provided in [48], including trees, outerplanar graphs, series-parallel graphs, chordal graphs and circular arc graphs with maximum clique size, respectively, 4 and 5. Note that many NP-hard problems can be solved efficiently in polynomial, and often in linear, time on the class of graphs with bounded treewidth (this is the case for example of the Maximum Independent Set problem [49]). The solving algorithms usually use the dynamic programming approach as follows. First, the problem is solved on each node set of the tree decomposition and then, using the computed solutions, an answer to the problem on the whole graph is constructed in a bottom-up order. Generally the algorithm is exponential only in the size of the node set (treewidth) and does not depend on the size of the graph, which is why we look for graph treewidth. A more detailed discussion on different characterizations and applications of tree decomposition for solving NP-hard problem can be found in [50].

[3] A *two-terminal* graph G_s^t is a graph that has a source node s and a target node t.

2.2. Computational complexity

Several search problems are simple and can be solved efficiently using a polynomial-time algorithm, *i.e.,* an algorithm of complexity $O(n^k)$, where n is the problem size and k is some constant that is independent of n. These problems are said to be in class P. For some other important problems, there are no known polynomial-time algorithms. However, these may still have a polynomial-time verification algorithm that can check whether or not a given solution provides a feasible solution to the problem. These problems (with polynomial-time verification algorithms) form the NP class. We note that P is a subclass of NP.

Furthermore, the hardest problems in the NP class is called NP-complete problems. NP-completeness is applied to the realm of decision problems, which are the problems for which the answer is either "yes" or "no" (they are also called yes or no problem). We say that a decision problem is NP-complete if: *(i)* it is in NP, *i.e.,* it has a polynomial-time verification algorithm, and *(ii)* all problems in NP are reducible to it in polynomial time. A decision problem X is polynomially reducible to a decision problem Y, denoted $X \leq_p Y$, if there exists a polynomial-time reduction A such that, for any instance $I \in X$ we have $A(I) \in Y$. If only the second condition holds, then X is said to be NP-hard.

We note that any *optimization* problem, *i.e.,* a problem involving minimization or maximization of the output value, can be easily formulated as a decision problem. In this survey, we will mainly discuss NP-complete problems, so for more information on NP-completeness and complexity classes, we refer the reader to the work by Garey and Johnson [30].

Parameterized Complexity. For a decision problem with input instance I, the corresponding parameterized problem consists of the input instance I and an additional part called *parameters*. The complexity of the problem is thus measured as a function of those parameters. So given a problem instance I with input size n and a parameter k, the objective of the parameterized complexity is to develop an algorithm with running time $f(k)n^{O(1)}$ where $f()$ is a function only depending on k, *i.e.,* an algorithm which is exponential only in the size of the fixed parameter k, and polynomial in the size of the input n. A problem that can be solved by such an algorithm is said to be a Fixed-Parameter Tractable (FPT), while an algorithm with such a running time is called an FPT algorithm. The theory of parameterized complexity was first developed by Downey et al. [51] and, for more details, we refer the reader to this work [52].

Kernelization. The kernelization (data reduction or pre-processing) technique is a method for reducing (but not necessarily solving) the given problem instance to an equivalent "smaller sized" instance in time polynomial in the input size. A slower exact algorithm can then be run on this smaller instance. Thus, given a parameterized problem instance I and a parameter k, the goal of kernelization is to design polynomial-time algorithms which take as input the instance (I, k) and return an instance (I', k') such that $|I'| \leq h(k)$ and $k' \leq g(k)$, for some computable functions $h()$ and $g()$. The returned instance is said to be the *kernel*.

2.3. Approximation algorithms

For some applications, finding the optimal solution to the intractable problem considered may not be a necessity, and a near-optimal solution will suffice. In these cases, heuristics and approximation algorithms are useful. They mainly form the practical approach for solving large instances of NP-complete problems.

Approximation algorithms often return a solution to a combinatorial optimization problem that is provably close to the optimal, *i.e.,* the returned solution differs no more than a fixed factor from the optimal solution. However, heuristics may or may not find a good solution, but in any case, they give reasonable estimates for which no proven guarantees exist.

Thus, the approximated solution returned by approximation algorithms is not very far from the optimal solution. To know how far it is from the optimal solution, we normally need to provide a bound or a factor α of the optimum. α is called the *approximation factor*: thus we say that the approximation algorithm finds a solution within a factor α of the optimum solution, *i.e.,* the approximated solution is at most α times the optimum solution for a minimization problem, and at least α times the optimum for a maximization problem. We have $\alpha > 1$ (resp. $\alpha < 1$) if the problem considered is a minimization (resp. maximization) problem. We call the *approximation ratio*, the value of the factor α over all instances of the problem.

Note that the value α can be viewed as the quality measurement of the approximation algorithm. The closer α is to 1, the better the algorithm is. Formally, we have the following: let X be a minimization (resp. maximization) problem, and let $\alpha = 1 + \varepsilon$ (resp. $\alpha = 1 - \varepsilon$) where $\varepsilon > 0$. An algorithm A is called a α-approximation algorithm for the problem X, if for all instances I of X, A delivers a feasible solution with an objective value $A(I)$ such that $|A(I) - Opt(I)| = \varepsilon \cdot Opt(I)$.

Furthermore, if the α-approximation algorithm finds a solution within a constant multiple of the optimum (ε is constant), then it is called a *constant-factor approximation algorithm*. Otherwise, it is called an *approximation scheme* (the solution is approximated overall $0 < \varepsilon < 1$). If approximation scheme complexity is polynomial in the input size, it is called a polynomial-time approximation scheme, abbreviated to PTAS. If, in addition to polynomial-time complexity, the scheme is also polynomial in $1/\varepsilon$ then it is called a fully polynomial-time approximation scheme (FPTAS).

3. Identifying critical nodes in networks

In this section, we first introduce a general formulation for the *CNDP* taking into account different variants of the problem. We then discuss its complexity.

3.1. Problem formulation

How the network is disconnected once the critical nodes have been deleted is of great importance, since disconnecting it regardless of how it is disconnected is usually not efficient. Indeed, when the goal is for example to totally disconnect the network, if only boundary nodes[4] are disconnected, then even though the whole network is disconnected, network connectivity is slightly affected. This is true for scale-free networks [53,54].

Thus, determining which nodes are "*critical*" depends on the network structure that we look for after deleting the nodes. This depends, in turn, on the nature and the objective of the application in question. According to this application, different objectives can be considered: the aim may be to only maximize the number of connected components in the remaining network, or to minimize the largest component size, or a combination of both objectives, etc. The relevant question here is: *how can we distinguish critical nodes with respect to our target objective*? Using a connectivity metric to be satisfied (actually optimized) in the network once the nodes have been deleted provides a good solution to the problem. In fact, defining these metrics with respect to the objective to be achieved allows the resulting network structure to satisfy the target criteria and hence the critical nodes to be accurately distinguished.

[4] The boundary nodes are those located in the periphery of the network.

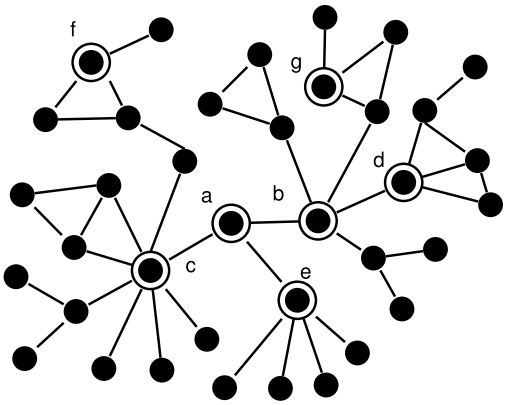

Fig. 2. A graph to illustrate optimal solutions for different *CNDP* variants.

Therefore, the *Critical Node Detection Problem* can be stated as the problem that aims to identify a set of nodes within a graph, the deletion of which minimizes or maximizes a predefined connectivity metric on the remaining graph. From now on, we denote σ as the predefined connectivity metric. This metric is usually modeled as an objective function to be optimized, which we denote $f(\sigma)$. Generally, the *CNDP* takes as input a graph $G = (V, E)$ and a connectivity metric σ, and returns as output a set of nodes $S \subseteq V$, the deletion of which optimizes $f(\sigma)$.

Critical Node Detection Problem

Input: *A graph* $G = (V, E)$ *and a connectivity metric* σ.

Output: *The set of nodes* $S \subseteq V$ *such that* $G[V \setminus S]$ *satisfies the metric* σ.

3.2. Connectivity metric importance

Defining the metric σ helps us accurately determine the concept of *criticality* with respect to the input problem. It describes how the graph structure must be disconnected and ensures the necessary specification of network connectivity once the nodes have been deleted. That allows the actual critical nodes to be identified. For instance, in an opponent environment, we suppose that at least t leaders must communicate to make a decision. If we aim to neutralize the adversary network by eliminating some individuals, then we have to disconnect the network into groups of less than t individuals. In such a case, a solution where all groups have $t - 1$ individuals is better than another where all individuals are isolated and only one group has $t + 1$ individuals, even though the network appears to be more disconnected in the second case.

According to the considered connectivity metric, different solutions for the *CNDP* can be generated for the same network. We illustrate this in the graph of Fig. 2. First, we assume that the number of critical nodes to be removed is given, say $|S| = k = 1$. Table 1 provides the optimal solution for four different connectivity metrics. Second, we assume that the number of nodes to be removed has to be minimized. Table 2 provides the solution that fits the connectivity metric, considering two different metrics.

Thus, it is easy to see that although in all cases our goal is to disconnect the graph by deleting nodes, the optimal solution depends on the considered metric. This has recently been proven by [55] through a comparative study considering different behaviors of an attacker who aims at disrupting network connectivity by neutralizing some nodes. They showed that although there are some nodes that are critical whatever the metric considered, criticality of the majority of nodes is mainly depends on the connectivity metric to be satisfied.

3.3. Problem complexity

As has already been mentioned, the *CNDP* is a variant of the well-known class of problems called Node-Deletion Problems [31,32]. These problems consist in deleting the smallest subset of nodes from a graph so that the remaining graph satisfies a predefined property π, such as: making the graph acyclic (known as the Feedback Vertex Set problem [56]), maximizing the length of the shortest path in the remaining graph (known as the Most Vital Nodes problem [57,58]), making the graph bipartite [59], chordal [60], planar [61], etc.

The computational complexity of these problems depends on the property π. In fact, Lewis and Yannakakis [32] proved that, for any *nontrivial hereditary* property π, it is NP-hard to compute the minimum number of nodes which must be deleted from a given graph so that the induced subgraph satisfies π (see Theorem 3.1 below). A property π is nontrivial if infinitely many graphs satisfy π and infinitely many graphs fail to satisfy it. It is also said to be hereditary on induced subgraphs if, in any graph satisfying π, every induced subgraph also satisfies π.

Theorem 3.1 (*[32]*). *Given a graph and a property* π. *If* π *is a "nontrivial" property and "hereditary" on an induced subgraph, then the node-deletion problem for* π *is NP-complete.*

In the *CNDP*, the predefined property π is related to making the remaining graph maximally disconnected according to a given connectivity measure. In the following sections, we see that all *CNDP* variants, for which the metric is nontrivial and hereditary, are NP-complete on general graphs [33,62,63,29,37,64], and some of them remain NP-complete even on some particular graph classes [65–68,46].

4. Solving CNDP

To solve hard problems, one of the most popular and effective approaches is to use greedy algorithms. These algorithms iteratively make the locally optimal choice with the hope of finding a global optimum. They often yield good solutions within a reasonable time. In what follows, we present a general greedy algorithm for solving the *CNDP*.

4.1. A general greedy approach for CNDP

Considering the greedy approach, a general algorithm for the *CNDP* can be stated as follows (see *Algorithm 1*). First, we select an appropriate measure from those used in the literature to identify the most important nodes, such as node centrality [69], and then we greedily select the node with the biggest value with respect to this measure. The idea behind selecting an appropriate measure is that solving the *CNDP* means to distinguishing critical nodes from non-critical nodes, and these measures are used to distinguish important nodes in networks.

Algorithm 1 *Greedy algorithm for solving CNDP*

1: **Input:** a graph $G = (V, E)$ and a connectivity metric σ.
2: **Output:** a set $S^* \subset V$ of k critical nodes.
3: Select a measure γ used to identify important nodes, e.g. *centrality*.
4: $S \leftarrow \{\}$
5: **while** $|S| < k$ **do**
6: $\quad i = arg\ max\{\gamma\}, \forall i \in V \setminus S$
7: $\quad S \leftarrow S \cup \{i\}$
8: **end while**
9: $S^* \leftarrow S$

However, while efficient, this approach often fails to yield an optimal solution for two main reasons:

Table 1
Optimal solutions for different *CNDP* variants where a single node is deleted.

The connectivity metric to be satisfied on $G[V \setminus S]$	Optimal solution (graph of Fig. 2)	The induced graph $G[V \setminus S]$
maximize the number of components	delete node $\{c\}$	generates 7 components
minimize pairwise connectivity	delete node $\{b\}$	impairs connectivity to 232 node pairs
minimize component size	delete node $\{a\}$	yields a largest component of 16 nodes
maximize the number of smallest components of *size* <2	delete node $\{e\}$	generates four components

Table 2
Optimal solutions for different *CNDP* variants where the aim is to minimize the number of nodes to be deleted.

total pairwise connectivity is <60	delete two nodes $\{b, c\}$	yields a pairwise connectivity $= 54$
the cardinality of each component is ≤ 5	delete four nodes $\{a, b, c, d\}$	component size at most 5

(i) If the selected measure is not rigorously related to network connectivity, *i.e.,* it is not designed to assess node importance in terms of connectivity. In such a case, the measure is not appropriate for distinguishing critical nodes. To illustrate this feature, we assume that the selected measure is degree centrality, and we consider the graph (a) in Fig. 3. We assume that we aim to maximize the number of connected components in the graph by deleting one node. Removing the most central node a has no effect on disconnecting graph, and the optimal solution is to delete node b, which has a lower centrality degree, resulting in three components. For more details, we refer the reader to Borgatti [4] (he called this feature *the goal issue*) and Shen et al. [64], where they illustrated the failure of node-centrality to identify critical nodes considering different connectivity metrics.

(ii) The *CNDP* asks for a set of nodes (together) and not individual nodes. In this case, critical nodes are those the deletion of which (all together) optimizes the connectivity metric considered in the *CNDP* variant. If a node is critical when considered alone, this does not necessarily imply that it remains critical when considered with others. In other words, the set of nodes which are critical when considered alone does not necessarily provide an optimal solution for the *CNDP*. We illustrate this concept using graph (b) of Fig. 3. We assume that we aim to maximize the number of connected components by deleting at most two nodes. We select cut-vertices [70] to be critical nodes. This seems to be an appropriate measure since the objective is to maximize the number of connected components, and cut-vertices form the connection between the components. However, the overall impact when removing a and b, which are non-cut vertices, is as good as removing c and b, which are two cut-vertices. Also the optimal solution is $\{d, e\}$, which yields three components, and contains a non-cut vertex e. This has also been detailed in [4].

In some cases, even if the selected measure is suitable, it may not be cost-effective in terms of time complexity, or not viable, particularly on large-scale networks, since many of these are computationally-expensive. This is the case of betweenness centrality [71,72] for example.

Despite the difficulty of the problem, many approaches have been explored in the literature to solve it, using: dynamic programming [65–68,46] and integer linear programming [33,62,73,36,74,64,37], to provide exact solutions. To provide approximated solutions with performance guarantee, a variety of methods have been considered, including heuristic algorithms [33,75,39,38,76], polynomial-time approximation algorithms [41,66], particularly rounding-based approximation approaches [77,37,78–80], stochastic search algorithms [62,81], fixed-parameter tractable algorithms [66,29], polynomial pseudo-approximation algorithms [63].

Recently, solving frameworks considering different connectivity metrics together have been developed. This is the case for [82], where the developed unifying integer programming framework takes into account four connectivity metrics. Similarly, authors in [83] presented an efficient evolutionary framework for solving different variants of the *CNDP*. The framework is potentially adaptable to deal with different variants of the *CNDP* considering different connectivity metrics.

5. Variants of CNDP

In this section, we present the variants of the *CNDP*, those that we rate as among the top fundamental variants of this problem. First, we begin by classifying them into two main classes, before reviewing, for each one, the recent important combinatorial results, as well as the algorithmic aspects proposed for solving it in different graph classes, if any.

5.1. Classification

As stated above, the *CNDP* consists in finding the set of nodes $S \subseteq V$, the deletion of which degrades network connectivity according to some predefined network-connectivity metrics σ. In the literature, this problem is handled according to different cases of $|S|$ and σ. The goal is usually whether to *optimize (minimize or maximize) the metric* σ, such that no more than k nodes are deleted ($|S| \leq k$), or to *minimize the set of deleted nodes*, such that the metric σ is bounded by a given threshold β. Then, we can provide a simple classification for different variants of the *CNDP* in two main classes, with respect to the objective, as follows (the two classes are complementary):

1. K-**vertex-CNDP**: given a graph $G = (V, E)$, a connectivity metric σ and an integer k, we aim to delete a *set of* k *nodes*, with the purpose of optimizing the objective function $f(\sigma)$.
2. β-**connectivity-CNDP**: given a graph $G = (V, E)$, a connectivity metric σ and an integer β, we aim to *bound the objective function* $f(\sigma)$ *to* β, with the purpose of minimizing the number of deleted nodes.

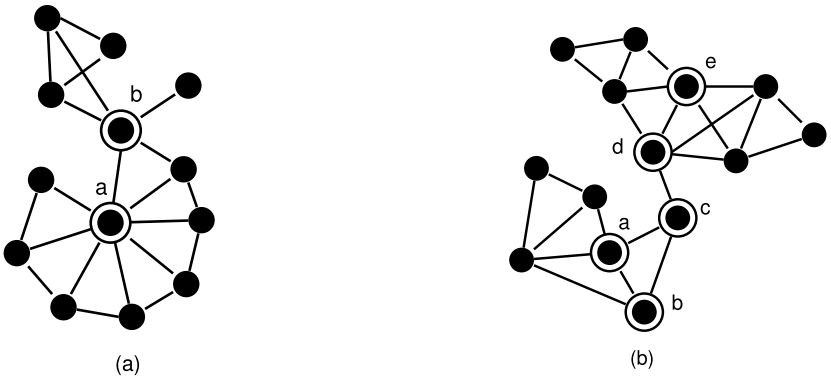

Fig. 3. (a) Maximizing the number of components using degree centrality measure, (b) Deleting cut-vertices as critical nodes.

The *K-vertex-CNDP* is relevant when we have information about the number of nodes to target but do not know how to optimally achieve our goal. For instance, given a terrorist network, we assume that we aim to break down communication between individuals by eliminating some of them, and that because of bounded resource capacity, we can neutralize only k persons. Then, we try to disconnect the network using the possible resources in order to achieve our goal. Now, considering the same terrorist network that we aim to totally neutralize this time, and we assume that at least t persons must communicate to make a decision. Then, to achieve our goal, we try to exterminate a set of individuals such that the network has a set of disconnected groups of less than t persons. Since exterminating these nodes usually has a cost, we seek for the minimum number. In such a case, where we do not know the number of nodes to target but have information about the network structure we need to obtain, the appropriate *CNDP* variant is the *β-connectivity-CNDP*.

Table 3 presents the list of variants considered in this survey, while Fig. 4 illustrates a classification of these variants with respect to the two main classes defined above, namely the *K-vertex CNDP* and the *β-connectivity CNDP*.

In what follows, we detail each variant of the *CNDP* starting with those of the *K-vertex CNDP* and then those of the *β-connectivity CNDP*. For each one, we first present different complexity results presented in the literature, followed by different approximation schemes, and then parameterized complexity results if any. We then describe different solving algorithms, starting with those proposed for solving the variant on general graphs, and then those considering the variant on different particular classes of graphs.

5.2. CNP — Critical node problem

Given a graph $G = (V, E)$ and an integer k, the *CNP* seeks to find a set $S \subseteq V$ of at most k nodes, the deletion of which minimizes pairwise connectivity in the remaining graph $G[V \setminus S]$. Minimizing pairwise connectivity simultaneously maximizes the number of connected components and minimizes cardinality variance among components in $G[V \setminus S]$. The objective can be formulated using the following function to be minimized [33]:

$$f(S) = \sum_{C_i \in G[V \setminus S]} \frac{\delta_i(\delta_i - 1)}{2} \tag{1}$$

Where $C_{i,i=1,\ldots,m}$ is the set of all connected components in $G[V \setminus S]$ once the k nodes have been deleted, and δ_i is the size of the component C_i. We note that determination of graph component sizes can be nicely computed in linear time using a Depth-first search algorithm [95].

Using the objective function in (1), the *CNP* formulation can be stated as follows:

Input: An undirected graph $G = (V, E)$ and an integer k.

Output: $\underset{S \subseteq V}{argmin}\, f(S) = \sum_{C_i \in G[V \setminus S]} \frac{\delta_i(\delta_i - 1)}{2}$, where $|S| \leq k$.

We note that minimizing $f()$ leads to a maximum number of components with minimum variance among their cardinalities, see [33] for a detailed proof. Therefore, we can easily see that the *CNP* is inherently a multi-objective optimization problem, while the above formulation is a single-objective version of the problem as the two objectives are merged in only one function. In the literature, all works dealing with this variant consider this single-objective formulation except for [92,96], where a detailed study of the multi-objective formulation has been conducted.

In the case where a nonnegative weight $w_j \geq 0$ and a nonnegative connection cost c_{ij} are associated with, respectively, each node $j \in V$ and each pair of distinct nodes $i, j \in V$, a generalization of the *CNP* can be stated as follows [67]:

$$\text{minimize } f(S) = \sum \{c_{ij} : i \text{ and } j \text{ are connected in } G[V \setminus S]\}$$
$$\text{subject to } \sum_{j \in S} w_j \leq k$$

In this case, the goal is to minimize the total connection cost, rather than the pairwise connectivity, in the induced graph, by deleting a set of nodes where the total weight, rather than the cardinality, is no more than k.

The *CNP* is the variant of the *CNDP* that has attracted much attention in the last few years. Its edge version belongs to the class of the so-called *network interdiction problems* [97]. These problems consist in finding any set of edges with a total cost of no more than a given budget, the deletion of which interdicts (hinders) the solving of a given optimization problem on the remaining graph. For instance, the *Shortest Path Interdiction Problem* consists in deleting a set of edges with total cost of no more than the given budget, so that the total weight of shortest paths between two given nodes u and v is maximized. Thus, the edge-version of the *CNP* can be defined as the problem that needs to delete a set of edges of no more than the given budget, so that pairwise connectivity in the remaining graph is minimized.

5.2.1. Complexity results

The recognition version of the *CNP* has been proven to be NP-complete on general graphs by Arulselvan et al. [33] through a polynomial reduction from the Maximum Independent Set problem, known to be NP-complete [30]. So we have:

Theorem 5.1 ([33]). *The Critical Node Problem is NP-complete on general graphs.*

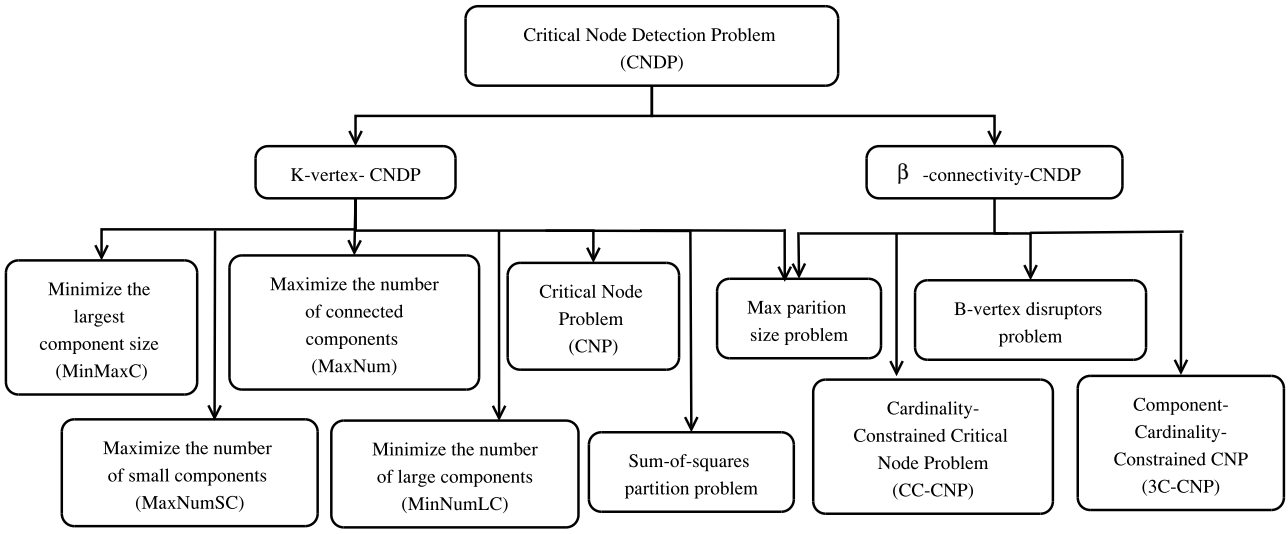

Fig. 4. Classification of different variants of the *CNDP* with respect to the two main classes, namely the *K-vertex CNDP* and the *β-connectivity CNDP*.

Table 3
The main variants of the CNDP considered in this survey.

Variant	Objective	Citation
CNP	Minimize pairwise connectivity by deleting k nodes (k is given as input)	[65,33,62,34,38,37,67] [84,85,82,78,79,86] [87–89,81,90,83] [77,91,92,73,76]
MaxNum	Maximize the number of connected components by deleting k nodes (k is given)	[46,64,66,29,83,82]
CC-CNP	Limit the maximal component size to a given bound by deleting the minimal set of nodes	[75,62,68,93,74,89] [85,83,80]
MinMaxC	Minimize the largest component size by deleting k nodes (k is given)	[46,64,39,83,82]
β-vertex disruptors problem	Bound pairwise connectivity to a given threshold by deleting the minimal set of nodes	[36,94,63,83]

Considering graphs with bounded degree, we prove the following new complexity result for the *CNP*.

Theorem 5.2. *The CNP is NP-complete on graphs of maximum degree* $\Delta = 4$.

Proof. The *CNP* has been proven to be NP-complete on general graphs [33]. Here, we show that it remains NP-complete on graphs of maximum degree $\Delta = 4$. For that, we define a polynomial reduction from the Minimum Vertex Cover problem (*MCV*), known to be NP-complete on cubic graphs [98], to the *CNP* on graphs of maximum degree 4, similarly as in the proof of *Theorem 2* in [68].

Given a *MVC* problem instance on a cubic graph $G = (V, E)$, let $G' = (V', E')$ be the graph obtained by attaching to each node of G, $v \in V$, a path of M nodes, such that the end node of the path coincides with v. Now, we consider the *CNP* on G', where we aim to determine whether pairwise connectivity in G' after deleting k nodes satisfies the following (the same inequality used in [33]):

$$\sum_{h \in \chi} \frac{\sigma_h(\sigma_h - 1)}{2} \leq \frac{kM(M-1)}{2} + \frac{M(n-k)(M+1)}{2} \quad (2)$$

The transformation can be carried out in polynomial time, and the graph G' has a maximum degree $\Delta = 4$. Now, we have to prove that the *MVC* has a solution on G if and only if the *CNP* has one on G', and vice-versa. This is exactly the same as what authors [33] did in the proof of *Theorem 1*, namely they proved that there is a one-to-one correspondence between the Maximum Independent Set Problem on G (rather than the *MVC* in our case) and the *CNP* on G'. So, for the rest of our proof, we refer the reader to the constructive proof of *Theorem 1* [33].

This completes the proof. ■

The *CNP* remains NP-complete even on particular classes of graphs, as can be deduced from the following theorem due to Shen et al. [37] and Addis et al. [65]:

Theorem 5.3 (*[65,37]*)**.** *The* CNP *is NP-complete on the following particular classes of graphs:*

1. *Split graph.*
2. *Bipartite graph and complement bipartite graph.*
3. *Unit-disk graph.*
4. *Power-law graph.*

For acyclic graphs, namely trees, Di Summa et al. [67] studied the weighted version of the *CNP* (the case of nonnegative connection costs and nonnegative node weights). They proved that the *CNP* is still NP-complete on trees where connections have nonnegative costs. This result has been proven through a reduction from the Multicut Tree Problem, known to be NP-complete [12]. Hence the following theorem.

Theorem 5.4 (*[67]*)**.** *The CNP is NP-complete on trees when general connection costs* ($c_{ij} > 0, \forall ij \in E$) *are specified even if the node weights are unit* ($w_i = 1, \forall i \in V$).

In this case, and despite the difficulty of the problem, authors [67] proposed an enumeration scheme with a time complexity within a factor of $O(1.618034^n)$ (*i.e.*, where $c_{ij} > 0$ and $w_i = 1$)

We can see that the optimal solution of the *CNP* is when the objective function $f()$ is minimized to a value 0, which generates an induced graph with $n - k$ isolated nodes, after deleting the k critical nodes. This is equivalent to find a vertex cover set of at most k nodes (or an independent set of at least $n - k$ nodes). Thus, solving the *CNP* in this case is equivalent to finding the minimum vertex cover set in the graph. Since approximating the Minimum Vertex-Cover Set problem is NP-hard [99], we obtain the following theorem due to Dinh et al. [63] and Addis et al. [65]:

Theorem 5.5 ([65,63]). *Unless P = NP, the CNP has no polynomial-time approximation scheme on general graphs.*

Addis et al. [65] generalized this result for all graphs on which solving the Maximum Independent Set problem is NP-hard, and hence introduced the following theorem.

Theorem 5.6 ([65]). *There is no polynomial-time approximation scheme for the CNP on planar graphs, cubic planar graphs, and graphs of bounded degree.*

An inapproximation ratio of the *CNP* on general graphs has been established by Shen et al. [37] through a reduction from the Maximum Clique Problem, which is NP-hard to approximate within $n^{1-\epsilon}$ [100]. — in [37], the *CNP* is called *CND*, for *Critical Node Disruptor*

Theorem 5.7 ([37]). *The CNP cannot be approximated on general graphs within a ratio better than $\Omega(\frac{n-k}{n\epsilon})$ for any $\epsilon < 1 - log_n 2$.*

The parameterized complexity aspects of the *CNP* has been only studied in [84]. We recall that the objective of parameterized complexity is to determine whether or not the problem is fixed parameter tractable (FPT) for given natural parameters. As noted above, the *CNP* with the objective function $f(S) = 0$ is equivalent to the Minimum Vertex Cover problem, and hence it is not FPT when parameterized by the pairwise connectivity parameter (denoted p hereafter). Also, it remains unlikely to be FPT with respect to the number of critical nodes (k) and the treewidth of the input graph (denoted T_w). A complete proof of these results can be found in [84].

Theorem 5.8 ([84]). *The CNP is W[1]-hard in general graphs, and therefore unlikely to be fixed parameter tractable, with respect to the following parameters:*

1. *The number of critical nodes (k).*
2. *The pairwise connectivity quantity (p).*
3. *The treewidth of the input graph (T_w).*

The result holds for the first parameterization (*i.e.*, considering the number of critical nodes k) even if the input graph is split, bipartite, or d-degenerate for $d \geq 2$.

Theorem 5.9 ([84]). *The CNP is not FPT with respect to the number of critical nodes k on split, bipartite, and d-degenerate (for $d \geq 2$) graphs.*

However, the *CNP* becomes FPT considering possible aggregations of the three parameters above, namely k, p, and T_w as shown in the following theorem.

Theorem 5.10 ([84]). *The CNP is fixed parameter tractable with respect to the following parameter aggregations: $k + p$, $p + T_w$ and $k + p + T_w$.*

Considering the pairwise connectivity of the critical nodes, denoted hereafter p_c, the *CNP* is FPT under this parameterization [84]. We note that this case is relevant when $k < p_c/2$ and $|C| \leq p_c/2$ for each component C of G, since if $k > p_c/2$ then removing any k nodes has the same effect, and if $|C| > p_c/2$ then removing one node from C involves the removal of at least p_c pairs of nodes. Hence, the following theorem holds.

Theorem 5.11 ([84]). *The CNP parameterized by the pairwise connectivity of the critical nodes is FPT, and it remains so with respect to: $p_c + k$, $p_c + T_w$, $p_c + k + T_w$.*

Also, the *CNP* has no polynomial kernel for the four parameters considered above, namely k, p, T_w and p_c, and for all their possible aggregations, except for the following, stated in the theorem below, due to Hermelin et al. [84] too.

Theorem 5.12 ([84]). *Considering the four parameters, namely k, p, p_c and T_w, and all their possible aggregations, the CNP admits a polynomial kernel only when parameterized by: $k+p$, $p+p_c$, $k+p+p_c$, $p + p_c + T_w$ and $k + p + p_c + Tw$.*

Table 4 summarizes the different *CNP* complexity results obtained on different classes of graphs.

5.2.2. Solving CNP

Before talking about solving the *CNP*, we give a brief analysis of its two complementary forms considered in the development of solving algorithms. Generally, the *CNP* can be seen as a problem that asks for:

(i) minimizing pairwise connectivity in the residual graph, *i.e.*, *minimize* $f(S) = \sum_{i \in G[V \setminus S]} \frac{\delta_i(\delta_i - 1)}{2}$, while deleting a set S of k nodes.
(ii) or, for a complementary form, maximizing the number of disconnected pairs of nodes in the residual graph, *i.e.*, *maximize* $f'(G, S) = \binom{n}{k} - \sum_{i \in G[V \setminus S]} \frac{\delta_i(\delta_i - 1)}{2}$, while deleting a set S of k nodes.

Hereafter, we denote the former as *form1* while the latter as *form2*. Although we may not notice, at first glance, much difference between the two forms, different approaches have been developed with respect to each form. In *form1* we focus on the remaining nodes, the pairwise connection of which has to be minimized, while in *form2* we focus on the nodes to be removed, the deletion of which maximize the number of disconnected pairs of nodes (*i.e.*, in *form1* we focus on the objective of the problem, while in *form2* we focus on the set of critical nodes)

In the following, we overview and classify different approaches investigated in the literature, in order to develop both exact and approximated solutions for the *CNP*.

(i) Exact approaches

Using Integer Linear Programming (ILP), different mathematical formulations of the *CNP* have been proposed. The first is a model with $O(n^2)$ variables and $O(n^3)$ constraints presented by Arulselvan et al. [33]. This formulation (see (3)–(7)) is described using two binary variables v_i and u_{ij}, where $v_i = 1$ if and only if the node $i \in V$ is deleted otherwise $v_i = 0$, and $u_{ij} = 1$ if and only if the edge (i, j) is in the residual graph otherwise $u_{ij} = 0$.

$$minimize \sum_{i,j \in V} u_{ij} \tag{3}$$

$$s.t. \sum_{i \in V} v_i \leq k \tag{4}$$

$$where : u_{ij} + v_i + v_j \geq 1, \forall (i, j) \in E \tag{5}$$

Table 4
CNP complexity results on different classes of graphs.

$\mathcal{CNDP}$ Variant	Metric	Graph class		Complexity	Citation
CNP	Minimize pairwise connectivity	General graphs		NP-complete	[33]
		Tree	$w_v = 1, c_{vu} = 1$ $w_v \geq 0, c_{vu} = 1$	Polynomial	[67]
			$w_v = 1, c_{vu} \geq 0$ $w_v \geq 1, c_{vu} \geq 0$	NP-complete	[67]
		Graph with bounded treewidth		Polynomial	[65]
		Split, Bipartite Complement Bipartite Unit-disk, Power-law graph		NP-complete	[65,37]

$$u_{ij} + u_{jk} + u_{ki} \neq 2, \forall i, j, k \in V \quad (6)$$

$$v_i, u_{ij} \in \{0, 1\}, \forall i, j \in V \quad (7)$$

Two other improved and detailed formulations have been presented by Di Summa et al. [73]. The first is an extended and enhanced formulation of that presented in [33], using a similar model to the Edge Deletion Problem model presented in [101]. This formulation has a non-polynomial number of constraints. It works in the branch-and-cut framework, and its relaxation can be solved in polynomial time (see [73] for a complete study). The second is a model based on quadratic programming reformulation of the *CNP* (see [73] too). Another ILP formulation considering the complete form of the *CNP* (*form2*) has also been presented by the same authors. In this formulation, the aim is to maximize the number of disconnected node pairs rather than minimize the number of connected pairs of nodes. In other words, they disconnect as many node pairs as possible in the residual graph, *i.e., maximize* $\sum_{i,j \in V}(1 - u_{ij})$ *s.t.* $\sum_{i \in V} v_i \leq k$.

These formulations [33,73] work only on small sparse networks, up to a few hundred nodes, due to the large number of constraints which is at least $O(n^3)$. An improved compact linear reformulation using only $\Theta(n^2)$ constraints has been developed by Veremyev et al. [85]. This formulation provides exact solutions for *CNP* on large networks, with up to 1200 nodes in a faster way, which was previously unfeasible using exact methods. Solving this formulation on large networks makes it possible to compare heuristic algorithms with exact approaches, which were previously compared either with other heuristic approaches or with exact approaches but only for small instances (this is the case for [33], where the proposed heuristic was compared with the ILP for small networks of at most 150 nodes). Recently, the same authors [82] developed a unifying integer programming framework for solving different *CNDP* variants, considering different connectivity metrics (including the *CNP*).

To provide approximated solutions for *CNP*, two approximation algorithms have been proposed in [78,79]. The first [78] is an algorithm based on the randomized rounding approach. It provides a $1/\theta$-approximation to the objective of the *CNP*, and a $1/(\theta - 1)$-approximation to the number of deleted nodes, where θ is the rounding threshold. The second [79] is a bicriteria approximation algorithm based on the region-growing approach. It provides a logarithmic approximation bound to the number of removed nodes, and ensures an optimal solution within a constant factor to the objective of the *CNP* (*i.e.,* it is a $O(1)$-approximation). We note that both approximation algorithms used the formation (3)–(7) above and are based on the idea that, given an ILP of a problem, one can rounding an optimal fractional solution of the ILP relaxation to an integral solution (see [102] for more details).

In a similar way, a hybrid iterative linear programming algorithm, also based on the rounding approach, has been developed in [37]. The algorithm used the ILP formulation above, and besides the iterative rounding of the solution obtained by solving the ILP, used a local search to enhance the solution. Its efficiency was examined by a simple comparison with the solution obtained by solving the ILP, and no approximation bound was provided.

(ii) Greedy approaches

To provide acceptable solutions for *CNP* within a reasonable time, a variety of algorithms have been developed using greedy approaches. The main idea of the proposed algorithms is as follows. First, note that there are two sets of nodes: the set of critical nodes S which must satisfy the constraint $|S| \leq k$, and the set of remaining nodes $V \setminus S$ which must have minimal pairwise connectivity. At the beginning, the two sets are initialized, and then we iteratively remove a node from one set and add it to the other one until the number of removed nodes satisfies $|S| \leq k$. With respect to the two forms of *CNP* cited above, two main algorithms have been presented.

Greedy1. (see *Algorithm 2*) According to *form1*, the algorithm starts with an initial solution $S = S_0$, which optimally achieves the objective, *i.e.,* minimizes pairwise connectivity in $G[V \setminus S]$, but generally does not satisfy $|S| \leq k$. Then, it iteratively selects nodes from S and adds them to $V \setminus S$ until $|S| = k$. At each iteration, the selected node to be added is the one that returns the minimum value for the pairwise connectivity of the formed graph $G[V \setminus S_0]$, *i.e., minimize* $f((V \setminus S) \cup \{i\})$. Finally, the k non-selected nodes are the critical ones.

Greedy2. (see *Algorithm 3*) According to *form2*, the algorithm starts with $S = \{\}$ and $V \setminus S = V$. Then, it iteratively selects nodes from $V \setminus S$ and adds them to S until $|S| = k$. At each iteration, the selected node to be added is the one that maximizes the number of disconnected pairs of nodes in $G[V \setminus S]$, *i.e., maximize* $f'((V \setminus S) \cup \{i\})$, where $f'(S) = \binom{n}{k} - \sum_{i \in G[V \setminus S]} \frac{\delta_i(\delta_i - 1)}{2}$. In other words, it iteratively removes from the graph rather than adds to it as in Greedy1, the node that returns the minimal value for (1). Finally, the k selected nodes are the critical ones.

Algorithm 2 *Greedy1*

Input: a graph $G = (V, E)$ and an integer k.
Output: a set $S \in V$ of k critical nodes.
$S_0 \leftarrow$ *initial solution.*
$(V \setminus S) \leftarrow V \setminus S_0$.
while $|S_0| > k$ **do**
  $i = arg\ minf((V \setminus S) \cup \{i\})$
  $S_0 \leftarrow S_0 \setminus \{i\}$
  $(V \setminus S) \leftarrow (V \setminus S) \cup \{i\}$
end while
$S \leftarrow S_0$

The *Greedy1* approach was first used by Arulselvan et al. [33]. The proposed algorithm begins by finding the Maximum Independent Set (MIS), assigned for $V \setminus S$, and hence the initial solution S_0

Algorithm 3 *Greedy2*

1: The same input, output and initialization as *Greedy1*.
2: **while** $|S_0| < k$ **do**
3: $i = arg\ max f'(S \cup \{i\})$
4: $(V \setminus S) \leftarrow (V \setminus S) \setminus \{i\}$
5: $S_0 \leftarrow S_0 \cup \{i\}$
6: **end while**
7: $S \leftarrow S_0$

is the vertex cover set of the graph G. It greedily adds a node to the MIS from S_0 until $|S_0| = k$. The added node is the one that returns the minimum value for the pairwise connectivity of the formed graph. The algorithm has a time complexity $O(n^2m)$. However, no approximation bound has been provided for solution quality, and its efficiency has been investigated only on small sparse networks, of at most 150 nodes.

An obvious drawback of this greedy method occurs when node v, which belongs to the optimal solution, is in the initial $(V \setminus S)$, and hence there is no possibility that v appears in any solution S.

To improve this approach (*Greedy1*), different local search methods were explored. In [33] authors used a 2-exchange local search method which proceeds as follows. For each pair of nodes i and j such that $i \in MIS$ and $j \notin MIS$, they swap the nodes, *i.e.,* $j \in MIS$ and $i \notin MIS$, and examine the change in the objective function (1). If it improves, then they keep the swap; otherwise, they undo the swap and continue to the next node pair. The exploration complexity in this method is $O(n^3)$ in the worst case when $k = |S| = |V|/2$. In [86] two more computationally efficient local search methods have been proposed. The two methods reduce the number of exchanges to $O(n^2)$ and yield the same best swap as the straightforward 2-exchange local search presented above. The first makes the swap for a pair of nodes, $i \notin S$ and $j \in V \setminus S$, which maximizes the disconnection in the graph (according to *form2*), while the second makes the swap which minimizes the increase in the objective function (1) (according to *form1*).

The *Greedy2* approach is considered in [76]. Authors proposed a linear-time algorithm with complexity $O(k(n + m))$ to identify critical nodes based on this approach. However, there is no proof of approximation quality for the obtained solution. This approach has a major weakness in that it misleads the algorithm to select nodes which are critical when considered alone but not when considered together (see Section 4).

A detailed study of the two greedy approaches, *Greedy1* and *Greedy2*, with different possible combinations is discussed in [87,88]. The developed combined greedy algorithms alternate the application of the two basic operations, *i.e.,* adding (resp. removing) nodes to $V \setminus S$ (resp. from S). The algorithms have been enhanced by a kind of local search operation, which consists in sequentially deleting/adding nodes from/to the remaining graph before returning a feasible solution in order to allow the algorithm to move into the unfeasible region, and hence explore the solution space. These hybrid algorithms are shown to be computationally efficient and better able to deal with large instances of the *CNP* than *Greedy1* and *Greedy2*.

In [89], a more sophisticated multi-start greedy algorithm has been developed. The proposed algorithm proceeds as follows. If the number of critical nodes to be removed is less than 5 (a fixed value), then an evaluation of all possible combinations is performed, and hence the optimal solution is returned. Otherwise, the algorithm primarily targets the larger connected components. At each iteration, it identifies the larger components as those whose size is greater than the average size, and then randomly removes nodes from them. The selected nodes to be removed are those which maximize the increase in function (1). If the solution is not improved after a removing operation, a partial restart is launched. The efficiency of the algorithm was evaluated through computational experiments using a set of real-world graphs, and the results were compared with exact solutions obtained using the ILP described in [82]. The distinguishing feature here is the comparison with exact solutions for real large networks, which was previously impossible since the ILP are addressed for small networks.

(iii) ***Metaheuristic approaches***

Considering stochastic search approaches, two advanced methods, namely simulated annealing and population-based incremental learning methods, have been explored [81] for solving the *CNP* on large networks (with up to a few thousand nodes). The efficiency of both methods has been investigated on random networks generated using different network models. It has been shown that the second approach outperforms the first one.

Moreover, two other metaheuristic approaches using different local search methods have been developed in [88]. The first is a general Variable Neighborhood Search (VNS) framework, based on the VNS scheme presented in [103]. The proposed framework considers different possible combinations of the two local search methods developed in [86] in order to improve *Greedy1*. The second metaheuristic is an Iterated Local Search (ILS) framework based on the ILS scheme discussed in [104]. We note that this scheme consists in fragmenting, at each iteration, the largest connected components in the residual graph in such a way that it simultaneously maximizes the number of components and minimizes the variance between their cardinalities. The efficiency of the two metaheuristics has been investigated using random networks generated using different network models.

In [90], a Greedy Randomized Adaptive Search Procedure (GRASP) with Path Relinking has also been proposed for solving the *CNP*. The proposed procedure has been shown to be a competitive method through a comparison with the previously proposed metaheuristic approaches, namely simulated annealing [81], population-based incremental learning [81] and VNS [88], using generated random networks. We recall that GRASP is a metaheuristic for finding approximate solutions to combinatorial optimization problems, and that Path-relinking is a good mechanism to combine with GRASP to achieve significant improvements in solution time and quality: see [105] for more details. Recently, the authors [106] proposed a hybridization of GRASP with a new version of the Path Relinking metaheuristic, namely the interior and the exterior path-relinking [107]. The competitiveness of the new scheme was demonstrated experimentally.

Recently, a solving approach of the *CNP* based on the *Memetic Algorithm* (MA) was provided in [108]. This algorithm can be seen as an extension of the traditional genetic algorithm that combines population-based search and local search techniques to reduce the likelihood of the premature convergence, and hence get a suitable balance between search intensification and diversification [109]. The efficiency of the proposed approach was evaluated on both synthetic and real-world networks, and compared with seven other algorithms presented in the literature, namely the two greedy algorithms Greedy3d and Greedy4d [87], the neighborhood search approach [86], the local search metaheuristic [88], the evolutionary based approach [83], the multi-start greedy algorithm [89] and the heuristic approach [110].

More interestingly, an efficient evolutionary framework for solving different variants of the *CNDP*, including the *CNP*, has been presented in [83]. The two main features distinguishing this framework are: *(i)* adaptability; the framework is potentially adaptable to deal with all variants of the *CNDP* considering different connectivity metrics, and *(ii)* efficiency; it embeds the two greedy approaches described above, see [87], within the two main reproduction and mutation operators in order to repair the solutions.

The role of the greedy operations here is to guide the search in the feasible solution space and provide good solutions.

Studying the *CNP* using a multi-objective approach has also been addressed in [92,96]. We can easily note that the *CNP* is inherently multi-objective, in the sense that it aims at simultaneously maximizing the number of connected components and minimizing the variance of their cardinalities. A multi-objective formulation of the problem has been proposed, and experimental results have been provided using different multi-objective evolutionary algorithms.[5] The authors evaluated the performance of the proposed approach using a set of random networks of at most 5000 nodes, generated using different network models. The results of different algorithms have been compared with each other in order to determine the best algorithm in terms of solution quality, and have also been compared with solutions available using the single-objective version in order to determine the relationship between the two versions. The comparison showed that the best solution to the multi-objective *CNP* does not necessarily lead to the best solution to the single-objective *CNP*.

(vi) *CNP on particular classes of graphs*

Considering the *CNP* on particular classes of graphs, different algorithms have been proposed. For acyclic graphs, namely trees, and especially for cases where all connections have a unit cost ($c_{ij} = 1$) (even if the node weights $w_j > 0, \forall j \in V$), Di Summa et al. [67] proposed a polynomial-time algorithm using the dynamic programming approach. The algorithm has a time complexity of $O(n^3k^2)$ if nodes have unit weights ($w_i = 1$), and $O(n^7)$ if nodes have arbitrary weights ($w_i > 0$). For the case where connections have nonnegative costs, proven to be NP-complete (see Theorem 5.4), Lalou et al. [68] developed a heuristic that exploits the exact algorithm developed for solving the *CC-CNP* on this case. The proposed heuristic provides an approached solution to the *CNP* without proof of guarantee on solution quality.

On graphs of bounded treewidth, even if the nodes have arbitrary weights, Addis et al. [65] presented a polynomial-time dynamic programming algorithm to solve the *CNP*. Hence, the proposed algorithm solves the problem on outerplanar graphs, series-parallel graphs and chordal graphs with maximum clique size (a detailed list is provided in [48]).

In [38], the *CNP* has been considered on dynamic networks with the purpose of assessing vulnerability in this kind of network. Authors proposed an adaptive algorithm to solve the problem (*adaptive* refers to the ability of the algorithm to identify and update the set of critical nodes, while taking into account the frequent changes occurring on dynamic networks). First, the algorithm proceeds to solve the integer linear program of the *CNP* using the hybrid algorithm presented in [37]. The returned solution serves as an initial set of critical nodes, which proceeds to update according to the events updating the networks by adding/ removing links or nodes. Despite its efficiency, the algorithm does not provide any performance guarantee on solution quality.

Similarly, to assess vulnerability on networks with uncertainty, Dinh et al. [77] considered the *CNP* on probabilistic graphs. They called it *pCNP*, for *probabilistic Critical Node Problem*. In probabilistic graphs, each edge has a value $p \in [0, 1]$ representing the probability that the edge exists. Authors [77] presented a mixed integer programming formulation for the problem, based on which they proposed a heuristic algorithm using a rounding technique to find critical nodes. To deal with the intractability of computing pairwise connectivity values on probabilistic graphs, they proposed a fully polynomial-time randomized approximation scheme (FPRAS). The FPRAS is a (ϵ, σ)-approximation for estimating the pairwise connectivity value, and runs in $O(mn^4\epsilon^{-3})$ times, where $\epsilon, \sigma > 0$. We note that a (ϵ, σ)-approximation method returns a solution to within a relative error of no more than ϵ, with a probability of at least $(1 - \sigma)$. This approximation is said to be a FPRAS if its running time is polynomial in $1/\epsilon$, $log(1/\sigma)$ and the size of the problem. Since the *CNP* is a particular case of *pCNP* when all edge probabilities $p = 1$, then for cases where the *CNP* is NP-complete, the *pCNP* remains NP-complete too.

A general version of the *CNP*, named the *Cascading Vulnerability Node Detection* problem (*CVND*), was defined in [91], for also studying network vulnerability with cascading failures. In cascading failures, a node fails (is removed) if a given number of its neighbors fail. This number is a threshold given as input for each node. Thus, in *CVND* we look for identifying the set of initial failure nodes (considered as critical nodes) which maximally disconnects the graph after d-hop cascading failures, where d is given as an input parameter. Authors [91] presented an integer linear programming formulation for the problem, and showed that it is NP-hard to approximate within a factor $\Omega(\frac{(1+\frac{d}{n^{1-\epsilon}-1})^2(n-k)}{n^{\epsilon}})$, for any $\epsilon < 1 - log_n 2$, through a reduction from the Maximum Clique Problem. Since the *CNP* is a special case of *CVND* when $d = 0$, it is easy deduce its NP-completeness on graph classes on which the *CNP* is NP-complete, namely: Power-law graphs, unit-disk graphs, split graphs, bipartite and complement bipartite graphs, etc. (see Section 5.2).

Also, a generalization of the *CNP*, called the *Distance Critical Node problem* (*D-CNP*), was defined and studied in [112]. This variant is based on the distance between nodes, instead of just either or not a path exists between them. Thus, the objective of the *D-CNP* is to delete a subset of nodes such that pairwise distances between nodes in the residual graph is minimized. The pairwise distance between a pair of nodes is evaluated by the length of the shortest path between them. In [112], an Integer Linear Programming (ILP) formulation was developed, based on which an exact algorithm was also proposed for solving the problem on large instances. The general case where costs are associated with edges was also considered, and extensive computational experiments were conducted on both synthetic and real-world networks to evaluate performance of the proposed approach. In [113,114], the problem was studied in more details. Indeed, based on the fact that the more distant the nodes, the lower their connectivity value, authors proposed three classes of the problem according to specific distance functions (for each version, the connectivity value is multiplied by a distance-based penalty). The problem was then analyzed on paths, trees and series–parallel graphs considering different cases according to node weights and connection costs, which can be general nonnegative or unit. For polynomially solvable cases, a dynamic programming algorithm was provided, otherwise proofs of NP-completeness were established and pseudo-polynomial algorithms were developed.

All previously presented studies, concerning both complexity and solving results, considered the *CNP* on undirected graphs. The only work that considered the problem on directed graphs is [115]. The authors presented a linear-time algorithm of complexity $O(m + n)$, where n is the number of nodes and m the number of edges, for case where $k = 1$. They then developed a heuristic for the general case by iteratively removing the most critical node, identified according to the first algorithm, until they have k nodes removed. The efficiency of the proposed heuristic, in terms of running time and solution quality, was analyzed experimentally and compared to other heuristics.

5.3. MaxNum — Maximize the number of connected components

In the *CNP* variant, we aim simultaneously to maximize the number of connected components and minimize the variance in their cardinalities. One can look for only maximizing the number

[5] For more details about Evolutionary Algorithms dealing with multi-objective optimization problems, see [111].

of connected components as proposed by Shen et al. [46,64]. They introduced a new variant, namely *MaxNum*, for *Max* imize the *Num* ber of connected components. The variant requires the deletion of a determined number of nodes from a graph in order to disconnect it as far as possible by maximizing the number of components. It can be formulated as follows:

Input: an undirected graph $G = (V, E)$ and an integer k.
Output: a set of nodes $S \subseteq V$, where $|S| \leq k$ and the number of connected components ρ in $G[V \setminus S]$ is maximized.

If $\rho = 1$, there is no fragmentation. Maximum fragmentation occurs when every node is isolated, creating as many components as nodes (*i.e.,* $|V| - k$ connected components). We can note the correspondence between *MaxNum* and the Maximum Independent Set problem (*MIS*). Indeed, considering a solution S of *MIS* in a graph $G = (V, E)$, S is also a solution for *MaxNum* on G if $k \geq |V \setminus S|$. Thus, solving *MaxNum* for $k \geq |V \setminus S|$ is equivalent to finding a MIS for the graph. Similarly, the correspondence between *MaxNum* and its edge version, known as the Minimum k-cut problem, can be defined as follows. Given an instance of *MaxNum*, we replace each edge by a node connected to the end-nodes of the edge. We note that the Minimum k-cut problem asks for deleting edges, rather than nodes, with the purpose of maximizing the number of connected components in the induced graph.

5.3.1. Complexity results

Although the considered metric (*i.e.,* the number of components) seems to be easier than the pairwise connectivity, the *MaxNum* variant remains NP-hard on general graphs. This result has been proven in [64] and [66] through a reduction from, respectively, the Maximum Independent Set problem [30] and the Minimum k-cut problem [9].

Theorem 5.13 (*[66,64]*)**.** *MaxNum is NP-complete on general graphs.*

MaxNum is still NP-complete even on particular classes of graphs. First, considering 3-regular planar graphs, the NP-completeness of the problem was established in [66] through a reduction from the Maximum Independent Set problem [30] known to be NP-complete on this class of graphs.

Theorem 5.14 (*[66]*)**.** *MaxNum remains NP-complete on 3-regular planar graphs.*

Considering split graphs, we show below that *MaxNum* remains NP-complete on this class. First, we prove the following equivalence between *MaxNum* and the *CNP* on this class of graphs.

Theorem 5.15. *The MaxNum and the CNP are equivalent on split graphs.*

Proof. We recall that a graph $G(V, E)$ is a split graph if the set of nodes can be partitioned into two subsets V_1 and V_2, $V = V_1 \cup V_2$, where V_1 is an independent set and V_2 is a clique. Given a split graph $G = (V, E)$ and a set of nodes $S \subseteq V$, we can easily notice that $G[V \setminus S]$ always contains a nontrivial connected component and isolated nodes, if any.

We recall that the recognition versions of the *CNP* and the *MaxNum* seek to find a set of at most k, the deletion of which, respectively, minimizes pairwise connectivity and maximizes the number of components in the remaining graph. According to the value of k, two cases can be considered:

Case 1: $k \geq |N(V_1)|$. This is a trivial case, where the optimal solution, for both variants, is to delete the nodes of $N(V_1)$ and any $k - |N(V_1)|$ nodes from V_2. We then obtain a residual graph that has $|V_1|$ isolated nodes and a connected component of size $|V_2| - (k - |N(V_1)|$.

Case 2: $k < |N(V_1)|$. In this case, we consider an optimal solution for the *CNP* and try to prove that it is also an optimal solution for *MaxNum*, and vice versa. Given an optimal solution s^* for the *CNP* on a split graph G, this solution aims to find a set of nodes $S \subseteq V$ so that the nontrivial connected component of $G[V \setminus S]$ is as small as possible and the surviving isolated nodes of the independent set be as large as possible. Therefore, we note that for s^* only nodes in V_2 are removed from G (*i.e.,* $S \subseteq V_2$), and given any optimal solution for the *CNP*, that an equivalent solution satisfying this condition ($S \subseteq V_2$) can be constructed in polynomial time (for proof see [65]). On the other hand, to solve *MaxNum* we aim to obtain a maximal number of components in the residual graph. In doing so, we try to maximize the number of isolated nodes from V_1 once the critical nodes have been deleted. For this purpose, only nodes in V_2 are removed from G, which is exactly the solution s^*. Hence, the solution s^* is also the optimal solution of *MaxNum*.

Therefore, an optimal solution of one variant is an optimal solution of the other, and so the *CNP* and the *MaxNum* are equivalent. ■

According to Theorem 5.24 and since the *CNP* is NP-complete on split graphs [65], we have the following theorem (which is also proved by Berger et al. [66] through a reduction from the k-clique problem).

Theorem 5.16. *MaxNum remains NP-complete on split graphs.*

Using the correspondence between the *MIS* and *MaxNum* (see Section 5.3), and taking into account the difficulty of approximating *MIS*, authors in [66] established the following theorem.

Theorem 5.17 (*[66]*)**.** *It is NP-hard to approximate MaxNum, on general graphs, within a factor of* $n^{(1-\epsilon)}$*, for any* $\epsilon > 0$.

Also, using the inapproximability result of the k-subgraph problem on split graphs presented in [116], Berger et al. [66] proved the inapproximability of *MaxNum* on this class of graphs. Hence, the following theorem holds.

Theorem 5.18 (*[66]*)**.** *MaxNum does not admit a PTAS, under a reasonable complexity, on split graphs.*

As *MaxNum* is still NP-hard even when restricted to planar graphs (see Theorem 5.14), authors in [66] focused on developing a polynomial-time approximation scheme (PTAS) for the problem on this class, which results in the following theorem. We recall that a PTAS outputs an approximate solution of value of at least $(1 - \epsilon)$ times the optimum, and the running time is polynomial in the size of the problem.

Theorem 5.19 (*[66]*)**.** *MaxNum admits a PTAS on planar graphs, with time complexity* $O(nk^2 f(\epsilon))$*, where* $\epsilon > 0$ *and* f *is a function only depending on* ϵ.

From the parameterized complexity perspective, *MaxNum* has been examined by Marx [29] and Berger et al. [66]. We recall that a problem is said to be a fixed-parameter tractable if it can be solved by algorithms that are exponential only in the size of a fixed parameter, while polynomial in the size of the input. The first result was presented in [29], where authors showed that *MaxNum* is unlikely to be fixed parameter tractable with respect to two parameters, namely the number of deleted nodes k, and the number of connected components in the induced graph ρ. This results in the following theorem.

Theorem 5.20 (*[29]*)**.** *MaxNum is W [1]-hard (i.e., not fixed-parameter tractable), with respect to both parameters k and* ρ*, where k is the number of nodes to be deleted, and* ρ *is the number of connected components in the induced graph.*

Table 5
MaxNum complexity results on different classes of graphs.

$\mathcal{CNDP}$ Variant	Metric	Graph class		Complexity
MaxNum	Maximize the number of connected components	Trees	$w_i = 1$	Polynomial [66,64]
			$w_i > 1$	
		k-hole graphs, Series-Parallel graphs, graphs with bounded TW		
		General graphs Split graphs Cubic planar graphs		NP-complete [66,64]

This result holds even on split graphs with respect to the parameter k, as can be deduced from the following theorem due to Berger et al. [66].

Theorem 5.21 (*[66]*). *MaxNum remains W[1]-hard with parameter k (the number of nodes to be delete) even on split graphs.*

However, for planar graphs, the same authors [66] proposed a fixed-parameter tractable algorithm of complexity $O(nk^{O(k)})$, with respect to k.

Table 5 summarizes different complexity results obtained in the literature considering *MaxNum* on different classes of graphs.

5.3.2. Solving MaxNum

Although important, solving this variant does not receive the attention it requires from the research community with the exception of the following works. In [64], a mixed-integer program formulation (MIP) was presented. Authors also studied bounds and validated inequalities for the proposed formulation using the dynamic programming algorithm, presented in [46] for solving *MaxNum* on k-hole subgraphs of the original graph (a detailed study was presented in [64]). The efficiency of the MIP and the quality of the bounds have been investigated on small networks, of at most 50 nodes.

Considering *MaxNum* on particular graph classes, Shen et al. [46] developed polynomial-time algorithms, using the dynamic programming approach to solve the problem on trees, k-hole and series-parallel graphs, with complexity $O(n^3)$, $O(n^{3+k})$ and $O(n^3)$, respectively. They also showed that, for the case where a deletion cost is associated with each node and so the aim is to find a set of nodes of a total deletion cost of no more than a given bound, *MaxNum* remains polynomially solvable using the same dynamic program with a slight modification.

As well, on graphs of bounded treewidth, a polynomial-time dynamic programming algorithm with complexity $O(nk^2w^w)$, where $w-1$ is the treewidth, has been developed in [66] for solving *MaxNum*. Hence, the algorithm is able to solve *MaxNum* on outerplanar graphs, series-parallel graphs, chordal graphs with maximum clique size, etc. (a detailed list is provided in [48]). It has also been shown by Berger et al. [66] that *MaxNum* generalizes its edge version, namely the Minimum k-cut problem.

5.4. MinMaxC — Minimize the largest connected component size

In this variant, the connectivity metric considered is the size of the connected components that we aim to minimize. Indeed, *MinMaxC* needs to delete a set of at most k nodes, such that the size of the largest connected component in the remaining graph is minimized. It can be formulated as follows.

Input: an undirected graph $G = (V, E)$ and an integer k.

Output: $\underset{S\subseteq V}{argmin}\ \sigma_{h,h\in G[V\setminus S]}$, where $|S| \leq k$ and h is the largest connected component in $G[V \setminus S]$.

We note that if the purpose is to bound the cardinality of the largest connected component to 1, then solving *MinMaxC* is equivalent to finding the minimum vertex cover set of the graph. Also, it is equivalent to finding an optimal solution for the *CNP* on this graph.

In the case where each node $i \in V$ has both a nonnegative weight $w_i \geq 0$ and a nonnegative deletion cost, $c_i \geq 0$, a generalization of *MinMaxC* can be stated as follows:

$$\text{minimize } f(S) = \sum_{i\in h}\{c_i, \text{ where } h \text{ is the largest component in } G[V \setminus S]\}$$

$$\text{subject to } \sum_{j\in S} w_j \leq k.$$

In this case, two versions of weighted *MinMaxC* have been considered:

MinMaxCw1. In this version, we have $w_i = 1, \forall i \in V$. We thus seek to delete a set of bounded total node-deletion cost rather than a set of bounded cardinality.

MinMaxCw2. In this version, we have $w_i > 1$ and $c_i > 1, \forall i \in V$. The goal is thus to minimize the maximum-weighted component in the remaining graph by deleting a set of bounded total node-deletion cost.

A closely related problem to *MinMaxC* is the so-called *Vertex Integrity* problem [117]. The concept of *integrity* is defined as follows. Given an opponent network represented by a graph $G = (V, E)$, in order to disrupt it by destroying a set X of its elements, two main quantities are considered: *(i)* the number of elements ($|X|$) we would destroy, which is the cost of the disruption, and *(ii)* the size of the largest connected component (denoted $mG[V \setminus X]$) in the remaining network, which measures the success in disrupting the network. The opponent generally wants to make the two quantities simultaneously small. Thus, the problem aims to remove $|X|$ nodes from G such that the largest connected component in the remaining graph has a bounded number of nodes. The minimum sum of these two quantities is called *graph integrity*. Formally, the vertex integrity $I(G)$ of the graph G is defined as follows:

$$I(G) = min|X| + m(G - X)|X \subset V,$$

where $m(G-X)$ is the number of nodes in the largest connected component of $G-X$. This parameter is considered to be a measure of network vulnerability. For an overview of the results obtained for the problem of computing graph integrity, we refer the reader to [118,119].

5.4.1. Complexity results

Considering *MinMaxC* on general graphs, Shen et al. [64] proved its NP-completeness, according to *Theorem 1* in [120]. Hence the following theorem holds.

Theorem 5.22 (*[64]*). *MinMaxC is NP-complete on general graphs.*

MinMaxC was also considered on interdependent-power networks in [39], called *IPND*, for *Interdependent-Power Network Disruptor* problem. We recall that such a network is characterized by cascading failures due to the interdependency of the two graphs, namely the power network and the communication network. We note that a failed node can be seen as deleted from the graph. In this case, *MinMaxC* asks for the initial failed nodes (the critical nodes) which minimize the largest component size after the cascading failures. Authors in [39] showed that *MinMaxC* remains NP-complete on this kind of network.

Theorem 5.23 (*[39]*). *MinMaxC remains NP-complete on interdependent-power networks.*

In what follows, we show that the *MinMaxC* remains NP-complete on split graphs. For this purpose, we shall first prove the following equivalence between the *MinMaxC* and the *CNP* on this class of graphs.

Theorem 5.24. *The CNP and the MinMaxC are equivalent on split graphs.*

Proof. Given a split graph $G = (V, E)$, for any set of nodes $S \subseteq V$, the induced graph $G[V \setminus S]$ contains a nontrivial connected component and isolated nodes, if any.

We recall that the recognition version of the *CNP* and *MinMaxC* seek to find a set of at most k nodes the deletion of which, respectively, minimizes pairwise connectivity, and minimizes the largest component size in the remaining graph. According to the value of k, two cases are considered. The first is a trivial case where $k \geq |N(V_1)|$. In this case, the optimal solution is to delete the nodes of $N(V_1)$ and any $k - |N(V_1)|$ nodes from V_2. This results in a graph with $|V_1| + 1$ components, where $|V_1|$ components are isolated nodes.

For the second case where $k < |N(V_1)|$, we prove that solving *MinMaxC* is equivalent to solving the *CNP*. Given an optimal solution, say s^*, for *MinMaxC* on graph G, s^* aims to delete the set of nodes $S \subseteq V$ so that the size of the nontrivial connected component of $G[V \setminus S]$ is minimized (note that the other components in $G[V \setminus S]$ are only isolated nodes). For that, we have to isolate as many as possible nodes from V_1 while deleting nodes only from V_2.

The solution s^* is also an optimal solution for the *CNP*, since the *CNP* tends to minimize pairwise connectivity in G. In doing so, it should minimize the size of the nontrivial connected component of $G[V \setminus S]$ and, at the same time maximize the number of surviving isolated nodes of the independent set $|V_1|$. This is exactly what the solution s^* aims to achieve.

Therefore, the *MinMaxC* and the *CNP* are equivalent on split graphs. ■

According to Theorem 5.24 and since the *CNP* is NP-complete on split graphs [65], we have the following corollary.

Corollary 5.25. *MinMaxC is NP-complete on split graphs.*

The weighted version of *MinMaxC*, namely *MinMaxCw2* (where $c_i > 1$ and $w_i > 1 \; \forall i \in V$), has also been proven to be NP-complete on trees [46] through a simple reduction from the Set Partition problem, known to be NP-complete on this class of graphs.

Theorem 5.26 (*[46]*). *MinMaxCw2 is NP-complete on trees.*

The inapproximability of *MinMaxC* on interdependent-power networks has been proven in [39]. The authors showed that the problem is NP-hard to approximate within a factor $(2 - \epsilon)$ for any $\epsilon > 0$ on this class of networks, as can be deduced from the following theorem.

Theorem 5.27 (*[39]*). *It is NP-hard to approximate MinMaxC on interdependent-power networks within a factor* $(2-\epsilon)$ *for any* $\epsilon > 0$.

Table 6 summarizes different complexity results obtained in the literature considering *MinMaxC* on different classes of graphs.

5.4.2. Solving MinMaxC

MinMaxC was formulated using a mixed-integer programming model by Shen et al. [64]. Authors derived valid inequalities to solve the model using the polynomial-time dynamic program used for solving the problem on k-hole graphs presented in [46]. They also established bounds for optimal objective function values. The efficiency of the proposed model was examined on networks of at most 50 nodes.

The only work using heuristic algorithms for solving *MinMaxC* is presented in [39]. Accordingly, the authors proposed a greedy framework to solve the problem on interdependent-power networks using three different heuristics. The common idea is to iteratively select the most critical node using a heuristic until the k nodes have been selected. The first heuristic consists in selecting at each iteration the node that maximizes the number of failed nodes after the cascading failure. The potential nodes, the removal of which maximizes the cascading failures are articulation nodes. Thus, based on this property, the algorithm first identifies, for each iteration, all articulation nodes in both residual networks, and computes the cascading failure for each one. It then selects the node with the maximum value of cascading failures, and so on until k nodes have been selected. The running time of this heuristic is $O(kn^2)$.

The second heuristic consists in selecting, at each iteration, the node that maximally decreases the structural strength of the network. On single networks, it has been shown that the nodes which greatly affect the network structure are the central nodes, as they are those which interconnect the other nodes in the network. Therefore, using the same principle, authors developed a new adapted centrality metric which measures node centrality for interdependent networks. The algorithm then selects critical nodes with respect to this measure. This heuristic also has a polynomial running time. The third heuristic is a hybrid of the first two and has a polynomial running time too.

We note that using articulation and central nodes to greedily determine critical nodes on single networks has major weaknesses (see Section 4), which makes the algorithm fail to accurately identify them. This needs to be checked for interdependent-power networks.

On particular graph classes, Shen et al. [46] proposed a polynomial-time algorithm, based on dynamic programming, to solve the *MinMaxC* problem on trees, k-hole and series-parallel graphs, with time complexity $O(n^3 log_n)$, $O(n^{3+k} log_n)$ and $O(n^5 log_n)$ respectively. They also showed that, using the same dynamic program, *MinMaxCw1* can be polynomially solved on trees. Furthermore, for *MinMaxCw2* on paths, they presented a polynomial-time algorithm that lists all possible combinations of deleting k nodes and then selects the one that minimizes the size of the connected components. In doing so, they transformed *MinMaxCw2* into a problem of finding the shortest path, where critical nodes are those through which the path passes.

5.5. CC-CNP: Cardinality constrained critical node problem

Given a graph $G = (V, E)$, the *CC-CNP* aims to minimize the set of nodes, the deletion of which constrains the pairwise connectivity of each connected component in the remaining graph to a given bound. Its formulation can be stated as follows:

Input: An undirected graph $G = (V, E)$ and an integer L.
Output: $\underset{S \subseteq V}{argmin} \; |S|$, *s.t.* $\sum_{v_i, v_j \in h} u_{ij} \leq L$, for each connected

Table 6
MinMaxC complexity results on different classes of graphs.

$\mathcal{CNDP}$ Variant	Measure	Graph class		Complexity
MinMaxC	Minimize the largest component size	Tree	$w_v = 1, c_{vu} = 1$	Polynomial [46]
			$w_v \geq 0, c_{vu} = 1$	
		Chain	$w_v \geq 1, c_{vu} \geq 0$	NP-complete [46]
				Polynomial [46]
		Interdependent power networks		NP-complete [39]
		k-hole graphs Series-Parallel graphs		Polynomial [46]

components $h \subseteq G[V \setminus S]$.

$$u_{ij} = \begin{cases} 1 & \text{if } v_i \text{ and } v_j \text{ are in the same component in } G[V \setminus S], \\ 0 & \text{Otherwise.} \end{cases}$$

In the following observation, we provide a direct relationship between the connected component order and its pairwise connectivity, in the case of unit connection costs, *i.e.*, $c_{ij} = 1, ij \in E$.

Observation 5.28. *Let L be an integer and h be a connected component of order L. If the pairwise connectivity of h is at most B, then $L \leq \frac{1+\sqrt{1+8B}}{2}$.*

Hence, the *CC-CNP* (in the case where $c_{ij} = 1$) seeks to find a minimal set of nodes to be deleted, such that the order (cardinality) of each connected component in the residual graph is no more than a given number L of nodes.

Input: A graph $G(V, E)$ and an integer L.
Output: A minimum set of nodes $S \subseteq V$, where $\sigma_h \leq L$, for each component $h \in G[V \setminus S]$.

Solving the *CC-CNP* where $L = 1$ amounts to finding a minimum vertex cover set of the graph. Hence the *CC-CNP* is a generalization of the Minimum Vertex Cover problem. We can also easily note that the *CC-CNP* is the reverse version of *MinMaxC*. In fact, the *CC-CNP* aims at bounding the connected component size in the remaining graph to a given threshold L by deleting a minimum set of S nodes. On the contrary, the *MinMaxC* aims at bounding the set of nodes S to a given budget while minimizing the size of connected components in $G[V \setminus S]$.

5.5.1. Complexity results

Arulselvan et al. [62] proved the NP-completeness of the *CC-CNP* using the result demonstrated by Krishnamoorthy and Deo [121] for a class of node-deletion problems. Hence the following theorem holds.

Theorem 5.29 (*[62]*). *The CC-CNP is NP-complete on general graphs.*

Even on bounded-degree graphs, the *CC-CNP* remains NP-complete. This was shown in [68]. Authors proved that the *CC-CNP*, called *3C-CNP* for *Component Cardinality Constrained CNP*, is NP-complete on graphs of maximum degree 4, through a reduction from the Minimum Vertex Set Cover problem on cubic graphs, known to be NP-complete [122]. Hence this theorem follows.

Theorem 5.30 (*[68]*). *The CC-CNP is NP-complete on graphs of maximum degree $\Delta = 4$.*

The *CC-CNP* remains NP-complete on trees for the case where nodes and connections have, respectively, nonnegative weight and cost. This has been proven in [68] through a simple reduction from the *MinMaxCw2* (which is the version of *MinMaxC* where nodes have nonnegative weight and nonnegative deletion cost, see Section 5.4).

Theorem 5.31 (*[68]*). *The CC-CNP is NP-complete on trees for the case where nodes and connections have, respectively, non-negative weight ($w_i > 0, \forall i \in V$) and cost ($c_{ij} > 0, \forall ij \in E$).*

On split graphs, the *CC-CNP* remains NP-complete. In [68], authors established the equivalence between the *CC-CNP* and the *CNP*. Since *CNP* is NP-complete on this class of graphs (see Theorem 5.3), the following theorem holds.

Theorem 5.32 (*[68]*). *The CC-CNP remains NP-complete on split graphs.*

As the *CC-CNP* is a generalization of the Minimum Vertex Cover problem (*MVC*), then using the inapproximability result of the *MVC* [99] we can easily deduce that the *CC-CNP* is NP-hard to approximate on general graphs. In [93], this result has been proven while providing the following inapproximability factor.

Theorem 5.33 (*[93]*). *The CC-CNP cannot be approximated in polynomial time within a factor $(\frac{n}{L})^{1-\epsilon}$, where $\epsilon > 0$.*

Table 7 summarizes different complexity results obtained in the literature considering *CC-CNP* on different classes of graphs.

5.5.2. Solving CC-CNP

The *CC-CNP* has been thoroughly explored from the integer programming perspective [74,93,62]. In [74], Oosten et al. proposed an integer programming formulation for the problem. They also studied the polyhedral structure of the formulation, as well as its relationship with other carefully polytopes. Based on the derived polyhedral results, a branch-and-cut algorithm has been implemented. In [93], more results from the polyhedral structure of the *CC-CNP* were provided, and a detailed study has been carried out.

Moreover, Arulselvan et al. [62] presented an integer programming formulation, similar to the one proposed for the *CNP* [33] with a slight modification. Also, Veremyev et al. [85] proposed a more compact linear formulation for the *CC-CNP* and provided valid inequalities to improve the performance of the model.

To provide approximated solutions for this problem, an approximation algorithm based on the randomized rounding approach [102] has been proposed in [80]. The proposed approach provides a $1/(\theta - 1)$-approximation to the number of removed nodes, where θ is the rounding threshold. Also, two $(L+1)$-approximation algorithms, where L is the bound on the connected component size, have been presented in [93] to solve the problem on general graphs.

Using heuristic approaches, Arulselvan et al. [75,62] reused the heuristic algorithm proposed for the *CNP* [33] to solve the *CC-CNP* on general graphs (see Section 5.2.2). Also, a multi-start greedy algorithm has been developed in [89]. This algorithm started with identifying the total number of nodes in excess of L in all graph components. This value is updated after each operation whenever a node is added or removed. Then, it proceeds as follows. First, it randomly selects a set of nodes S to be removed such that the

Table 7
CC-CNP complexity results on different classes of graphs.

$\mathcal{CNDP}$ Variant	Metric	Graph class		Complexity
CC-CNP	Limit the cardinality of each connected component	General graph		NP-complete [62]
		Graph with $\Delta \leq 4$ Split graph		NP-complete [68]
		Tree	$w \geq 1, c \geq 0$	Linear [68]
			$w = 1, c = 1$ $w = 1, c \geq 0$	
			$w \geq 0, c = 1$	Polynomial [68]
		Proper interval graph		
		mK2-free, weakly-free, interval, interval-filament, asteroidal triple-free, and circular-arc graphs		Polynomial [93]

induced graph has components of at most L nodes. Note that the number of nodes to be deleted k is fixed as a target value and given in input. Then, it proceeds to update S by repeating the operation of adding/removing nodes from S until the number of nodes removed is equal to the target value given in input, and all components have at most L nodes. The selected node to be added is the one, the removal of which minimizes the increase in the number of nodes in excess of L. A partial restart is periodically activated to prevent search stagnation. The efficiency of the algorithm was evaluated through computational experiments using a set of real-world graphs, and the results were compared with exact solutions obtained using the ILP described in [82].

Considering metaheuristic approaches, two algorithms have been proposed [62,83]. The first is a genetic algorithm developed in [62] and was shown to outperform the heuristic based on the local search presented in the same work in terms of solution quality. The second is a more sophisticated evolutionary platform developed in [83], where the initial solutions obtained using the reproduction and mutation operations are corrected using greedy operations before returning the final solution.

On particular graph classes, the *CC-CNP* has been studied by Lalou et al. [68] on trees, split and proper interval graphs (*i.e.,* the subclass of the interval graphs where no interval is properly contained in another). On trees, authors considered different cases according to node weight and connection cost. For the case where connection costs are unit and node weights are arbitrary, a polynomial-time algorithm based on the dynamic programming approach has been developed to solve the problem with time complexity $O(nL^2)$. Also, for the case where nodes have unit weights and connections have arbitrary costs, a polynomial-time algorithm of complexity $O(n^2)$ has been developed. The algorithm begins with a post-order traversal of the tree. Then, for each subtree (formed in the order) with a cardinality of more than L nodes, it deletes the root node. We note here that the optimal solution for the *CC-CNP* on a subtree, with a cardinality of more than L nodes, is to delete the root node. The set of critical nodes is thus the set of all deleted root nodes. Also, considering the *CC-CNP* on proper interval graphs, authors in [68] proposed a polynomial-time algorithm of complexity $O(n^2)$.

Based on a general reduction of the *CC-CNP* from the Maximum Weight Independent problem established in [93], authors [93] showed that the *CC-CNP* is polynomially solvable on mK2-free [123], interval-filament [124], asteroidal triple-free [43], weakly-free [125], interval [43] and circular-arc graphs [43].

In the following proposition, we give a polynomial-time algorithm, of better complexity than that in the literature, for solving *MinMaxC* on trees. For this purpose, we use the linear algorithm presented in [68] for solving the *CC-CNP* on trees for the case of unit connection cost and unit node weight.

Proposition 5.34. *There is a polynomial-time algorithm of complexity $O(n^2)$, where n is the number of nodes, for solving the MinMaxC problem on trees.*

Proof.

Given a tree $T = (V, E)$, where $|V| = n$, Shen et al. [46] proposed a polynomial-time algorithm for solving the *MinMaxC* problem on T of complexity $O(n^3)$. Using the linear algorithm presented by Lalou et al. [68] for solving the *CC-CNP* on T, we develop a polynomial-time algorithm of complexity $O(n^2)$ as follows. First, we recall that for the *CC-CNP*, the linear algorithm [68] returns the minimum set of critical nodes, the deletion of which limits the connected component size in the remaining graph to a given bound L. While in the *MinMaxC* problem, we aim to minimize the size of connected components by deleting at most k nodes. We thus run the linear algorithm of the *CC-CNP* n times ranging from $L = 0$ to $L = n - 1$. In each iteration $L = m$, where $m \in \{0, \ldots, n-1\}$, the linear algorithm returns a set of critical nodes S_m. It is clear that for every pair value (m_1, m_2) of L, if $m_1 < m_2$, then $|S_{m_1}| \geq |S_{m_2}|$. Based on this observation, we can stop the execution at the smallest integer m_i for which $|S_{m_i}| > k$, and return S_{m_i-1} as the set of critical nodes for the *MinMaxC* problem, where m_i is the minimum component size in the induced graph. It is easy to see that we require at most n executions of the linear algorithm and, hence, that the new algorithm runs on $O(n^2)$.

This completes the proof. ■

5.6. β-vertex disruptor Problem

The *β-vertex disruptor* problem asks for minimizing the number of nodes, the deletion of which reduces overall pairwise connectivity to a specific level. This level is defined by a given fraction β of the total pairwise connectivity of the input graph. The problem can be formulated as follows.

Input: A graph $G(V, E)$ where $|V| = n$, and a fixed fraction number
$0 \leq \beta \leq 1$.
Output: A minimum set of nodes $S \subseteq V$ the deletion of which constrains pairwise connectivity in $G[V \setminus S]$ to $\beta\binom{n}{2}$.

Note that using β as an input parameter allows a variety of disconnectivity levels to be treated in the remaining graph. Also, we can easily see that the β-vertex disruptor problem and the *CNP* are complementary versions of the same problem. Indeed, in both variants we look for a set of at most k nodes, the deletion of which bounds pairwise connectivity to a threshold L for the *CNP*, while for the β-vertex disruptor problem, the threshold is given by a fraction β of the aggregate pairwise connectivity. We recall that the total pairwise connectivity in a graph of n nodes is $\binom{n}{2} = \frac{n(n-1)}{2}$). Thus, the two variants have the same recognition version, allowing us to make the following observation.

Observation 5.35. *The recognition version of the β-vertex disruptor problem is equivalent to the recognition version of the CNP where $\beta = \frac{2L}{n(n-1)}$; L is the pairwise connectivity bound in the CNP.*

5.6.1. Complexity results

This variant was defined and studied only by Dinh et al. [36,94,63]. In [63], authors proved the NP-completeness of the problem on directed graphs through a reduction from the Minimum Vertex Cover problem, resulting in the following theorem.

Theorem 5.36 (*[63]*)**.** *The β-vertex disruptor problem is NP-complete on directed graphs.*

The following corollary is a direct consequence of the *Observation* 5.35 establishing the relationship between the recognition version of the *CNP* and that of the β-vertex disruptor problem. We can deduce that the β-vertex disruptor problem is NP-complete on classes of graphs on which the *CNP* is NP-complete and vice versa. Hence, we have the following lemma.

Lemma 5.37.
Solving the β-vertex disruptor problem is:

(i) *NP-complete on split, bipartite, complement bipartite, unit-disk, power-law and 4-bounded degree graphs, and on trees for general connection costs even for unit node weight.*
(ii) *Polynomially solvable on trees with unit connection costs even for general node weight, and on graphs with bounded treewidth.*

We can note that solving the β- *vertex disruptor* problem where $\beta = 0$ is equivalent to solving the Minimum Vertex Cover problem, which is known to be NP-hard to approximate within a constant factor of less than 1.36 [99]. The following theorem due to Dinh et al. [63] follows.

Theorem 5.38 (*[63]*)**.** *The β-vertex disruptor problem does not admit a PTAS scheme unless* P = NP.

5.6.2. Solving the β-vertex disruptor problem

Using the integer programming approach, Dinh et al. [36] presented a formulation of the β-vertex disruptor problem and proposed a branch-and-cut algorithm to solve it. In order to reduce the number of explored branches in the branch-and-bound search tree, they used a greedy rounding heuristic.

Despite the intractability of the problem, authors in [63] proposed a polynomial pseudo-approximation algorithm that provides an approximation within a factor $O(lognloglogn)$ of the optimal solution.

In [94], an extended version of the β- *vertex disruptor* problem was defined. The new version considers the identification of both critical nodes and edges. Authors studied the NP-completeness of the problem and proposed a bi-criteria approximation algorithm as well as a metaheuristic approach for solving the problem on both directed and undirected graphs.

In the following theorem, we prove that a polynomial-time algorithm for solving the *CNP* can be used to polynomially solve the β-vertex disruptor problem on the same graph.

Theorem 5.39. *Given a graph G, if the CNP is polynomially solvable on G with time complexity $O(\alpha)$, then there is a polynomial-time algorithm of complexity $O(n\alpha)$ for solving the β-vertex disruptor problem on G.*

Proof. Given a graph $G = (V, E)$, where $|V| = n$, let $\mathcal{A}$ be a polynomial-time algorithm of complexity $O(\alpha)$ that solves the *CNP* on G. We can derive a polynomial-time algorithm $\mathcal{A}^*$ of complexity $O(n\alpha)$ to solve the β-vertex disruptor problem on G. To do so, we proceed as follows. Let s_i^* denote the optimal solution of the *CNP* on G by deleting i nodes, using the algorithm $\mathcal{A}$. We run $\mathcal{A}$ on G for $i = \{1, \ldots, n\}$ (where i is the number of nodes to be deleted). After that, we sort, in increasing order according to i, all optimal solutions s_i^*. For each s_i^*, we check if the connectivity of the induced graph is no more than a fraction β (we recall that the β-vertex disruptor problem asks for the minimal set of nodes, the removal of which bounds the pairwise connectivity on the induced graph by a given fraction β). The first solution in the order, for which the connectivity of the residual graph is less than β, represents the optimal solution for the β-vertex disruptor problem on G. This is true given that the pairwise connectivity in the residual graph obtained by deleting a set of k_1 nodes is less than the pairwise connectivity obtained by deleting a set of k_2 nodes if $k_1 \leq k_2$.

It is obvious that the time complexity of the algorithm $\mathcal{A}^*$ is $O(n\alpha)$. We recall that we run the algorithm $\mathcal{A}$, of complexity $O(\alpha)$, n times for $i = \{1, \ldots, n\}$.

This completes the proof. ■

5.7. Other variants

In this section, we present other variants that did not receive much attention in the literature, or are a combination of some variants mainly studied in the literature, and which are already presented in previous sections. These include: the *MinNumLC* (for *Minimizing Number of Large Components*) [65], the *MaxNumSC* (for *Maximizing Number of Small Components*) [65], the *Sum-of-squares partition problem* [41], the *Min–Max BC optimization problem* [126] and the *Max partition size* [127,128] .

(i) MinNumLC and MaxNumSC. Given a graph $G = (V, E)$ and two positive integers k and c, the *MinNumLC* (resp. *MaxNumSC*) problem consists in minimizing (resp. maximizing) the number of connected components with a cardinality of at least c (resp. at most c), by deleting k nodes from G. Let $f_c(S)$ (resp. $f^c(S)$) be the function that returns the number of connected components in $G[V \setminus S]$ with a cardinality of at least (resp. at most) c. The formulation can be stated as follows:

Input: A graph $G = (V, E)$, and two integers c and k.
Output (MinNumLC): $\underset{S \subseteq V}{argmin}\, f_c(S)$, where $|S| \leq k$.
Output (MaxNumSC): $\underset{S \subseteq V}{argmax}\, f^c(S)$, where $|S| \leq k$.

The two variants have only been considered in [65]. The authors showed that the two variants are polynomially solvable on graphs of bounded treewidth.

It is obvious that *MaxNum* is a special case of the *MinMaxSC* problem where $c = |V|$. Thus, as the *MinMaxSC* problem is polynomially solvable on weighted graphs (where $w_i \geq 0, \forall v_i \in V$) with bounded treewidth, we have the following corollary.

Corollary 5.40. *MaxNum is polynomially solvable on graphs with bounded treewidth.*

(ii) Sum-of-squares partition problem. This variant aims at partitioning a given graph by removing k nodes, such that the sum-of-squares of connected component sizes, once the nodes have been deleted, is minimized. It can be formulated as follows:

Input: A graph $G = (V, E)$ and an integer k.
Output: a set $S \subseteq V$, where $|S| \leq k$, the deletion of which partitions the graph into connected components $h_1, \ldots, h_m$, such that $\sum_i |h_i|^2$ is minimized.

This variant was defined by Aspnes et al. [41] through a study of the inoculation problem. This problem consists in selecting on which nodes we install the anti-virus software to contain the

spread of computer viruses in a network. Authors reduced the inoculation problem to the Sum-of-squares partition problem. They proved that the problem is NP-complete on general graphs, and is NP-hard to approximate within a bi-criterion factor $(15/14-\epsilon)$ for any $\epsilon > 0$. However, they proposed a polynomial-time approximation algorithm that finds a solution with an approximation ratio of at most $O(log^{1.5}n)$.

(iii) Min–Max BC optimization problem. In this variant, we seek for a set of k nodes, the deletion of which minimizes the maximum Betweenness Centrality (BC)[6] value of the graph. Let $BC(G, v)$ denote the function that computes the maximum betweenness centrality value of node v in G, the problem can be formulated as follows:

Input: A graph $G = (V, E)$, and an integer k.
Output: $\underset{S\subseteq V}{argmin} \ \underset{\forall v\in V\setminus S}{max} BC(G[V \setminus S], v)$, where $|S| \leq k$.

This variant was defined by Lozano et al. [126] for studying the problem of designing effective network attacks. The aim is to degrade the network structure in such a way that the residual graph (representing the network) has no strong betweenness centrality leaders, *i.e.* the network will have the minimum nodes that mainly influence the information flow.

An approach for solving the problem on general graphs was developed using the Artificial Bee Colony (ABC) algorithm [129], which is a good metaheuristic used to efficiently solve many NP-hard optimization problems. Its main characteristic is the use of BC update procedures that efficiently recompute the BC values of a graph according to modifications of its structure by node deletions and/or insertions (note that computing the BC value is very expensive). The performance of the proposed algorithm was shown by detailed computational experiments conducted on up to 60 graphs.

(iv) Max partition size problem. This variant aims at simultaneously minimizing the number of nodes to be deleted and the component size in the residual graph once the nodes have been deleted, while partitioning the graph on at least m components (*i.e.* on m components not necessary connected). The *Max partition size problem* can seen as a fusion of the two variants already presented, namely the *CC-CNP* and the *MinMaxC*. Indeed, it does not have a constraint on the number of nodes to be deleted and seeks to minimize it, as the *CC-CNP*. On the other hand, it does not have a constraint on the component size and seeks also to minimize it, which is the case of the *MinMaxC*. However, it adds a constraint on the number of connected components, that is the graph has to be divided in a fixed number m of partitions.

This variant was considered in [127], where a heuristic for solving it on general graphs has been proposed. The heuristic is based on a spectral clustering method applied on the line graph that represents the network. This method consists in partitioning the graph into m balanced partitions, called communities, *i.e.* with many edges joining nodes of the same partition and comparatively few edges joining nodes of different partitions. Then, the critical nodes are those the deletion of which disconnects communities.

The same authors [130] defined another slightly different version where the number of connected components in the residual graph, once the critical nodes have been deleted, is also maximized instead of being bounded. Hence, the new problem seeks to, at the same time: *(i)* minimize the number of nodes to be deleted; *(ii)* minimize the component size in the residual graph; and *(iii)* maximize the number of disconnected components. They presented an integer linear programming formulation for the problem, and also a heuristic for solving it on general large graphs.

[6] The BC value of a node $v \in G$ is the number of shortest paths between any two nodes of G that pass through node v.

6. Discussion and future research directions

From this review, we can easily note that the *CNDP* is generally NP-complete and that almost all its variants remain NP-complete even on particular graph classes. Also, proofs of NP-completeness have been thoroughly detailed. However, the development of solving algorithms, which is even more important, has attracted less attention, except for the first variant, namely the *CNP*. We can also note that the development of improved ILP formulations has drawn a large attention, where a variety of ILP models have been proposed for all variants [33,85,112,73,64,62,74,36,77,91,130], including a unifying ILP framework considering different connectivity metrics [82]. Moreover, the approximation algorithms based on these formulations, specially for the *CNP*, have been explored [78,79,37,80,38,77]. However, these methods are usually able to deal only with relatively small networks.

Table 8 depicts the various approaches proposed for solving different variants of the *CNDP*, where we provide, for each one, the time complexity, if any. We note that, for general graphs, we do not consider exact and approximated algorithms based on mathematical formulations.

From Table 8, we can see that tackling the *CNDP* on real-world networks, which is an important issue, has only been considered for the *CNP*. Indeed, many experimental studies were carried out and detailed comparisons between different approaches in terms of both solution quality and running time can be found in [81,92,96,86,88,83,89,87,90,85,82]. To experimentally evaluate the *CNDP* solving approaches, a variety of appropriately chosen instances derived from real problems,[7] as well as different benchmark random graphs that utilize Erdos–Renyi, Watts–Strogatz, Forest Fire and Barabasi–Albert models[8] (described in detail in [81]) are widely used in the literature. These graphs were defined in such a way that they present different network structures. Other real-world network instances can be obtained from the University of Florida Sparse Matrix Collection database [131]. Moreover, a nice collection of large and real-world network datasets, including social networks, web graphs, citation networks, collaboration networks, and communication networks, can be found in the Stanford Network Analysis Project home page.[9]

Table 9 presents an example of comparison between different approaches proposed for solving the *CNP*, namely the Integer Linear Programming (ILP) method [73], the Population-based Incremental Learning (PBIL) and the simulated annealing (SA) approaches presented in [81], the Variable Neighborhood Search algorithm (VNS) [86], the Greedy Randomized Adaptive Search Procedure with Path Relinking (GRASP_PR) [90], the GRASP with the Exterior Path Relinking (GRASP_EPR) [106], the greedy algorithm (GrdA) [87], the multi-start greedy algorithm (Ms_GrdA) [89], and the Iterated Local Search metaheuristic (ILS) [88]. The table provides the minimum values of the objective function for the 16 benchmark instances presented in [81]. More details about these results are reported in [81,86,90,106,87,89,88].

The other variants have generally been considered on some particular classes of graphs, for which polynomial algorithms exist, and the only algorithm proposed for solving them on large networks is the evolutionary method presented in [83]. Moreover, other approaches and categories of metaheuristics that efficiently work in large networks, such as Swarm intelligence, have not been investigated in depth, especially for these variants. Furthermore, the problem parameterization technique, which is an interesting alternative as it helps develop efficient exact exponential algorithms, has only been explored for the *CNP* and *MaxNum*. Another

[7] http://www.di.unito.it/~aringhie/cnp.html.
[8] http://individual.utoronto.ca/mventresca/cnd.html.
[9] https://snap.stanford.edu/index.html.

Table 8
Different solving approaches used to develop algorithms for solving the *CNDP*.

Variant	Graph class	Solving approach	Time
CNP	General graphs	Greedy heuristic [33]	$O(n^2m)$
		enhanced using a 2-exchange local search	
		Combined greedy algorithm enhanced by more sophisticated local search procedures [87,88]	NC
		Multi-start greedy algorithm [89]	NC
		Genetic algorithm [83]	NC
		Simulated Annealing [81]	NC
		Population Based Incremental Learning [81]	NC
		Iterated Local Search [88]	NC
		Variable Neighborhood Search [88]	NC
		GRASP with Path Relinking [90,106]	NC
		Multi-objective evolutionary algorithms [92]	NC
	Trees	Dynamic programming [67]	$O(n^3k^2)$
	Graphs with bounded T_w	Dynamic programming [65]	NC
MaxNum	General graphs	Genetic algorithm [83]	NC
	Trees		$O(n^3)$
	k-hole graphs	Dynamic programming [46]	$O(n^{3+k})$
	Series–parallel graphs		$O(n^3)$
	Graphs with bounded T_w	Dynamic programming [66]	$O(nk^2w^w)$
MinMaxC	General graphs	Greedy heuristic [39]	$O(kn^2)$
		Genetic algorithm [83]	NC
	Trees		$O(n^3log_n)$
	k-hole graphs	Dynamic programming [46]	$O(n^{3+k}log_n)$
	Series–parallel graphs		$O(n^5log_n)$
CC-CNP	General graphs	Greedy heuristic [62]	$O(n^2+nm)$
		Multi-start greedy algorithm [89]	NC
		Genetic algorithm [62,83]	NC
	Trees	Polynomial algorithm [68]	$O(nL^2)$
	Proper interval graphs		$O(n^2)$
β-vertex DP	General graphs	Genetic algorithm [83]	NC

Table 9
A comparison of minimum values of the objective function obtained by different approaches used for solving the *CNP* on the 16 graph instances presented in [81]. For each instance, #Nd, #Eg and #CN denote respectively the number of nodes, the number of edges, and the number of critical nodes. Note that for the *CNP*, the objective is to minimize pairwise connectivity in the residual graph. The best known values are presented in bold.

Graph instance				ILP	PBIL	SA	VNS	GRASP_PR	GRASP_EPR	GrdA	Ms_GrdA	ILS
				[73]	[81]	[81]	[86]	[90]	[106]	[87]	[89]	[88]
	#Nd	#Eg	#CN									
Barabasi–Albert (BA)	500	499	50	**195**	892	997	**195**	**195**	**195**	**195**	**195**	**195**
	1000	999	75	**558**	3057	3770	559	**558**	**558**	559	**558**	559
	2500	2499	100	**3704**	28044	31171	**3704**	**3704**	**3704**	3722	**3704**	3722
	5000	4999	150	**10196**	146753	170998	**10196**	**10196**	**10196**	**10196**	**10196**	10222
Erdos–Renyi (ER)	235	350	50	**295**	6700	7700	298	297	**295**	313	297	313
	466	700	80	NA	44255	48627	1542	1575	**1536**	1938	1548	1874
	941	1400	140	NA	229576	234479	5628	5482	**5102**	8106	5345	5544
	2344	3500	200	NA	2009132	2011122	1052406	1048464	**1010487**	112685	1014340	1062536
Watts–Strogatz (WS)	250	1246	70	NA	13786	14251	6610	9351	**3192**	11401	4465	3241
	500	1496	125	NA	53779	54201	2230	2209	2099	**11981**	2141	2282
	1000	4996	200	NA	308596	311700	154813	297241	128657	318003	171635	**148856**
	1500	4498	265	NA	703241	717369	15692	14177	**13773**	243190	14461	14681
Forest Fire (FF)	250	514	50	**194**	1386	1841	**194**	**194**	**194**	197	**194**	195
	500	828	110	**257**	1904	2397	**257**	**257**	**257**	262	**257**	261
	1000	1817	150	**1260**	59594	92800	1263	1261	**1260**	1271	**1260**	1276
	2000	3413	200	**4545**	256905	387248	4584	4554	**4545**	4592	4548	4583

interesting solving approach, which has not yet been considered, is the possibility of distributed and parallel computing of critical nodes. On the other hand, we can note that some variants of the *CNDP*, particularly the β-vertex disruptor variant, have received less attention both on general and particular classes of graphs.

Thus, given the importance of the problem, and looking at the different studies explored in previous works, there are gaps that need to be considered for a thorough study of the problem. In what follows, we give some general ideas and perspectives to be developed in future works.

(i) It would be interesting to study the weighted version of problems where nodes may have different weights, since nodes do not usually have the same influence in the network. We can note that few studies have been conducted especially in the case of general graphs for almost all variants. Examples of the few works dealing with this are [112,113]. It is also interesting to consider different variants on directed graphs, as many real-world networks are modeled by digraphs, for example: web, scheduling networks, research citation networks, transport networks, etc. It is known that connectivity problem on digraphs is much harder than on undirected graphs, and hence many approaches designed for identifying critical nodes on undirected graphs do not work for digraphs. Therefore, more effort is required for studying the *CNDP* on this class of graphs. In the literature, there is only the work in [115] that considered the *CNP* on digraphs .

(ii) Some *CNDP* variants, such as *MinNumLC* [65], *MaxNumSC* [65] and the *distance-based CNP* [112–114], have received less attention in the literature, despite their importance, and hence they need more consideration.

(iii) For a more detailed study of problem complexity, it is interesting to consider parameterized complexity. From this perspective, the only two variants considered in the literature are the *CNP* and *MaxNum*, and there is a gap in the literature for the other variants. Thus, it is very useful to examine the possible parameterizations of these variants. Note that this helps develop efficient solving algorithms under small values of some parameters, as the parameterization allows the complexity of the problem to be thoroughly explored and properly understood in terms of how different parameters affect this complexity.

(iv) Despite the numerous classes of graphs considered in previous works, there are some other interesting classes that need to be addressed. For instance, the consideration of *CNDP* on mobile networks, represented by dynamic graphs, would be an interesting issue. In fact, many mobile systems, such as ad hoc and wireless networks, expand in a decentralized manner which usually results in a weak topology presenting different critical elements that need to be maintained. Thus, studying the *CNDP* on this kind of network is of major importance, particularly since only a few works have considered this issue in the literature [38].

(v) Given that the *CNDP* usually finds its applications in large-scale networks, corresponding to real-world networks, thus development of efficient algorithms to solve it on this kind of network is very important. However, except for the *CNP* variant, only one work [83] considered the other variants on general graphs. Therefore this calls for further attention. Also, metaheuristic approaches such as ant colony optimization, particle swarm optimization, etc., which work efficiently in this kind of networks, have been little investigated for almost all variants.

(vi) Another important CNDP aspect, which has not yet been explored, is the possibility of distributed and parallel computing of critical nodes. This is an alternative way for solving hard problems within reasonable time. This may be worth considering, especially with recent advances in grid computing where efficient algorithms have been developed.

(vii) Generally, key players in networks are nodes or edges. We can then seek to find critical edges instead of nodes, which is useful, for example, for studying individual behaviors in social networks (in this kind of network, edges represent social relationships). This edge version of the problem has already been considered in different works in the literature, including the edge-deletion problem [120], graph partitioning [6], the edge multiterminal cut problem [132] and graph clustering [133]. Consequently, it would be interesting to exploit and adapt different existing approaches proposed to solve the *CNDP* when dealing with this version of the problem.

7. Applications of CNDP

The *CNDP* has many real-world applications in a number of fields, including network risk management [75], network vulnerability assessment [36,63,37], social network analysis [134,135], biological molecule studies [34,35] and network immunization [33,136], network communication [137–139]. In this section, we present a variety of applications, considered in the literature, for which critical nodes of the corresponding network have been identified to tackle the problem being studied. We also show how identifying these nodes may be a useful parameter for the applications not yet considered.

Computational biology. Biological organisms, such as bacteria and viruses, are sets of interconnected proteins that interact with each other to form a protein-interaction network. These networks can be represented by a graph where nodes are proteins and edges are interactions between them. Protein structure and interactions are widely studied in biology [140–143]. Identifying critical nodes (proteins), which maintain connectivity between all proteins, can provide useful information for many biological applications. For instance, in rational drug design [144,145], these critical proteins need to be targeted to destroy and neutralize the corresponding harmful organism, resulting, from a pharmaceutical perspective, in a therapeutic benefit for the patient. This is what V. Tomaino *et al.* [35] investigated with the purpose of destroying aggressive cancer cells. To do so, they identified critical nodes in the human protein interaction network using the *CNP* variant.

Likewise, V. Boginski et al. [34] explored the identification of such proteins using the *CC-CNP* variant. This work is a good application of the concept of critical nodes in computational biology.

Network vulnerability and survivability. For a long time now, network vulnerability assessment has been of major importance for network risk management. A network could be vulnerable to several potential disruptive events, such as disasters. Generally, network weaknesses are nodes and links, the corruption of which significantly degrades network performance. Therefore, assessing network vulnerability using the *CNDP* makes it possible to explore the main weaknesses in the network and, hence, to protect it against any disruptive events. Different works [4,36,63,112] showed that identifying critical nodes is more suitable for assessing network vulnerability than other methods commonly used in the literature, such as centrality [146].

In [36,63], Dinh et al. converted the network vulnerability assessment problem into a problem of identifying critical nodes in the network using the β-vertex disruptor variant. Network vulnerability is then measured by the minimum number of nodes, the removal of which impairs network pairwise connectivity. Authors showed that this variant ensures a good assessment of vulnerability for two main reasons. First, it is based on pairwise connectivity metric, which quantifies the network disconnectivity while ensuring the balance between disconnected components.

Second, it allows different disconnectivity levels in the network, after removing critical nodes, using the fraction β.

Similarly, Shen et al. [37,38] assessed network vulnerability by studying the *CNP* variant on three important classes of graphs, namely unit-disk graphs, used to represent ad hoc wireless networks [147], power-law graphs, used to represent social and biological networks [148], and dynamic graphs. Authors showed the efficiency of the *CNP* variant to assess the vulnerability on real-world networks.

Also, Nguyen et al. [39] explored the vulnerability of the interdependent-power networks, characterized by cascading failures on their interdependent communication networks. We note that the role of a node may be totally different between single and interdependent networks with respect to the vulnerability assessment, and also that some existing approaches for identifying critical nodes on single networks fail to cast the cascading failures in interdependent networks. Authors proposed the use of the *MinMaxC* variant to assess the vulnerability of this kind of network, and showed its efficiency compared with the previous approaches [149–151].

For analyzing network vulnerability in the case of unexpected events, called $N - k$ contingency, Fan et al. [152] introduced a new method based on the critical nodes concept. We recall that the $N - k$ contingency starts with a contingency selection step, where k failed components have to be selected. Authors in [152] proposed to select critical nodes as the k-contingencies using the *CNP* variant, and showed the efficiency of this approach.

Communication network study. In telecommunication networks, identifying critical nodes may be used for both defensive and offensive approaches. Indeed, these nodes are those that must either be destroyed to disable communication capacity in the adversary network [153,154], or kept intact to avoid any attempt to neutralize the own communication network. For instance, for the offensive approach, applications can be found on terrorist and insurgent networks where the objective is to break down communication between individuals by removing some of them [155,153,154]. This is what Arulselvan et al. [137] explored to destroy communications on the terrorist network established by Krebs [155] using the *CNP* and the *CC-CNP* variants. Note that Krab's terrorist network depicts the relationships between the terrorists constructed after the 9/11 attacks. Furthermore, Commander et al. [139] formulated the *Wireless Network Jamming Problem* as a *CC-CNP* problem, where critical nodes are those which must be jammed. Then, they aimed at minimizing the total jamming cost such that the connectivity index of each node does not exceed a predefined level L. We recall that the *Wireless Network Jamming Problem* aims to determine the optimal number and locations for a set of jamming devices in order to neutralize an adversary wireless communication network.

Complex network analysis. In the literature, the study of complex networks is considered both for analyzing network structure and for studying phenomena and event behavior. Considering a social network, which is an example of complex networks, detection of communities is an example of analyzing the network structure, while examination of contagion spread, product recommendation, trust propagation, information diffusion, etc., are examples of studying phenomena. Identifying critical nodes is very useful for studying such a network. On the one hand, it is useful for network structure analysis, since both individuals and ties between individuals do not have the same importance or the same influence. This was proved by Fan et al. [134], where they used the *CNDP* combined with the graph partition problem to analyze complex networks and find the influential individuals within the network communities.

On the other hand, it is useful for studying phenomena occurring in the network, since nodes do not have the same role. Critical nodes are those that should be destroyed for an outbreak of undesirable events or maintained for an outbreak of desirable ones. An example of the former is immunization against epidemics, where we ask for a specific number of individuals, the vaccination of which limits spreading of the epidemic among the population. An example of the latter is advertising products, where we ask for individuals who maximize the diffusion of recommendations about a product.

Through many works, the efficiency of using the *CNDP* in studying such phenomena has been proven by studying one of the most interesting phenomena in complex networks, namely *diffusion* [156–158]. Examples of diffusions in networks are epidemic propagation in populations, rumor spreading in social networks, virus diffusion in computer networks, etc. Note that the study of this process has been considered for different objectives, including blocking contagion diffusion [159,160], early detection of diffusion outbreaks [156,135] and locating sources of diffusion [158,161,162].

Considering the diffusion blocking objective, using critical nodes has turned out to be a useful tool for containing and blocking the spreading of contagion. Indeed, in immunization against epidemics [163], Arulselvan et al. [33] provided an efficient strategy to immunize populations against diseases using the *CNP* variant. They suggested vaccinating critical nodes in order to stop infection propagation at minimum cost. They showed the efficiency of the *CNP* in such a problem, and proved that the proposed strategy is better than the classical ones introduced in the literature [163,164]. Moreover, He et al. [40] provided exact and approximation algorithms for controlling infection in social networks by studying a general version of the *CNP*, where they asked for a subset of both critical nodes and links to be secured.

In the case of complex contagion spreading, where an individual in the network acquires a social contagion through interaction with $t > 1$ other affected individuals [165,136], the *CC-CNP* variant may be a useful parameter, where the constraint on the component size is t. Thus, once the critical nodes have been deleted, each connected component has at most t nodes, meaning that there are two cases: either all t nodes are affected and then there is no contagion spread since the t nodes are disconnected from the remaining network, or at least one node is unaffected in each component, and thus $t - 1 < t$ are affected, so there is no contagion spread (since a node acquires the contagion through interaction with t other affected nodes).

With respect to diffusion source identification, Lalou et al. [166] showed the efficiency of the *CC-CNP* variant in localizing the diffusion source. They first designed an observation model using critical nodes (identified by *CC-CNP*) as observers to acquire information about diffusion (e.g. the arrival-time of diffusion). They then used a statistical method exploiting this information in order to estimate the source node. Also, identifying critical nodes can be an efficient method for early detection of diffusion in the network. Indeed, considering contamination in a water distribution network, where we seek for the best nodes on which sensors must be placed to quickly detect contaminants, placing sensors on critical nodes identified using the *CC-CNP* or the *MinMaxC* variant ensures that contamination is detected after at most L hops in the network, where L is the component cardinality constraint of the considered variant.

Security issues. Protecting network applications against malicious behaviors is one of the main concerns when designing network applications. A secured application must take into account any network weaknesses that an attacker may exploit. One of the major weaknesses is network disconnectivity. Many systems, such as *P2P* and ad hoc networks, expand in a decentralized manner which usually results in a weak topology that can be easily fragmented by targeting a few nodes. These nodes are those that maintain the

whole network connectivity, known as the critical nodes. Thus, for these systems, exploiting these nodes, by studying the *CNDP*, is an efficient way to design robust and secured applications. When designing such applications using the *CNDP*, two approaches considered : the preventive approach and the reactive approach.

In the preventive approach, we usually seek to design a secured application before running it. We note that network application design is often based on the selection of a kernel set of nodes, such as the maximum independent set or dominating set. Therefore, this is based on the hypothesis that the more critical nodes there are in the kernel, the more the application is vulnerable, and, conversely, the fewer critical nodes there are in the kernel, the harder this application is to be destroyed: the application security is proportional to the selected kernel. Therefore, using the *CNDP* is a very nice approach to be explored here, and a kernel is said to be secured if it has as few critical nodes as possible. In the second approach, we seek to predict potential attacks before they occur and reconstruct the destroyed nodes later. In this case, since the targeted nodes are generally the critical ones, identifying them is an efficient way to react to any attack.

Transportation. In road network planning problems [167], traffic resource managers are concerned with the relative importance or criticality of various nodes in a road network. In this area, identification of critical nodes can be used to plan good emergency evacuations in disaster cases [168].

Applications of the *CNDP* are not limited to what has been mentioned above, and there are many other applications in several other areas.

8. Conclusion

This survey deals with one of the most important issues, namely the problem of finding critical nodes in networks. Critical nodes are those the deletion of which disconnects the network according to some predefined connectivity metrics. This problem has attracted much attention in recent years, and many variants have been defined in the literature depending on the connectivity metric to optimize once the nodes have been removed. The problem is generally NP-complete even on some particular graph classes. In this work, we first classified different variants tackled in the literature on two main classes, before reviewing and summarizing different results for each variant. By establishing relationships between different variants, we deduced new results and complexity bounds. We concluded the survey by a discussion and a proposal of different important open issues for future consideration.

References

[1] D. Kempe, J. Kleinberg, É. Tardos, Influential nodes in a diffusion model for social networks, in: Automata, Languages and Programming, Springer, 2005, pp. 1127–1138.
[2] H. Corley, D.Y. Sha, Most vital links and nodes in weighted networks, Oper. Res. Lett. 1 (4) (1982) 157–160.
[3] C.-T. Li, S.-D. Lin, M.-K. Shan, Finding influential mediators in social networks, in: Proceedings of the 20th International Conference Companion on World Wide Web, ACM, 2011, pp. 75–76.
[4] S.P. Borgatti, Identifying sets of key players in a social network, Comput. Math. Org. Theory 12 (1) (2006) 21–34.
[5] A. Hellwig, L. Volkmann, Maximally edge-connected and vertex-connected graphs and digraphs: A survey, Discrete Math. 308 (15) (2008) 3265–3296.
[6] P.-O. Fjällström, Algorithms for Graph Partitioning: A Survey, Vol. 3, Linköping University Electronic Press Linköping, 1998.
[7] A. Pothen, Graph partitioning algorithms with applications to scientific computing, in: Parallel Numerical Algorithms, Springer, 1997, pp. 323–368.
[8] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, C. Schulz, Recent advances in graph partitioning, in: Algorithm Engineering, Springer, 2016, pp. 117–158.
[9] O. Goldschmidt, D.S. Hochbaum, A polynomial algorithm for the *k*-cut problem for fixed k, Math. Oper. Res. 19 (1) (1994) 24–37.
[10] X. He, An improved algorithm for the planar 3-cut problem, J. Algorithms 12 (1) (1991) 23–37.
[11] M.-C. Costa, L. Létocart, F. Roupin, Minimal multicut and maximal integer multiflow: A survey, European J. Oper. Res. 162 (1) (2005) 55–69.
[12] N. Garg, V.V. Vazirani, M. Yannakakis, Primal-dual approximation algorithms for integral flow and multicut in trees, Algorithmica 18 (1) (1997) 3–20.
[13] J. Guo, R. Niedermeier, Fixed-parameter tractability and data reduction for multicut in trees, Networks 46 (3) (2005) 124–135.
[14] S. Chopra, M.R. Rao, On the multiway cut polyhedron, Networks 21 (1) (1991) 51–89.
[15] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, M. Yannakakis, The complexity of multiterminal cuts, SIAM J. Comput. 23 (4) (1994) 864–894.
[16] S. Chopra, J.H. Owen, Extended formulations for the A-cut problem, Math. Program. 73 (1) (1996) 7–30.
[17] A. Avidor, M. Langberg, The multi-multiway cut problem, Theoret. Comput. Sci. 377 (1–3) (2007) 35–42.
[18] T.N. Bui, C. Jones, Finding good approximate vertex and edge partitions is NP-hard, Inform. Process. Lett. 42 (3) (1992) 153–159.
[19] R.J. Lipton, R.E. Tarjan, A separator theorem for planar graphs, SIAM J. Appl. Math. 36 (2) (1979) 177–189.
[20] J. Fukuyama, NP-completeness of the planar separator problems, J. Graph Algorithms Appl. 10 (2) (2006) 317–328.
[21] C. de Souza, E. Balas, The vertex separator problem: algorithms and computations, Math. Program. 103 (3) (2005) 609–631.
[22] M.D. Biha, M.-J. Meurs, An exact algorithm for solving the vertex separator problem, J. Global Optim. 49 (3) (2011) 425–434.
[23] U. Benlic, J.-K. Hao, Breakout local search for the vertex separator problem, in: IJCAI, 2013, pp. 461–467.
[24] J. Chen, Y. Liu, S. Lu, An improved parameterized algorithm for the minimum node multiway cut problem, Algorithmica 55 (1) (2009) 1–13.
[25] S. Guillemot, Fpt algorithms for path-transversals and cycle-transversals problems in graphs, in: International Workshop on Parameterized and Exact Computation, Springer, 2008, pp. 129–140.
[26] G. Călinescu, C.G. Fernandes, B. Reed, Multicuts in unweighted graphs and digraphs with bounded degree and bounded tree-width, J. Algorithms 48 (2) (2003) 333–359.
[27] C. Papadopoulos, Restricted vertex multicut on permutation graphs, Discrete Appl. Math. 160 (12) (2012) 1791–1797.
[28] J. Guo, F. Hüffner, E. Kenar, R. Niedermeier, J. Uhlmann, Complexity and exact algorithms for vertex multicut in interval and bounded treewidth graphs, European J. Oper. Res. 186 (2) (2008) 542–553.
[29] D. Marx, Parameterized graph separation problems, Theoret. Comput. Sci. 351 (3) (2006) 394–406.
[30] R.G. Michael, S.J. David, Computers and Intractability: A Guide to the Theory of NP-Completeness, WH Freeman & Co., San Francisco, 1979.
[31] J.M. Lewis, M. Yannakakis, The node-deletion problem for hereditary properties is NP-complete, J. Comput. System Sci. 20 (2) (1980) 219–230.
[32] M. Yannakakis, Node-deletion problems on bipartite graphs, SIAM J. Comput. 10 (2) (1981) 310–327.
[33] A. Arulselvan, C.W. Commander, L. Elefteriadou, P.M. Pardalos, Detecting critical nodes in sparse graphs, Comput. Oper. Res. 36 (7) (2009) 2193–2200.
[34] V. Boginski, C.W. Commander, Identifying critical nodes in protein–protein interaction networks, Clustering Challenges Biol. Netw. (2009) 153–167.
[35] V. Tomaino, A. Arulselvan, P. Veltri, P.M. Pardalos, Studying connectivity properties in human protein–protein interaction network in cancer pathway, in: Data Mining for Biomarker Discovery, Springer, 2012, pp. 187–197.
[36] T.N. Dinh, M.T. Thai, Precise structural vulnerability assessment via mathematical programming, in: Military Communications Conference, 2011-MILCOM 2011, IEEE, 2011, pp. 1351–1356.
[37] Y. Shen, N.P. Nguyen, Y. Xuan, M.T. Thai, On the discovery of critical links and nodes for assessing network vulnerability, IEEE/ACM Trans. Netw. 21 (3) (2013) 963–973.
[38] Y. Shen, T.N. Dinh, M.T. Thai, Adaptive algorithms for detecting critical links and nodes in dynamic networks, in: Military Communications Conference, MILCOM 2012, Citeseer, 2012, pp. 1–6.
[39] D.T. Nguyen, Y. Shen, M.T. Thai, Detecting critical nodes in interdependent power networks for vulnerability assessment, Smart Grid, IEEE Trans. 4 (1) (2013) 151–159.
[40] J. He, H. Liang, H. Yuan, Controlling infection by blocking nodes and links simultaneously, in: Internet and Network Economics, Springer, 2011, pp. 206–217.
[41] J. Aspnes, K. Chang, A. Yampolskiy, Inoculation strategies for victims of viruses and the sum-of-squares partition problem, in: Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2005, pp. 43–52.
[42] J.A. Bondy, U.S.R. Murty, Graph Theory with Applications, Vol. 6, Macmillan, London, 1976.
[43] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Vol. 57, Elsevier, 2004.
[44] J.R. Jungck, G. Dick, A.G. Dick, Computer-assisted sequencing, interval graphs, and molecular evolution, Biosystems 15 (3) (1982) 259–273.

[45] A. Pal, M. Pal, Interval tree and its applications, Adv. Model. Optim. 11 (3) (2009) 211–224.
[46] S. Shen, J.C. Smith, Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs, Networks 60 (2) (2012) 103–119.
[47] N. Robertson, P.D. Seymour, Graph minors. II. Algorithmic aspects of tree-width, J. Algorithms 7 (3) (1986) 309–322.
[48] H.L. Bodlaender, Classes of Graphs with Bounded Tree-Width, Department of Computer Science, University of Utrecht, Netherlands, 1986.
[49] H.L. Bodlaender, A tourist guide through treewidth, Acta Cybernet. 11 (1–2) (1994) 1.
[50] H.L. Bodlaender, Treewidth: characterizations, applications, and computations, WG 4271 (2006) 1–14.
[51] R.G. Downey, M.R. Fellows, Parameterized Complexity, Springer Science & Business Media, 2012.
[52] J. Flum, M. Grohe, Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
[53] R. Cohen, K. Erez, D. Ben-Avraham, S. Havlin, Resilience of the internet to random breakdowns, Phys. Rev. Lett. 85 (21) (2000) 4626.
[54] R. Albert, H. Jeong, A.-L. Barabási, Error and attack tolerance of complex networks, Nature 406 (6794) (2000) 378–382.
[55] L. Faramondi, G. Oliva, R. Setola, F. Pascucci, A.E. Amideo, M.P. Scaparra, Performance analysis of single and multi-objective approaches for the critical node detection problem, in: International Conference on Optimization and Decision Science, Springer, 2017, pp. 315–324.
[56] P. Festa, P.M. Pardalos, M.G. Resende, Feedback set problems, in: Handbook of Combinatorial Optimization, Springer, 1999, pp. 209–258.
[57] B. Schieber, A. Bar-Noy, S. Khuller, The Complexity of Finding Most Vital Arcs and Nodes, University of Maryland at College Park, College Park, MD, USA, 1995.
[58] L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, V. Gurvich, G. Rudolf, J. Zhao, On short paths interdiction problems: total and node-wise limited interdiction, Theory Comput. Syst. 43 (2) (2008) 204–233.
[59] H.-A. Choi, K. Nakajima, C.S. Rim, Graph bipartization and via minimization, SIAM J. Discrete Math. 2 (1) (1989) 38–47.
[60] D. Marx, Chordal deletion is fixed-parameter tractable, Algorithmica 57 (4) (2010) 747–768.
[61] D. Marx, I. Schlotter, Obtaining a planar graph by vertex deletion, Algorithmica 62 (3–4) (2012) 807–822.
[62] A. Arulselvan, C.W. Commander, O. Shylo, P.M. Pardalos, Cardinality-constrained critical node detection problem, in: Performance Models and Risk Management in Communications Systems, Springer, 2011, pp. 79–91.
[63] T.N. Dinh, Y. Xuan, M.T. Thai, E. Park, T. Znati, On approximation of new optimization methods for assessing network vulnerability, in: INFOCOM, 2010 Proceedings IEEE, IEEE, 2010, pp. 1–9.
[64] S. Shen, J.C. Smith, R. Goli, Exact interdiction models and algorithms for disconnecting networks via node deletions, Discrete Optim. 9 (3) (2012) 172–188.
[65] B. Addis, M. Di Summa, A. Grosso, Identifying critical nodes in undirected graphs: complexity results and polynomial algorithms for the case of bounded treewidth, Discrete Appl. Math. 161 (16) (2013) 2349–2360.
[66] A. Berger, A. Grigoriev, R. Zwaan, Complexity and approximability of the k-way vertex cut, Networks 63 (2) (2014) 170–178.
[67] M. Di Summa, A. Grosso, M. Locatelli, Complexity of the critical node problem over trees, Comput. Oper. Res. 38 (12) (2011) 1766–1774.
[68] M. Lalou, M.A. Tahraoui, H. Kheddouci, Component-cardinality-constrained critical node problem in graphs, Discrete Appl. Math. 210 (2016) 150–163.
[69] L.C. Freeman, Centrality in social networks conceptual clarification, Soc. Netw. 1 (3) (1979) 215–239.
[70] G. Chartrand, Introduction to graph theory, Tata McGraw-Hill Education, 2006.
[71] D.A. Bader, S. Kintali, K. Madduri, M. Mihail, Approximating betweenness centrality, in: Algorithms and Models for the Web-Graph, Springer, 2007, pp. 124–137.
[72] U. Brandes, A faster algorithm for betweenness centrality, J. Math. Sociol. 25 (2) (2001) 163–177.
[73] M. Di Summa, A. Grosso, M. Locatelli, Branch and cut algorithms for detecting critical nodes in undirected graphs, Comput. Optim. Appl. 53 (3) (2012) 649–680.
[74] M. Oosten, J.H. Rutten, F.C. Spieksma, Disconnecting graphs by removing vertices: a polyhedral approach, Stat. Neerl. 61 (1) (2007) 35–60.
[75] A. Arulselvan, C.W. Commander, P.M. Pardalos, O. Shylo, Managing network risk via critical node identification, in: Risk Management in Telecommunication Networks, Springer, 2007.
[76] M. Ventresca, D. Aleman, A fast greedy algorithm for the critical node detection problem, in: Combinatorial Optimization and Applications, Springer, 2014, pp. 603–612.
[77] T.N. Dinh, M.T. Thai, Assessing attack vulnerability in networks with uncertainty, in: IEEE International Conference on Computer Communications (INFOCOM), IEEE, 2015.
[78] M. Ventresca, D. Aleman, A derandomized approximation algorithm for the critical node detection problem, Comput. Oper. Res. 43 (2014) 261–270.
[79] M. Ventresca, D. Aleman, A region growing algorithm for detecting critical nodes, in: Combinatorial Optimization and Applications, Springer, 2014, pp. 593–602.
[80] M. Ventresca, D. Aleman, A randomized algorithm with local search for containment of pandemic disease spread, Comput. Oper. Res. 48 (2014) 11–19.
[81] M. Ventresca, Global search algorithms using a combinatorial unranking-based problem representation for the critical node detection problem, Comput. Oper. Res. 39 (11) (2012) 2763–2775.
[82] A. Veremyev, O.A. Prokopyev, E.L. Pasiliao, An integer programming framework for critical elements detection in graphs, J. Combin. Optim. 28 (1) (2014) 233–273.
[83] R. Aringhieri, A. Grosso, P. Hosteins, R. Scatamacchia, A general evolutionary framework for different classes of critical node problems, Eng. Appl. Artif. Intell. 55 (2016) 128–145.
[84] D. Hermelin, M. Kaspi, C. Komusiewicz, B. Navon, Parameterized complexity of critical node cuts, Theoret. Comput. Sci. 651 (2016) 62–75.
[85] A. Veremyev, V. Boginski, E.L. Pasiliao, Exact identification of critical nodes in sparse networks via new compact formulations, Optim. Lett. 8 (4) (2014) 1245–1259.
[86] R. Aringhieri, A. Grosso, P. Hosteins, R. Scatamacchia, VNS solutions for the critical node problem, Electron. Notes Discrete Math. 47 (2015) 37–44.
[87] B. Addis, R. Aringhieri, A. Grosso, P. Hosteins, Hybrid constructive heuristics for the critical node problem, Ann. Oper. Res. 238 (1–2) (2016) 637–649.
[88] R. Aringhieri, A. Grosso, P. Hosteins, R. Scatamacchia, Local search metaheuristics for the critical node problem, Networks 67 (3) (2016) 209–221.
[89] W. Pullan, Heuristic identification of critical nodes in sparse real-world graphs, J. Heuristics 21 (5) (2015) 577–598.
[90] D. Purevsuren, G. Cui, N.N.H. Win, X. Wang, Heuristic algorithm for identifying critical nodes in graphs, Adv. Comput. Sci.: Int. J. 5 (3) (2016) 1–4.
[91] Y. Shen, M.T. Thai, Network vulnerability assessment under cascading failures, in: 2013 IEEE Global Communications Conference, GLOBECOM 2013, Atlanta, GA, USA, December 9-13, 2013, 2013, pp. 1526–1531.
[92] M. Ventresca, K.R. Harrison, B.M. Ombuki-Berman, An experimental evaluation of multi-objective evolutionary algorithms for detecting critical nodes in complex networks, in: European Conference on the Applications of Evolutionary Computation, Springer, 2015, pp. 164–176.
[93] E. Balas, C.C. de Souza, The vertex separator problem: a polyhedral investigation, Math. Program. 103 (3) (2005) 583–608.
[94] T.N. Dinh, M.T. Thai, Network under joint node and link attacks: Vulnerability assessment methods and analysis, IEEE/ACM Trans. Netw. 23 (3) (2015) 1001–1011.
[95] T.H. Cormen, Introduction to Algorithms, MIT press, 2009.
[96] M. Ventresca, K.R. Harrison, B.M. Ombuki-Berman, The bi-objective critical node detection problem, European J. Oper. Res. 265 (3) (2018) 895–908.
[97] R.K. Wood, Deterministic network interdiction, Math. Comput. Modelling 17 (2) (1993) 1–18.
[98] M.R. Garey, D.S. Johnson, L. Stockmeyer, Some simplified NP-complete problems, in: Proceedings of the Sixth Annual ACM Symposium on Theory of Computing, ACM, 1974, pp. 47–63.
[99] I. Dinur, S. Safra, On the hardness of approximating minimum vertex cover, Ann. of Math. (2005) 439–485.
[100] J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$, in: Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on, IEEE, 1996, pp. 627–636.
[101] Y.-S. Myung, H.-j. Kim, A cutting plane algorithm for computing k-edge survivability of a network, European J. Oper. Res. 156 (3) (2004) 579–589.
[102] P. Raghavan, C.D. Tompson, Randomized rounding: a technique for provably good algorithms and algorithmic proofs, Combinatorica 7 (4) (1987) 365–374.
[103] P. Hansen, N. Mladenović, J.A.M. Pérez, Variable neighbourhood search: methods and applications, Ann. Oper. Res. 175 (1) (2010) 367–407.
[104] H.R. Lourenço, O.C. Martin, T. Stützle, Iterated local search: Framework and applications, in: Handbook of Metaheuristics, Springer, 2010, pp. 363–397.
[105] M.G. Resendel, C.C. Ribeiro, Grasp with path-relinking: recent advances and applications, in: Metaheuristics: Progress As Real Problem Solvers, Springer, 2005, pp. 29–63.
[106] D. Purevsuren, G. Cui, M. Qu, N.N.H. Win, Hybridization of GRASP with exterior path relinking for identifying critical nodes in graphs, IAENG Int. J. Comput. Sci. 44 (2) (2017).
[107] F. Glover, Exterior path relinking for zero-one optimization, Int. J. Appl. Metaheuristic Comput. 5 (3) (2014) 1–8.
[108] Y. Zhou, J.-K. Hao, F. Glover, Memetic search for identifying critical nodes in sparse graphs, 2017. ArXiv Preprint arXiv:1705.04119.

[109] P. Moscato, C. Cotta, Memetic algorithms, Handb. Appl. Optim. 157 (2002) 168.
[110] Y. Zhou, J.-K. Hao, A fast heuristic algorithm for the critical node problem, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, ACM, 2017, pp. 121–122.
[111] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P.N. Suganthan, Q. Zhang, Multiobjective evolutionary algorithms: A survey of the state of the art, Swarm Evol. Comput. 1 (1) (2011) 32–49.
[112] A. Veremyev, O.A. Prokopyev, E.L. Pasiliao, Critical nodes for distance-based connectivity and related problems in graphs, Networks 66 (3) (2015) 170–195.
[113] R. Aringhieri, A. Grosso, P. Hosteins, R. Scatamacchia, A preliminary analysis of the distance based critical node problem, Electron. Notes Discrete Math. 55 (2016) 25–28.
[114] R. Aringhieri, A. Grosso, P. Hosteins, R. Scatamacchia, Polynomial and pseudo-polynomial time algorithms for different classes of the distance critical node problem, Discrete Appl. Math. (2018).
[115] N. Paudel, L. Georgiadis, G.F. Italiano, Computing critical nodes in directed graphs, in: 2017 Proceedings of the Ninteenth Workshop on Algorithm Engineering and Experiments, ALENEX, SIAM, 2017, pp. 43–57.
[116] S. Khot, Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique, SIAM J. Comput. 36 (4) (2006) 1025–1071.
[117] C.A. Barefoot, R. Entringer, H. Swart, Vulnerability in graphs -a comparative survey, J. Combin. Math. Combin. Comput. 1 (38) (1987) 13–22.
[118] K.S. Bagga, L.W. Beineke, W.D. Goddard, M.J. Lipman, R.E. Pippert, A survey of integrity, Discrete Appl. Math. 37 (1992) 13–28.
[119] P.G. Drange, M.S. Dregi, P. van't Hof, On the computational complexity of vertex integrity and component order connectivity, in: International Symposium on Algorithms and Computation, Springer, 2014, pp. 285–297.
[120] M. Yannakakis, Node-and edge-deletion NP-complete problems, in: Proceedings of the Tenth Annual ACM Symposium on Theory of Computing, ACM, 1978, pp. 253–264.
[121] M.S. Krishnamoorthy, N. Deo, Node-deletion NP-complete problems, SIAM J. Comput. 8 (4) (1979) 619–625.
[122] P. Alimonti, V. Kann, Hardness of approximating problems on cubic graphs, in: Algorithms and Complexity, Springer, 1997, pp. 288–298.
[123] E. Balas, C.S. Yu, On graphs with polynomially solvable maximum-weight clique problem, Networks 19 (2) (1989) 247–253.
[124] F. Gavril, Maximum weight independent sets and cliques in intersection graphs of filaments, Inform. Process. Lett. 73 (5) (2000) 181–188.
[125] R.B. Hayward, Weakly triangulated graphs, J. Combin. Theory Ser. B 39 (3) (1985) 200–208.
[126] M. Lozano, C. García-Martínez, F.J. Rodríguez, H.M. Trujillo, Optimizing network attacks by artificial bee colony, Inform. Sci. 377 (2017) 30–50.
[127] L. Faramondi, G. Oliva, F. Pascucci, S. Panzieri, R. Setola, Critical node detection based on attacker preferences, in: Control and Automation, MED, 2016 24th Mediterranean Conference on, IEEE, 2016, pp. 773–778.
[128] L. Faramondi, G. Oliva, R. Setola, F. Pascucci, A.E. Amideo, M.P. Scaparra, Performance analysis of single and multi-objective approaches for the critical node detection problem, in: International Conference on Optimization and Decision Science, Springer, 2017, pp. 315–324.
[129] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Global Optim. 39 (3) (2007) 459–471.
[130] L. Faramondi, R. Setola, S. Panzieri, F. Pascucci, G. Oliva, Finding critical nodes in infrastructure networks, Int. J. Crit. Infrastruct. Prot. (2017).
[131] T.A. Davis, Y. Hu, The university of florida sparse matrix collection, ACM Trans. Math. Softw. 38 (1) (2011) 1.
[132] M. Xiao, Simple and improved parameterized algorithms for multiterminal cuts, Theory Comput. Syst. 46 (4) (2010) 723–736.
[133] S.E. Schaeffer, Graph clustering, Comput. Sci. Rev. 1 (1) (2007) 27–64.
[134] N. Fan, P.M. Pardalos, Robust optimization of graph partitioning and critical node detection in analyzing networks, in: Combinatorial Optimization and Applications, Springer, 2010, pp. 170–183.
[135] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, N. Glance, Cost-effective outbreak detection in networks, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2007, pp. 420–429.
[136] C.J. Kuhlman, V.A. Kumar, M.V. Marathe, S. Ravi, D.J. Rosenkrantz, Finding critical nodes for inhibiting diffusion of complex contagions in social networks, in: Machine Learning and Knowledge Discovery in Databases, Springer, 2010, pp. 111–127.
[137] A. Arulselvan, Network Model for Disaster Management, University of Florida, 2009.
[138] D. Callahan, P. Shakarian, J. Nielsen, A.N. Johnson, Shaping operations to attack robust terror networks, in: Social Informatics, SocialInformatics, 2012 International Conference on, IEEE, 2012, pp. 13–18.
[139] C.W. Commander, P.M. Pardalos, V. Ryabchenko, S. Uryasev, G. Zrazhevsky, The wireless network jamming problem, J. Combin. Optim. 14 (4) (2007) 481–498.
[140] T. Pawson, P. Nash, Protein–protein interactions define specificity in signal transduction, Genes Dev. 14 (9) (2000) 1027–1047.
[141] C. Prieto, J. De Las Rivas, APID: agile protein interaction DataAnalyzer, Nucleic Acids Res. 34 (suppl. 2) (2006) W298–W302.
[142] J. Westermarck, J. Ivaska, G.L. Corthals, Identification of protein interactions involved in cellular signaling, Mol. Cell. Proteomics 12 (7) (2013) 1752–1763.
[143] C. Yan, F. Wu, R.L. Jernigan, D. Dobbs, V. Honavar, Characterization of protein–protein interfaces, Protein J. 27 (1) (2008) 59–70.
[144] H. Jhoti, A.R. Leach, Structure-based Drug Discovery, Springer, 2007.
[145] T. Liljefors, P. Krogsgaard-Larsen, U. Madsen, Textbook of Drug Design and Discovery, CRC Press, 2002.
[146] T.H. Grubesic, T.C. Matisziw, A.T. Murray, D. Snediker, Comparative approaches for assessing network vulnerability, Int. Reg. Sci. Rev. 31 (1) (2008) 88–112.
[147] B.N. Clark, C.J. Colbourn, D.S. Johnson, Unit disk graphs, Discrete Math. 86 (1) (1990) 165–177.
[148] J. Kleinberg, The small-world phenomenon: An algorithmic perspective, in: Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, ACM, 2000, pp. 163–170.
[149] J. Gao, S.V. Buldyrev, H.E. Stanley, S. Havlin, Networks formed from interdependent networks, Nature Phys. 8 (1) (2012) 40–48.
[150] X. Huang, J. Gao, S.V. Buldyrev, S. Havlin, H.E. Stanley, Robustness of interdependent networks under targeted attack, Phys. Rev. E 83 (6) (2011) 065101.
[151] R. Parshani, S.V. Buldyrev, S. Havlin, Interdependent networks: Reducing the coupling strength leads to a change from a first to second order percolation transition, Phys. Rev. Lett. 105 (4) (2010) 048701.
[152] N. Fan, H. Xu, F. Pan, P.M. Pardalos, Economic analysis of the N- k power grid contingency selection and evaluation by graph algorithms and interdiction methods, Energy Syst. 2 (3–4) (2011) 313–324.
[153] M. Ovelgonne, C. Kang, A. Sawant, V. Subrahmanian, Covertness centrality in networks, in: Advances in Social Networks Analysis and Mining, ASONAM, 2012 IEEE/ACM International Conference on, IEEE, 2012, pp. 863–870.
[154] R.R. Petersen, C.J. Rhodes, U.K. Wiil, Node removal in criminal networks, in: Intelligence and Security Informatics Conference, EISIC, 2011 European, IEEE, 2011, pp. 360–365.
[155] V. Krebs, Uncloaking terrorist networks, First Monday 7 (4) (2002).
[156] N.A. Christakis, J.H. Fowler, Social network sensors for early detection of contagious outbreaks, PLoS One 5 (9) (2010) e12948.
[157] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, C. Faloutsos, Efficient sensor placement optimization for securing large water distribution networks, J. Water Resour. Plan. Manage. 134 (6) (2008) 516–526.
[158] P.C. Pinto, P. Thiran, M. Vetterli, Locating the source of diffusion in large-scale networks, Phys. Rev. Lett. 109 (6) (2012) 068702.
[159] R. Pastor-Satorras, A. Vespignani, Immunization of complex networks, Phys. Rev. E 65 (3) (2002) 036104.
[160] C.J. Kuhlman, G. Tuli, S. Swarup, M.V. Marathe, S. Ravi, Blocking simple and complex contagion by edge removal, in: Data Mining, ICDM, 2013 IEEE 13th International Conference on, IEEE, 2013, pp. 399–408.
[161] A.Y. Lokhov, M. Mézard, H. Ohta, L. Zdeborová, Inferring the origin of an epidemic with a dynamic message-passing algorithm, Phys. Rev. E 90 (1) (2014) 012801.
[162] D. Shah, T. Zaman, Rumors in a network: Who's the culprit?, IEEE Trans. Inform. Theory 57 (8) (2011) 5163–5181.
[163] R. Cohen, S. Havlin, D. Ben-Avraham, Efficient immunization strategies for computer networks and populations, Phys. Rev. Lett. 91 (24) (2003) 247901.
[164] Z. Tao, F. Zhongqian, W. Binghong, Epidemic dynamics on complex networks, Prog. Natural Sci. 16 (5) (2006) 452–457.
[165] D. Centola, M. Macy, Complex contagions and the weakness of long ties1, Amer. J. Sociol. 113 (3) (2007) 702–734.
[166] M. Lalou, H. Kheddouci, Least squares method for diffusion source localization in complex networks, in: International Workshop on Complex Networks and their Applications, Springer, 2016, pp. 473–485.
[167] P. Bovy, R. Thijs, Modelling for transportation systems planning, ISTE, DUP Satellite, 2002.
[168] B. Vitoriano, M.T. Ortuño, G. Tirado, J. Montero, A multi-criteria optimization model for humanitarian aid distribution, J. Global Optim. 51 (2) (2011) 189–208.