Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 28, no. 1, pp. 1–26 (2024)
DOI: 10.7155/jgaa.v28i1.2922

On Critical Node Problems with Vulnerable Vertices

Jannik Schestag[1] *Niels Grüttemeier*[2] *Christian Komusiewicz*[1]
Frank Sommer[1]

[1]Institute of Computer Science, Friedrich Schiller Universität Jena, Germany.
[2]Fraunhofer IOSB-INA Lemgo, Germany.

Submitted: August 2022 Accepted: March 2024 Published: May 2024

Article type: Regular paper Communicated by: M. Nöllenburg

Abstract. A vertex pair in an undirected graph is called *connected* if the two vertices are connected by a path. In the NP-hard Critical Node Problem (CNP), the input is an undirected graph G with integers k and x, and the question is whether one can transform G by deleting at most k vertices into a graph whose total number of connected vertex pairs is at most x. In this work, we introduce and study two NP-hard variants of CNP where a subset of the vertices is marked as *vulnerable*, and we aim to obtain a graph with at most x connected vertex pairs containing at least one vulnerable vertex. In the first variant, which generalizes CNP, we may delete vulnerable and non-vulnerable vertices. In the second variant, we may only delete non-vulnerable vertices.

We perform a parameterized complexity study of both problems. For example, we show that both problems are FPT with respect to $k + x$. Furthermore, in the case of deletable vulnerable nodes, we provide a polynomial kernel for the parameter $\mathrm{vc} + k$, where vc is the vertex cover number. In the case of non-deletable vulnerable nodes, we prove NP-hardness even when there is only one vulnerable node.

1 Introduction

Detecting important vertices in graphs is a central task in network analysis. There is an abundance of different formalizations of this natural task, many of which adopt the view that a vertex set is important if its removal severely affects the connectivity of the remaining graph [13]. One concrete

A preliminary version of this work appeared in the *Proceedings of the 33rd International Workshop on Combinatorial Algorithms (IWOCA '22)* [16]. The full version contains all missing proofs. Some of the results of this work are also contained in the first author's Master thesis [15]. The majority of work was done while all authors were at Philipps-Universität Marburg, Germany.

Frank Sommer: Supported by the DFG, project Efficient Algorithms for Group Centrality (EAGR), KO 3669/6-1.

E-mail addresses: j.t.schestag@uni-jena.de (Jannik Schestag) niels.gruettemeier@iosb-ina.fraunhofer.de (Niels Grüttemeier) c.komusiewicz@uni-jena.de (Christian Komusiewicz) frank.sommer@uni-jena.de (Frank Sommer)

formulation, known as the Critical Node Problem, measures connectivity by the number of connected pairs of vertices, that is, the number of pairs of vertices that are in the same connected component. The aim is to look for a set of vertices whose deletion decreases this number as much as possible.

Critical Node Problem (CNP)
Input: A graph $G = (V, E)$, and two integers $k, x \in \mathbb{N}$.
Question: Is there a vertex set $C \subseteq V$ of size at most k such that $G - C$ has at most x connected pairs of vertices?

One application of this formulation is to model the influence of vertices in the spreading of viruses in computer networks or social networks [13]. Taking the latter view, the entities represented by a set C that minimizes the number of connected pairs in $G-C$ would be good candidates for being vaccinated or removed from the network via other interventions. The number x of connected pairs would be a rough measure for the amount of virus spreading in the remaining network, as vertices that are connected to many other vertices are more likely to contract the virus. For some vertices in the network, however, it may be irrelevant whether they contract the virus, for example, because they are not prone to develop a severe disease in case of infection. Conversely, it may be critical that some vertices in the network are protected from the virus because they belong to a high-risk group. One way to model this aspect is to label some vertices as *vulnerable* and to consider only the number of connected pairs that contain at least one vulnerable vertex.

Definition 1 *Let $G = (V, E)$ be a graph and let A be a set of* vulnerable *vertices. A vertex pair $\{u, v\}$ is a* vulnerable connection *(with respect to A) in G if $\{u, v\} \cap A \neq \emptyset$ and u and v are in the same connected component of G. The A-*vulnerability *of G is the number of vulnerable connections of G.*

Note that the A-vulnerability of the subgraph $G' = (V', E')$ is always defined, even if $A \not\subseteq V'$. Replacing the number of connected pairs by A-vulnerability leads to the following problem definition.

Critical Node Problem with Vulnerable Nodes (CNP-V)
Input: A graph $G = (V, E)$, a vertex set A, and two integers $k, x \in \mathbb{N}$.
Question: Is there a vertex set $C \subseteq V$ of size at most k such that the A-vulnerability of $G - C$ is at most x?

Regarding the motivation of choosing persons who get a vaccine, one might be unable to vaccinate the vulnerable people themselves for medical reasons. In our graph setting, this means that vulnerable vertices may not be removed. This is modeled by the following problem.

Critical Node Problem with Non-Deletable Vulnerable Nodes (CNP-NDV)
Input:A graph $G = (V, E)$, a vertex set A, and two integers $k, x \in \mathbb{N}$.
Question: Is there a vertex set $C \subseteq V \setminus A$ of size at most k such that the A-vulnerability of $G - C$ is at most x?

The set C in the problem definition is called a *critical node cut*. We study the parameterized complexity of these two problems with respect to several natural parameters.

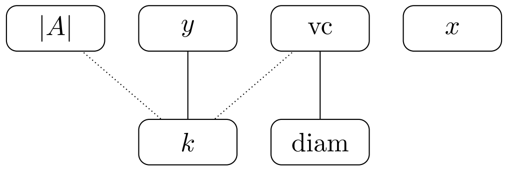

Figure 1: Dependencies between the considered parameters after preprocessing (see Section 2). If two parameters are connected by a solid line, then the parameter above is never smaller than the parameter below; if they are connected by a dotted line, then this relation holds only for CNP-V.

Related Work. Arulselvan et al. [4] showed that CNP is NP-complete; the NP-hardness follows also directly from the fact that CNP is a generalization of Vertex Cover ($x = 0$). As a consequence, CNP is NP-hard even on subcubic graphs. CNP is also NP-hard on split and bipartite graphs [1] and on power-law graphs [17]. In contrast, CNP can be solved in polynomial time on trees [6] and, more generally, on graphs with constant treewidth [1]. The parameterized complexity of CNP has been studied with respect to the parameters k, x, and the treewidth tw of G [12]: On the negative side, CNP is W[1]-hard with respect to k [12] or tw [12], and even with respect to $k + \mathrm{tw}$ [2]. On the positive side, the problem is FPT with respect to $k + x$ [12] and with respect to the parameter y [12] which is defined as $\ell - x$, where ℓ is the number of connected pairs in G. In other words, for CNP, y is the minimum number of connected pairs we want to remove by deleting k vertices.

Other formulations of graph modifications for limiting disease spreading consider for example edge deletions and limiting the size of the largest remaining connected component [8]. For an overview of different formulations of critical vertex detection, refer to the survey of Lalou et al. [13].

Our Results. We study the parameterized complexity of the problems CNP-V and CNP-NDV with respect to a number of natural parameters. An overview of the parameters in our study and their relations is shown in Figure 1. Our main findings are as follows (an overview is given in Table 1).

We transfer the FPT-algorithm for $k + x$ from CNP to the two new problems (Corollary 2, Theorem 5, Corollary 1). We then show that, while being solvable in polynomial time for constant values of x (Proposition 2), CNP-NDV is W[1]-hard with respect to x even when $|A| = 1$ (Theorem 2). In contrast, CNP-V is solvable in polynomial time for constant $|A|$ (Proposition 3) and NP-hard already for $x = 0$ [12]. Thus, the complexity of the two problems differs quite drastically with respect to very natural parameters.

We also study parameterization by the number y of deleted connected pairs containing at least one vulnerable vertex. Formally, let ρ be the number of connected pairs containing at least one vulnerable vertex. Then, $\rho = x + y$. Note that for CNP-V and CNP-NDV the definition of y is different than for CNP. The reason for this is that in CNP-V and CNP-NDV we are only interested in connected pairs containing at least one vulnerable vertex. We also observe a drastically different behavior for both problems by parameterization with y: CNP-V has a subexponential FPT-algorithm (Theorem 6) while CNP-NDV is W[1]-hard even with respect to $k + y$ (Theorem 7). We remark that the algorithm for CNP-V with subexponential running time for parameter y improves on a previous algorithm for CNP with exponential running time in y [12].

Finally, we consider parameterizations using the vertex cover number vc of G. This is moti-

Table 1: Overview of our results.

Parameter	CNP-V	CNP-NDV
x	NP-hard for $x = 0$ [12]	W[1]-hard (Thm. 2)
		XP (Prop. 2)
y	FPT (Thm. 6)	W[1]-hard (Thm. 7)
	No poly kernel [12]	XP (Prop. 3)
k	W[1]-hard [12]	W[1]-hard (Thm. 7)
	XP (Prop. 1)	XP (Prop. 1)
$k + x$	FPT (Cor. 2, Thm. 5)	FPT (Cor. 1)
$k + y$	FPT (Thm. 6)	W[1]-hard (Thm. 7)
$\|A\|$	XP (Prop. 3)	NP-hard for $\|A\| = 1$ (Thm. 2)
$\|A\| + x$	FPT (Cor. 4)	
vc	FPT (Thm. 8)	FPT (Thm. 8)
vc $+ x$	poly kernel (Thm. 9)	FPT (Thm. 8)
vc $+ k + x$		poly kernel (Thm 9)

vated by the fact that CNP is W[1]-hard with respect to the treewidth tw [2,12] and thus larger structural parameters need to be considered. We show that both problems are FPT with respect to vc (Theorem 8), and provide polynomial kernels for both problems parameterized by vc $+x$ (for CNP-V) and vc $+k + x$ (for CNP-NDV), respectively (Theorem 9).

Further FPT results for parameters such as the neighborhood diversity of G or $|V \setminus A|$ have been obtained in the first author's Master thesis [15].

Preliminaries. For two integers p and q, $p \leq q$, we denote $[p, q] := \{p, \ldots, q\}$. We consider undirected simple graphs $G = (V, E)$ and let $V(G)$ denote the vertex set and $E(G)$ the edge set of a graph G. We use n_G to denote the number of vertices and m_G to denote the number of edges of G, we omit the subscript if G is clear from context. For a vertex set S, we let $N(S) = \{u \mid \{u, v\} \in E(G), v \in S\} \setminus S$ and $N[S] := S \cup N(S)$ denote the *open* and *closed neighborhood of* S, respectively. For a vertex v, we denote $N(v) := N(\{v\})$ and $N[v] := N[\{v\}]$. For a vertex set S, we let $G[S] := (S, \{\{u, v\} \in E(G) \mid u, v \in S\})$ denote the *subgraph induced by* S, and $G - S := G[V(G) \setminus S]$ denote the subgraph of G obtained by deleting S and its incident edges.

Two vertices $u, v \in V$ are *connected*, if G contains a path P starting in u and ending in v. A graph $G = (V, E)$ is *connected*, if every two vertices $u, v \in V$ are connected. A *connected component* Z of a graph is a maximal vertex set such that $G[Z]$ is connected.

A *vertex cover* of a graph $G = (V, E)$ is a subset $S \subseteq V$ such that every edge of G has at least one endpoint in S. For a graph G, the *vertex cover number* vc(G) is the size of a smallest vertex cover of G. If it is clear from the context to which graph we refer, we write vc instead of vc(G).

A *branching rule* for some problem L is a computable function that maps an instance I of L to a tuple of instances $(I_1, \ldots, I_t)$ of L. A branching rule is called *correct* if I is a yes-instance of L if and only if there is some $i \in \{1, \ldots, t\}$ such that I_i is a yes-instance of L. A *reduction rule* for some problem L is a computable function that maps an instance I of L to an instance I' of L. In some reduction rules, we write *return yes* or *return no*, which formally means that we return a constant-size yes-instance or no-instance, respectively.

Parameterized Complexity is the analysis of the complexity of problems depending on the input size $|I|$ and a problem parameter k [5,7]. A parameterized problem L is in XP if there is an algorithm with running time $|I|^{f(k)}$ that solves L; such an algorithm is called an XP-algorithm. That is, a problem is in XP if it can be solved in polynomial time whenever the parameter is constant. A parameterized problem L is *fixed-parameter tractable (FPT)* if there exists an algorithm with running time $f(k) \cdot |I|^{\mathcal{O}(1)}$ for some computable function f that solves L; such an algorithm is called an FPT-algorithm. An important tool in the development of parameterized algorithms is *problem kernelization*, which is a polynomial-time preprocessing by reduction rules. Formally, a problem L admits a problem kernel if there exists an algorithm that given any instance (I, k) of L computes an equivalent instance (I', k') of L in polynomial time such that *a)* $k' \leq k$ and *b)* $|I'| \leq g(k)$. Here, g is some computable function only depending on k. The function g is called *kernel size*. A kernel has *polynomial size* if g is a polynomial.

A *parameterized reduction* of a parameterized problem L to a parameterized problem L' is an algorithm that maps any instance (I, k) of L in $f(k) \cdot |I|^{\mathcal{O}(1)}$ time to an equivalent instance (I', k') of L' such that $k' \leq g(k)$ for some computable functions f and g. If a parameterized problem is W[1]-hard, then it is assumed to be not fixed-parameter tractable. If there is a parameterized reduction from a W[1]-hard problem L to a parameterized problem L', then the problem L' is W[1]-hard as well.

Organization. In Section 2 we provide simple basic observations on the studied problems. Afterwards, we consider the parameterized complexity. In Section 3.2 we study parameterization by the target vulnerability x. In Section 4, we study parameterization by the number of vulnerable connections y that shall be removed from the graph. Finally, in Section 5, we study structural parameterization by the vertex cover number of the input graph.

2 Basic Observations

In this section, we provide basic observations which we use throughout this work. We first describe how to compute the A-vulnerability of a given graph and a set of vulnerable vertices. Afterwards, we consider simple reduction rules, so that we may use additional assumptions regarding the input instances. Finally, we establish a technical term of *component information* and show that CNP-NDV is NP-hard.

Vulnerability. First, observe that the A-vulnerability of a graph can be computed in linear time via depth-first search.

Lemma 1 *Let $G = (V, E)$ and let $A \subseteq V$. The A-vulnerability of G can be computed in $\mathcal{O}(n + m)$ time.*

Proof: In $\mathcal{O}(n + m)$ time all connected components can be computed using depth-first search. Then, for every connected component K, by iterating over all vertices of the connected component, the number of vulnerable vertices d and the size $c := |K|$ of the connected component K can be computed in linear time. The A-vulnerability of the connected component K is

$$\binom{d}{2} + d \cdot (c - d).$$

The A-vulnerability of G is the sum of the A-vulnerability of the connected components. Altogether, we can compute the A-vulnerability of any given graph in $\mathcal{O}(n+m)$ time. □

For a constant k, CNP-V and CNP-NDV can thus be solved in polynomial time by trying all $\mathcal{O}(n^k)$ possibilities of deleting k vertices (in the case of CNP-NDV only deletions in $V \setminus A$ are considered).

Proposition 1 CNP-V *and* CNP-NDV *can be solved in* $\mathcal{O}(n^k \cdot (n+m))$ *time.*

Recall that a *critical node cut* C is the set of removed vertices from the problem. For CNP-NDV at most x non-vulnerable vertices can be connected to vulnerable vertices in $G - C$. Thus, one can find a critical node cut by considering all $\mathcal{O}(n^x)$ possible sets B for these vertices, deleting all neighbors of $A \cup B$, and checking whether the number of deletions is at most k and the A-vulnerability of the resulting graph is at most x. definition.

Proposition 2 CNP-NDV *can be solved in* $\mathcal{O}(n^x \cdot (n+m))$ *time.*

Reduction Rules. We provide a collection of simple reduction rules for CNP-V and CNP-NDV that establish useful instance properties and parameter relations. The first rule removes trivial components from the input.

Reduction Rule 1 *Let* $I := (G, A, k, x)$ *be an instance of* CNP-V *or* CNP-NDV *and let* Z *be a connected component of* G*. If* Z *contains no vulnerable vertex or* Z *is an isolated vulnerable vertex, then delete* Z *from* G*.*

Reduction Rule 1 is safe since no vertex of Z is part of a vulnerable connection. For the remainder of this work, we assume that all instances of CNP-V and CNP-NDV are reduced with respect to Reduction Rule 1. The next rule identifies instances of CNP-V and CNP-NDV that are trivial because k is sufficiently large.

Reduction Rule 2 *a)* *Let* (G, A, k, x) *be an instance of* CNP-V*. If* $y \leq k$*, then return* yes*.*

b) *Let* (G, A, k, x) *be an instance of* CNP-NDV *such that* $y \leq k$*. If* $|V \setminus A| \geq y$*, then return* yes*. If* $|V \setminus A| < y$*, check if the number of vulnerable connections in* $G - (V \setminus A)$ *is at most* x*. If this is the case, return* yes*. Otherwise, return* no*.*

Lemma 2 *Reduction Rule 2 is safe and can be exhaustively applied in* $\mathcal{O}(n+m)$ *time.*

Proof: *Safeness:* Since the instance is reduced with respect to Reduction Rule 1, every vertex of the graph is in at least one vulnerable connection. Next, we show *a)* and *b)* separately.

First, we prove *a)*. We iteratively do the following y-times: If there is a vertex v such that at least one vulnerable connection contains v, then remove v from G. If there is no such vertex or G is empty, no vulnerable connection remains, and we have a yes-instance. Otherwise, we removed $y < k$ vertices from G and we thereby removed y vulnerable connections from G. Thus, we have a yes-instance.

Second, we show *b)*. Suppose that $|V \setminus A| \geq y$. Since Reduction Rule 1 is applied exhaustively, for each vertex in $|V \setminus A|$ we remove from G, we remove at least one vulnerable pair from G. Thus, we have a yes-instance. Otherwise, we have $|V \setminus A| < y$. Since we can only remove vertices in $V \setminus A$, we can now check whether $G[A]$ has at most x vulnerable connections.

Running Time: Clearly, this rule can be exhaustively applied in linear time. □

In the remainder of this work, we assume that Reduction Rule 1 has been applied exhaustively. Thus, we may assume $y > k$ throughout the rest of this work.

In the case of CNP-V, we can identify a further class of yes-instances. An instance of CNP-V with $|A| \leq k$ is a trivial yes-instance since adding all vulnerable vertices to a critical node cut destroys all vulnerable connections.

Reduction Rule 3 *Let (G, A, k, x) be an instance of* CNP-V*. If $|A| \leq k$, then return* yes.

The final rule deals with the case where one vertex has too many vulnerable neighbors. The idea behind the rule is that a vertex that causes too many vulnerable connections in its neighborhood belongs to every possible critical node cut.

Reduction Rule 4 *a) If an instance (G, A, k, x) of* CNP-V *contains a vertex $v \in V$ with $|N(v) \cap A| > k + \sqrt{2x}$, then remove v from G and decrease k by* 1.

b) If an instance (G, A, k, x) of CNP-NDV *contains a vertex $v \in V \setminus A$ with $|N(v) \cap A| > \sqrt{2x}$, then remove v from G and decrease k by 1.*

Lemma 3 *Reduction Rule 4 is safe and can be applied exhaustively in $\mathcal{O}(m + n)$ time.*

Proof: *Safeness:* The safeness of Reduction Rule 4 follows from the observation that, if a vertex v has more than $\sqrt{2x}$ vulnerable vertices as neighbors, then there are more than $\sqrt{2x}$ vulnerable connections $\{v, w\}$ with $w \in N(v)$ and there are $\binom{\lceil \sqrt{2x} \rceil}{2}$ vulnerable connections $\{w_1, w_2\}$ with $w_1, w_2 \in N(v)$. Altogether there are at least

$$\left\lceil \sqrt{2x} \right\rceil + \binom{\lceil \sqrt{2x} \rceil}{2} = \frac{2 \lceil \sqrt{2x} \rceil + \lceil \sqrt{2x} \rceil \cdot (\lceil \sqrt{2x} \rceil - 1)}{2} \geq \frac{2x + \lceil \sqrt{2x} \rceil}{2} > x$$

vulnerable connections in G. Thus, every critical node cut contains v.

Running time: For every vertex in G, we can compute the number of neighbors in A in $\mathcal{O}(m+n)$ time. For CNP-NDV we then remove all vertices from $V \setminus A$ that have sufficiently many neighbors in A. This is an exhaustive application of the rule since the vertex removal does not change the number of neighbors in A and also not the degree threshold for removal. For CNP-V, the procedure is slightly more complicated: we put all the vertices in a bucket queue, that is, a dynamic data structure of elements equipped with integral priorities, according to their number of neighbors in A with values ranging from 0 to $|A|$. We then consider the entries in this bucket queue by decreasing values starting with $i = |A|$. As long as $i > k + \sqrt{2x}$, we choose some vertex v with value i, remove v from G, update the values for all neighbors of v in $\mathcal{O}(|N(v)|)$ time, and decrement k. If there is no vertex with value i, then we decrement i. Overall, this procedure takes $\mathcal{O}(m + n)$ time. □

Throughout the remainder of this work, we assume that all instances are reduced with regard to the reduction rules stated above. We will thus make the following assumptions about instances of CNP-V and CNP-NDV: First, let (G, A, k, x) be an instance of CNP-V. We assume that

- every connected component of G contains a vertex from A (Reduction Rule 1),

- $y > k$ (Reduction Rule 2),
- $|A| > k$ (Reduction Rule 3), and
- $|N(v) \cap A| \leq k + \sqrt{2x}$ for every vertex v of G (Reduction Rule 4 *a*)).

Next, let (G, A, k, x) be an instance of CNP-NDV. Then, we assume that

- every connected component of G contains a vertex from A (Reduction Rule 1),
- $y > k$ (Reduction Rule 2), and
- $|N(v) \cap A| \leq \sqrt{2x}$ for every vertex v of G (Reduction Rule 4 *b*)).

Recall that CNP-V and CNP-NDV can be solved in $\mathcal{O}(n^k \cdot (n+m))$ time due to Proposition 1. By the parameter relations stated above, we obtain the following.

Proposition 3 CNP-V *and* CNP-NDV *can be solved in* $\mathcal{O}(n^y \cdot (n+m))$ *time;* CNP-V *can be solved in* $\mathcal{O}(n^{|A|} \cdot (n+m))$ *time.*

Component Information. We next show that CNP-V and CNP-NDV are solvable in polynomial time if we have additional information about the connected components of the input graph. We apply this fact to obtain efficient algorithms for both problems when the connected components are small. Let $I := (G, A, k, x)$ be an instance of CNP-V or CNP-NDV, and let $Z_1, \dots, Z_t \subseteq V$ be the connected components of the input graph G. Intuitively, a *component information* $T[i, k']$ for some integers $i \in [1, t]$ and $k' \in [0, k]$ provides information on how small the number of vulnerable connections in Z_i can be after k' vertex deletions. Formally, $T[i, k']$ is defined as the minimal number of vulnerable connections in $G[Z_i] - S$ among all subsets $S \subseteq Z_i$ of size exactly k'. If there is no such subset S containing exactly k' vertices that may be removed, we set $T[i, k'] = \infty$. A table T containing all component information $T[i, k'] \neq \infty$ is called a *component table of the instance* I. We now show that CNP-V and CNP-NDV can be solved in polynomial time if we have a component table of the input instance. The algorithm was also described by Hermelin et al. [12] for CNP.

Lemma 4 *Given an instance* $I := (G, A, k, x)$ *of* CNP-V *or* CNP-NDV *and a component table* T *of* I*, we can check in* $\mathcal{O}(n \cdot k^2)$ *time whether* I *is a yes-instance.*

Proof: *Algorithm:* Let $Z_1, \dots, Z_t$ be the connected components of the input graph G. We show the lemma by providing a dynamic programming algorithm. In the dynamic programming table Q, each entry $Q[i, k']$ with $i \in [1, t]$ and $k' \in [0, k]$ stores the minimum number of vulnerable connections in $G[Z_1 \cup \dots \cup Z_i] - S$ among all subsets $S \subseteq Z_1 \cup \dots \cup Z_i$ of size exactly k'.

For the value $i = 1$, we set $Q[1, k'] := T[1, k']$ for every $k' \in [0, k]$. The recurrence to compute an entry for $i > 1$ is

$$Q[i, k'] := \min_{0 \leq k'' \leq k'} Q[i-1, k''] + T[i, k' - k''].$$

Here, we check all possibilities of k'' on how many deletions are done in the i-th connected component. If $Q[t, k] \leq x$ return yes. Otherwise, return no.

Running time: The table has $t \cdot (k+1) \in \mathcal{O}(n \cdot k)$ entries. Each entry can be computed in $\mathcal{O}(k)$ time by iterating over all possible values of k''. Consequently, the algorithm has a running time of $\mathcal{O}(n \cdot k^2)$. $\square$

Observe that, for an instance where the input graph has maximum component size c for some constant c, a component table can be computed in $\mathcal{O}(2^c \cdot (n+m)) = \mathcal{O}(n)$ time by iterating over every subset of each connected component. Note that $m \in \mathcal{O}(n)$ for instances with constant maximum component size.

Proposition 4 CNP-V *and* CNP-NDV *can be solved in* $\mathcal{O}(2^c \cdot m + n \cdot k^2)$ *time, where* c *is the size of the largest connected component of the input graph.*

Component information will be utilized in Theorem 6 of Section 4 to provide an efficient FPT-algorithm with respect to y.

NP-Hardness of CNP-NDV. In contrast to CNP-V, the problem CNP-NDV is not an obvious generalization of VERTEX COVER. We show the following by a simple reduction.

Theorem 1 CNP-NDV *is* NP*-hard on planar graphs, even if the input graph has maximum degree* 4.

Proof: We reduce from INDEPENDENT SET where the input is a graph G and an integer κ and the question is whether there exists a set X of κ vertices which are pairwise nonadjacent. INDEPENDENT SET is NP-hard on planar graphs with maximum degree 3 [11].

Construction: Let (G, κ) be an instance of INDEPENDENT SET where G is a planar graph with maximum degree 3. We construct an instance (G', A, k, x) of CNP-NDV as follows: Intuitively, we add a copy of each vertex in V to A, connecting each copy to its respective original by an edge. Formally, we set $V' := V \cup A$, where $A := \{a_v \mid v \in V\}$ and $E' := E \cup \{\{v, a_v\} \mid v \in V\}$. Furthermore, we set $k := n_G - \kappa$ and $x := \kappa$. Clearly, $(G' = (V', E'), A, k, x)$ can be computed in polynomial time. Note that every vertex in G', that is not in G has degree 1 and $G'[V]$ is isomorphic to G. Thus, G' is planar and has maximum vertex degree 4. It remains to show that the two instances are equivalent.

Correctness: Let (G, κ) be a yes-instance of INDEPENDENT SET. Then, there exists an independent set $I \subseteq V(G)$ of size κ. We show that the vertex set $V \setminus I$ is a critical node cut: The size of $V \setminus I$ is $n_G - \kappa$ and $V' \setminus (V \setminus I) = A \cup I$. As I is an independent set in G, I is an independent set in G' as well. Thus, there are exactly κ edges in the graph $G' - (V \setminus I)$. More precisely the edges remaining in $G' - (V \setminus I)$ are $\{\{u, a_u\} \mid u \in I\}$. Thus, the A-vulnerability of $G' - (V \setminus I)$ is κ. Hence, $(G' = (V', E'), A, k, x)$ is a yes-instance of CNP-NDV.

Conversely, let $(G' = (V', E'), A, k, x)$ be a yes-instance of CNP-NDV. Thus, there exists a critical node cut C of size $n_G - \kappa$. The graph $G' - C$ contains κ vertices that are not vulnerable and by the construction of G', each of them is adjacent to a vulnerable vertex. Thus, these are all vulnerable connections in $G' - C$. Therefore, $V(G) \setminus C$ is an independent set in G'. Thus, (G, κ) is a yes-instance of INDEPENDENT SET □

3 Parameterization by the Targeted Vulnerability

3.1 Hardness for Non-Deletable Vulnerable Vertices

First, we consider parameterization by x alone. CNP-V is NP-hard for $x = 0$ since it is a generalization of VERTEX COVER. Recall that in contrast CNP-NDV can be solved in polynomial time for constant x due to Proposition 2. We complement this result by showing that CNP-NDV is W[1]-hard with respect to x, even if G contains only one vulnerable vertex.

Theorem 2 CNP-NDV *is* W[1]*-hard with respect to the parameter x, even if $|A| = 1$ and G has diameter 2.*

Proof: We reduce from CUTTING AT MOST k VERTICES WITH TERMINAL. Here, the input is a graph G, a vertex $s \in V(G)$, and two integers k and t. The question is whether there exists a vertex set $S \subseteq V(G)$ of size at most k such that $s \in S$ and $|N(S)| \leq t$. Fomin et al. [9] showed that CUTTING AT MOST k VERTICES WITH TERMINAL is W[1]-hard with respect to k.

Construction: Let (G, s, k, t) be an instance of CUTTING AT MOST k VERTICES WITH TERMINAL. Let G' be the graph obtained by adding a universal vertex $\bar{u}$ to graph G. In polynomial time we compute the instance $(G', A := \{s\}, k' := t + 1, x := k - 1)$ of CNP-NDV in which $|A| = 1$. Because of the universal vertex $\bar{u}$, the diameter of G' is 2.

Correctness: Let (G, s, k, t) be a yes-instance of CUTTING AT MOST k VERTICES WITH TERMINAL. Let S with $|S| \leq k$ be the vertex set such that $s \in S$ and $|N(S)| \leq t$. We define a set $C \subseteq V(G')$ by $C := N(S) \cup \{\bar{u}\}$. Thus, $|C| \leq t + 1$. Furthermore, the connected component of s in $G' - C$ contains at most $|S| - 1 = k - 1 = x$ other vertices. Hence, there are at most x vulnerable connections in $G' - C$ and C is a critical node cut. We conclude that (G', A, k', x) is a yes-instance of CNP-NDV.

Conversely, let (G', A, k', x) be a yes-instance of CNP-NDV. Therefore, there exists a vertex set C of size at most $k' = t + 1$ such that in $G' - C$ there are at most x vulnerable connections. Thus, there are at most x other vertices in the connected component of s in $G' - C$. Now, we distinguish between the cases that $\bar{u}$ is in C or not.

Case 1: $\bar{u} \in C$. Define S as the connected component of s in $G' - C$. It follows that S has size at most $x + 1$ and is a subset of $V(G)$. The neighborhood of S in G is a subset of $C \setminus \{\bar{u}\}$ of size at most t. Thus, S is a solution of CUTTING AT MOST k VERTICES WITH TERMINAL.

Case 2: $\bar{u} \notin C$. There are at most $k' + x + 1$ vertices in G', because every vertex that has not been cut is then in a vulnerable connection with x. Thus, G has at most $k + t$ vertices and any set S of size k is a solution of CUTTING AT MOST k VERTICES WITH TERMINAL, as long as $s \in S$.

In both cases we proved that (G, s, k, t) is a yes-instance of CUTTING AT MOST k VERTICES WITH TERMINAL. □

3.2 A general FPT-Algorithm for $\mathbf{k + x}$

In the following, we provide an FPT-algorithm for CNP-V and CNP-NDV parameterized by $k+x$. To unify the description, we consider the following more general problem. Roughly speaking, we allow for a set of non-deletable vertices which do not have to be vulnerable.

CNP-VNDV
Input: A graph $G = (V, E)$, two sets A, N, and two integers $k, x \in \mathbb{N}$.
Question: Is there a vertex set $C \subseteq V \setminus N$ of size at most k such that the A-vulnerability of $G - C$ is at most x?

Observe that CNP-V is the special case of CNP-VNDV where $N = \emptyset$ and CNP-NDV is the special case of CNP-VNDV where $N = A$.

Hermelin et al. [12] showed that CNP can be solved in $\mathcal{O}(3^{k+x} \cdot (x^{k+2} + n))$ time. The idea of this algorithm is to branch for each edge $\{u, v\}$ whether one of u and v is deleted or whether this is one of the x remaining connections. In the following, we use similar ideas to provide two search tree algorithms for the more general CNP-VNDV. The first algorithm solves instances of CNP-VNDV with $A \subseteq N$ in $\mathcal{O}(2^{k+x} \cdot (n + m))$ time. This implies that CNP-NDV can be solved within the same running time. The second algorithm solves arbitrary instances of CNP-VNDV in $\mathcal{O}(3^{k+x} \cdot (n+m))$ time, which implies that CNP-V can be solved in $\mathcal{O}(3^{k+x} \cdot (n+m))$ time. Moreover, since CNP is a special case of CNP-V this improves over the algorithm for CNP by Hermelin et al. [12].

The first algorithm is based on the following branching rule. The idea of the rule is that for each deletable vertex, we may either decide to delete the vertex, which reduces the deletion budget by one, or to make this vertex non-deletable.

Branching Rule 1 *Let $I = (G, A, N, k, x)$ be an instance of* CNP-VNDV *and let $v \in V(G) \setminus N$. Then, I is a yes-instance of* CNP-VNDV *if and only if $I_1 = (G - \{v\}, A, N, k-1, x)$ or $I_2 = (G, A, N \cup \{v\}, k, x)$ is a yes-instance of* CNP-VNDV*.*

Lemma 5 *Branching Rule 1 is correct.*

Proof: Let I be a yes-instance of CNP-VNDV. Then, there is a set $C \subseteq V \setminus N$ such that $|C| \leq k$ and $G - C$ has A-vulnerability of at most x. If $v \in C$, then the instance I_1 is also a yes-instance, because the set $C' = C \setminus \{v\}$ is a critical node cut of I_1. Otherwise, if $v \notin C$, then I_2 is a yes-instance, as C is also a critical node cut for I_2.

Conversely, let I_1 or I_2 be a yes-instance of CNP-VNDV. First, suppose there exists a critical node cut C for the instance I_1. Then, the set $C' := C \cup \{v\}$ is also a critical node cut for I. Second, suppose that I_2 be a yes-instance of CNP-VNDV and let C be the corresponding critical node cut. It directly follows that C is also a critical node cut for I. □

The idea of the first algorithm, which works for the special case of CNP-VNDV with $A \subseteq N$, is now to apply Branching Rule 1 in such a way that in the second branch we increase the A-vulnerability of the subgraph that is induced by the non-deletable vertices. This allows us to bound the depth of the search tree.

Theorem 3 *An instance $I := (G, A, N, k, x)$ of* CNP-VNDV *with $A \subseteq N$ can be solved in time $\mathcal{O}(2^{k+x} \cdot (n+m))$.*

Proof: We use a search tree algorithm based on Branching Rule 1. Each node of the search tree corresponds to an instance of CNP-VNDV with $A \subseteq N$. At each search tree node, we perform the following steps. First, if $k < 0$ or the A-vulnerability of $G[N]$ is greater than x, we return no. Second, if the A-vulnerability of G is at most x, we return yes. Third, we choose a vertex $v \in V(G) \setminus N$ which is connected to some vertex of A via a path containing only internal vertices of N and branch on vertex v using Branching Rule 1 (see Figure 2 for an illustration).

The correctness of this algorithm can be seen as follows. The first two steps identify trivial no- or yes-instances and are obviously correct. For the third step, observe that such a vertex v must exist since 1) the A-vulnerability of G is larger than the A-vulnerability of $G[N]$ and every path connecting a vertex from A with a vertex from $V(G) \setminus N$ starts in N since $A \subseteq N$. Thus, the correctness of the third step follows from Branching Rule 1.

The running time bound can be seen as follows. Branching is only performed while $k \geq 0$ and $x \geq 0$. Hence, the depth of the search tree is at most $k + x$: in the first branch deleting v decreases k by one and in the second branch adding v to N increases the A-vulnerability of $G[N]$ by at least one. Thus, the search tree has a size of $\mathcal{O}(2^{k+x})$. Also, note that the steps at each search tree node can be performed in linear time. Hence, the overall running time follows. □

Corollary 1 CNP-NDV *can be solved in $\mathcal{O}(2^{k+x} \cdot (n+m))$ time.*

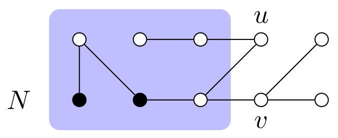

Figure 2: Illustration of the choice of the branching vertex in the proof of Theorem 3. Vulnerable vertices are black. The algorithm chooses either u or v as branching vertex. For both choices, adding the vertex to N increases the A-vulnerability of $G[N]$.

To provide an FPT-algorithm for the case when vulnerable vertices may be deleted, we make use of another branching rule. This is necessary because we may face the situation that there is no vertex in $V(G) \setminus N$ whose addition to N increases the A-vulnerability of $G[N]$. For example, this is the case when $N = \emptyset$. The new rule is applied on edges with at least one deletable vertex. It considers all possibilities to either delete an endpoint of the edge or to add the endpoints to the set of non-deletable vertices.

Branching Rule 2 *Let* $I := (G = (V, E), A, N, k, x)$ *be an instance of* CNP-VNDV *and let* $\{u, v\} \in E$ *be an edge of* G *with* $v \notin N$*. Then,* I *is a yes-instance of* CNP-VNDV *if and only if*

1. $I_1 := (G, A, N \cup \{u, v\}, k, x)$ *is a yes-instance of* CNP-VNDV*, or*
2. $I_2 := (G - \{v\}, A, N, k - 1, x)$ *is a yes-instance of* CNP-VNDV*, or*
3. $u \notin N$ *and* $I_3 := (G - \{u\}, A, N, k - 1, x)$ *is a yes-instance of* CNP-VNDV*.*

Lemma 6 *Branching Rule 2 is correct.*

Proof: Let I be a yes-instance of CNP-VNDV and let C be a critical node cut. If $v \in C$, then $C \setminus \{v\}$ is a critical node cut for the instance $(G - \{v\}, A, N, k - 1, x)$ and thus I_2 is a yes-instance. Now, if $u \in C$ then $u \notin N$. Consequently, the branching rule creates the instance I_3. The set $C \setminus \{u\}$ is a critical node cut for the instance $(G - \{u\}, A, N, k - 1, x)$ and thus I_3 is a yes-instance. Otherwise, if $u, v \notin C$, then I_1 is a yes-instance, as C is also a critical node cut for I_1.

Conversely, suppose that I_1 or I_2 or I_3 is a yes-instance of CNP-VNDV. First, assume that I_1 is a yes-instance of CNP-VNDV and let C be a corresponding critical node cut. Then, C is also a critical node cut for I. Second, if I_2 is a yes-instance with a critical node cut C, then because $v \notin N$, the set $C \cup \{v\}$ is a critical node cut for I. Finally, if I_3 is a yes-instance with a critical node cut C, then because of the condition for creating the instance I_3, we know that $u \notin N$. Thus, $C \cup \{u\}$ is is a critical node cut for I. □

Theorem 4 CNP-VNDV *can be solved in* $\mathcal{O}(3^{k+x} \cdot (n + m))$ *time.*

Proof: We use a search tree algorithm based on Branching Rule 2. Each node of the search tree corresponds to an instance of CNP-VNDV. Before branching, we exhaustively remove all degree-0 vertices. This is correct since these vertices are irrelevant for computing the A-vulnerability of any subgraph of G. At each search tree node, we perform the following steps. First, if $k < 0$ or the A-vulnerability of $G[N]$ is greater than x, we return no. Second, if the A-vulnerability of G is at most x, return yes. Third, we choose an edge $\{u, v\} \in E(G)$ such that the A-vulnerability of $G[N \cup \{u, v\}]$ is larger than the A-vulnerability of $G[N]$. Observe that such an edge contains at

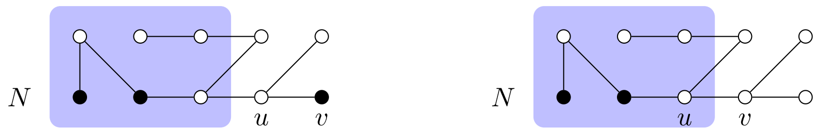

Figure 3: Illustration of the choice of the edge $\{u, v\}$ on which branching is performed in the proof of Theorem 4. Vulnerable vertices are black. Left: The algorithm may choose a vulnerable vertex v outside of N and some neighbor u. Right: If $A \subseteq N$, then the algorithm may choose any vertex v with a neighbor $u \in N$ that is connected in $G[N]$ to some vulnerable vertex.

least one node which is not contained in N. Hence, we may apply Branching Rule 2 on this edge (see Figure 3 for an illustration).

The correctness of this algorithm can be seen as follows. The first two steps identify trivial no- or yes-instances and are obviously correct. For the third step, we show that such an edge $\{u, v\}$ must exist. Then, the correctness of the third step follows from Branching Rule 2. If $V(G) \setminus N$ contains a vulnerable vertex v, then the edge $\{u, v\}$ where u is any neighbor of v may be chosen. Otherwise, $A \subseteq N$ and there exists a vertex v in $V(G) \setminus N$ which is connected to a vertex in A via a path P whose other vertices are all from N, because the A-vulnerability of $G[N]$ is strictly smaller than the A-vulnerability of G. We may branch on the edge $\{u, v\}$, where u is the neighbor of v on P. Altogether, this shows the correctness of the algorithm.

The running time bound can be seen as follows. Branching is only performed while $k \geq 0$ and $x \geq 0$. Hence, the depth of the search tree is at most $k + x$: In the first branch, adding $\{u, v\}$ to N increases the A-vulnerability of $G[N]$ by at least one; in the second and, possibly, third branch the value of k is decreased by one. Thus, the search tree has a size of $\mathcal{O}(3^{k+x})$ since each internal node has at most three children. Since the steps at each search tree node can be performed in $\mathcal{O}(n + m)$ time, the overall running time follows. □

Corollary 2 CNP-V *can be solved in* $\mathcal{O}(3^{k+x} \cdot (n + m))$ *time.*

Corollary 3 CNP *can be solved in* $\mathcal{O}(3^{k+x} \cdot (n + m))$ *time.*

3.3 An Alternative FPT-Algorithm for Deletable Vulnerable Vertices

In the following, we provide an algorithm that solves CNP-V in $\mathcal{O}((\frac{4}{3}x + 2)^k \cdot (m + n))$ time. This running time is preferable when x is much larger than k. The idea of the algorithm is that we search a set B of at most $\frac{4}{3}x + 2$ vertices of G such that the A-vulnerability of $G[B]$ is larger than x. Then, if there exists a critical node cut C, at least one vertex of B is in C.

Theorem 5 *An instance* $I := (G, A, k, x)$ *of* CNP-V *can be solved in* $\mathcal{O}((\frac{4}{3}x + 2)^k \cdot (m + n))$ *time.*

Proof: We use a search tree algorithm. Each node of the search tree corresponds to an instance of CNP-V. At each search tree node, first exhaustively perform Reduction Rule 1 which removes connected components that do not contribute to the A-vulnerability. Then, if $k < 0$, return no. Otherwise, if the A-vulnerability of G is at most x, return yes. If all remaining connected components have size at most 3, then return the result of the algorithm of Proposition 4.

Now, in the remaining case, compute a vertex set B on which we can branch as follows. Let $Z_1, \ldots, Z_q$ denote the connected components of G that have size at least 4. If $G[Z_1 \cup \ldots \cup Z_q]$ has A-vulnerability at most x, then set $B := Z_1 \cup \ldots \cup Z_q$. Otherwise, let $i \leq q$ be the smallest number such that $G[Z_1 \cup \ldots \cup Z_i]$ has A-vulnerability at least $x+1$. First, add $Z_1 \cup \ldots \cup Z_{i-1}$ and some vulnerable vertex v of Z_i to B. Then, add any vertex of Z_i to B that has a neighbor in B until $G[B]$ has A-vulnerability at least $x+1$. Afterwards, branch on B as follows: If $I_v := (G-\{v\}, A, k-1, x)$ is a yes-instance of CNP-V for some vertex v of B, then return yes. Otherwise, return no.

To show the correctness of the algorithm, it is sufficient to show that the above-described branching is correct.

Claim 1 *Let B be a vertex set computed as described above. Then, I is a yes-instance of* CNP-V *if and only if there exists a vertex $v \in B$ such that I_v is a yes-instance of* CNP-V.

Proof. Suppose that I is a yes-instance. We show that I has some critical node cut C that contains at least one vertex v of B. Then, I_v is a yes-instance. If the A-vulnerability of $G[B]$ is larger than x, then any critical node cut C of I must necessarily contain at least one vertex v of B. Otherwise, B contains by construction all vertices that are in connected components of size at least 4 in G. Let C be any critical node cut of I. Assume that $C \cap B = \emptyset$. Then, C contains only vertices of connected components of size at most 3. Let u be any vertex of C and let v be any vulnerable vertex in B. Then, the set $C' := (C \cup \{v\}) \setminus \{u\}$ is a critical node cut: The deletion of u in G decreases the A-vulnerability by at most 3 and the deletion of v decreases the A-vulnerability by at least 3, since v is vulnerable and in a connected component of size at least 4. Thus, C' is a critical node cut of G that contains at least one vertex of B.

Conversely, assume that I_v is a yes-instance of CNP-V for some vertex $v \in B$ and let $C \subseteq V(G-\{v\})$ be a critical node cut of I_v. Clearly, $C' = C \cup \{v\}$ is a critical node cut of I. $\diamond$

It remains to show the running time bound. At each search tree node, all steps until the computation of B can be performed in $\mathcal{O}(n+m)$ time. Now, the set B can also be computed in $\mathcal{O}(n+m)$ time: First, we can compute the A-vulnerability of all connected components Z_j, $1 \leq j \leq q$ in total time $\mathcal{O}(n+m)$. From this, we can compute the index i and the A-vulnerability of $G[Z_1 \cup \ldots \cup Z_{i-1}]$. Now, for each vertex u of Z_i that is added to B, we can compute in constant time the increase in A-vulnerability that is obtained by adding u. We may attribute the time needed to compute the instance I_v to the search tree node that is created by the recursive call for searching I_v. Hence, the total running time spent at each search tree node is $\mathcal{O}(n+m)$.

To bound the size of the search tree, first observe that the search tree has depth at most k, since one vertex is deleted in each recursive call and branching is only performed while $k \geq 0$. The number of recursive calls is $|B|$ and thus it remains to bound $|B|$.

Let $v_1, \ldots, v_i$ denote the initial vulnerable vertices added from each Z_i, and let $D := B \setminus \{v_1, \ldots, v_i\}$ denote the other vertices in B. Note that D may contain vulnerable as well as non-vulnerable vertices. We have $|D| \leq x+1$, since the addition of each element in D to B increases the A-vulnerability of B by at least one. Thus, $|B| \leq i + x + 1$. Now, since $G[Z_1 \cup \ldots \cup Z_{i-1}]$ has A-vulnerability at most x, we have that $|D \setminus Z_i| \leq x$, that is, the first $i-1$ components contain at most x vertices from D. Each of these $i-1$ connected components contains at least 3 vertices from D since we consider only connected components of size at least 4. Thus, $i-1 \leq x/3$. Consequently $|B| \leq i + x + 1 = i - 1 + x + 2 \leq \frac{4}{3}x + 2$.

Altogether, the size of the search tree is $\mathcal{O}((\frac{4}{3}x+2)^k)$. Thus, the overall running time of the algorithm is $\mathcal{O}((\frac{4}{3}x+2)^k \cdot (m+n))$. $\square$

After Reduction Rule 3 is applied, we can assume $|A| > k$ for instances of CNP-V. Hence, we also obtain the following.

Corollary 4 CNP-V *has an FPT-algorithm for the parameter $|A| + x$.*

4 Parameterization by the Decrease in Vulnerability

In this section, we consider the parametrization by $y := \ell - x$, where ℓ is the A-vulnerability of the input graph. In other words, y counts how many vulnerable connections shall be removed.

4.1 An FPT Algorithm for Deletable Vulnerable Vertices

CNP is fixed-parameter tractable with respect to y [12], based on the following observations: If some connected component has more than y vertices, then we have a yes-instance, as we can delete y vulnerable connections by removing a single vertex. Afterward, we may compute the component information in $\mathcal{O}(2^y \cdot y^2 \cdot (n+m))$ time and combine it using the dynamic programming algorithm presented also in Section 2. We now extend the FPT result to the more general CNP-V problem. Moreover, we improve the running time to a subexponential running time in y.

Theorem 6 CNP-V *can be solved in* $2^{\mathcal{O}(\sqrt{y}\log y)} \cdot n^{\mathcal{O}(1)}$ *time.*

Proof: Let $I := (G, A, k, x)$ be an instance of CNP-V and let $Z_1, \ldots, Z_t$ be the connected components of G. Recall that we assume that I is reduced regarding Reduction Rule 1 and therefore each connected component has a non-empty intersection with A. Moreover, we assume that $k \geq 1$ since otherwise we can solve I in polynomial time by computing the number of vulnerable connections of G.

We first assume that there exists a connected component Z_i of size at least $y + 1$. Since we assume that every connected component of G contains some vertices from A, let $v \in Z_i \cap A$. Since $|Z_i| \geq y + 1$, we can remove at least y vulnerable connections by deleting v. Together with the fact that $k \geq 1$ we conclude that I is a yes-instance. Throughout the rest of the proof, we assume that $|Z_i| \leq y$ for every connected component of G.

In the remainder of the proof, we show that a component table T of I can be computed in $2^{\mathcal{O}(\sqrt{y}\log y)} \cdot n^{\mathcal{O}(1)}$ time. Recall that a component table T of I has entries of type $T[i, k']$ with $i \in [1, t]$ and $k' \in [0, k]$ such that $T[i, k']$ is the minimum number of vulnerable connections in $G[Z_i]$ that remain after deleting exactly k' vertices in Z_i. With a component table at hand, we can then solve CNP-V in polynomial time due to Lemma 4.

Let Z_i be a connected component. We now describe how to compute all component information $T[i, k']$ with $k' \in [0, k]$ in $2^{\mathcal{O}(\sqrt{y}\log y)} \cdot n^{\mathcal{O}(1)}$ time.

We first consider the case where $k < \sqrt{y}$. For each $k' \in [0, k]$, we compute $T[i, k']$ by iterating over all subsets $S \subseteq Z_i$ and computing the vulnerability of $G[Z_i] - S$. Note that for each $k' \in [0, k]$, there are at most $\binom{|Z_i|}{k'} \leq |Z_i|^{k'}$ subsets $S \subseteq Z_i$ of size k'. Since $|Z_i| \leq y$ and $k' \leq k < \sqrt{y}$, we can compute each component information $T[i, k']$ in $y^{\sqrt{y}} \cdot n^{\mathcal{O}(1)} = 2^{\mathcal{O}(\sqrt{y}\log y)} \cdot n^{\mathcal{O}(1)}$ time. Since $k \leq n$ in non-trivial instances, we can compute all component information $T[i, k']$ for $k' \in [0, k]$ in $2^{\mathcal{O}(\sqrt{y}\log y)} \cdot n^{\mathcal{O}(1)}$ time.

Next, let $k \geq \sqrt{y}$. For this, we first identify a further case, where I is a yes-instance.

Claim 2 *If* $k \geq \sqrt{y}$ *and there exists a connected component* Z_i *such that* $|Z_i| \geq \frac{3\sqrt{y}+1}{2}$ *and* $|Z_i \cap A| \geq \sqrt{y}$, *then* I *is a yes-instance.*

Proof. Since $|Z_i \cap A| \geq \sqrt{y}$ and $k \geq \sqrt{y}$, we may delete $\sqrt{y}$ vulnerable vertices from Z_i. This decreases the number of vulnerable connections by at least

$$\underbrace{\binom{\sqrt{y}}{2}}_{=:c_1} + \underbrace{\sqrt{y} \cdot (|Z_i| - \sqrt{y})}_{=:c_2},$$

where c_1 corresponds to the vulnerable connections between the deleted vertices and c_2 corresponds to vulnerable connections between the deleted vertices and the remaining vertices in Z_i. Then, since $|Z_i| \geq \frac{3\sqrt{y}+1}{2}$, the number of vulnerable connections is decreased by at least $\binom{\sqrt{y}}{2} + \sqrt{y} \cdot \left(\frac{3\sqrt{y}+1}{2} - \sqrt{y}\right) = \frac{y-\sqrt{y}}{2} + \frac{3y+\sqrt{y}}{2} - y = y$. Therefore, I is a yes-instance. $\diamond$

Since we assumed $k \geq \sqrt{y}$, we may immediately return *yes* if Z_i satisfies the two constraints stated in the claim. For the rest of the proof, we may assume that this is not the case. Consequently, we have $|Z_i| < \frac{3\sqrt{y}+1}{2}$ or $|Z_i \cap A| < \sqrt{y}$. Consider the following cases.

Case 1: $|Z_i| < \frac{3\sqrt{y}+1}{2}$. We can then compute the component information of the connected component Z_i by iterating over all subsets $S \subseteq Z_i$ and computing the number of vulnerable connections in $G[Z_i] - S$. Since $|Z_i| < \frac{3\sqrt{y}+1}{2}$, there are at most $2^{\frac{1}{2}\cdot(3\sqrt{y}+1)} \in 2^{\mathcal{O}(\sqrt{y})}$ subsets. Therefore, all component information $T[i, k']$ can be computed in $2^{\mathcal{O}(\sqrt{y}\log y)} \cdot n^{\mathcal{O}(1)}$ time.

Case 2: $|Z_i \cap A| < \sqrt{y}$. Then, since $k \geq \sqrt{y}$, we have $T[i, k'] = 0$ for all $k' \geq |Z_i \cap A|$ since one may remove all vulnerable vertices in Z_i and afterwards, no vertex of Z_i is part of a vulnerable connection anymore. It remains to compute component information $T[i, k']$ with $k' < \sqrt{y}$. This is done by iterating over every set $S \subseteq Z_i$ of size k' and computing the vulnerability of $G[Z_i]-S$. Since there are at most $|Z_i|^{k'}$ such subsets and $|Z_i| \leq y$, one component information $T[i, k']$ with $k' < \sqrt{y}$ can be computed in $y^{\mathcal{O}(\sqrt{y})} \cdot n^{\mathcal{O}(1)} = 2^{\mathcal{O}(\sqrt{y}\log y)} \cdot n^{\mathcal{O}(1)}$ time. Since $k \leq n$ in non-trivial instances, we can compute all component information $T[i, k']$ for $k' \in [0, k]$ in $2^{\mathcal{O}(\sqrt{y}\log y)} \cdot n^{\mathcal{O}(1)}$ time.

By the above, we can compute the component table T of I in $2^{\mathcal{O}(\sqrt{y}\log y)} \cdot n^{\mathcal{O}(1)}$. Together with Lemma 4, we conclude that CNP-V can be solved within the claimed running time. □

4.2 Hardness for Non-Deletable Vulnerable Vertices

Now, we show that—in contrast to CNP-V—the CNP-NDV problem is W[1]-hard with respect to the parameter $k + y$. In the algorithm for CNP-V behind Theorem 6, we exploit that we may bound the size of a single connected component in the parameter y if one single vulnerable vertex lies in that component. This relies on the fact that the deletion of this vertex provides a critical node cut. Note that this is not possible in the case of CNP-NDV. We show that CNP-NDV is W[1]-hard for $k + y$ even if the input graph consists of a single connected component with exactly one vulnerable vertex. We reduce from Clique which has as input graph G and an integer ℓ, and asks whether G contains a set of ℓ vertices that are pairwise adjacent. It is well-known that Clique is W[1]-hard with respect to ℓ [5, 7].

The reduction follows the spirit of a reduction of Fomin et al. [9] that shows W[1]-hardness of the Cutting at most k Vertices with Terminal problem. The reduction of Fomin et al. [9] already shows W[1]-hardness of CNP-NDV with respect to the parameter k, even if $|A| = 1$. We adapt the reduction to show hardness with respect to the larger parameter $k + y$.

Theorem 7 CNP-NDV *is* W[1]*-hard with respect to the parameter $k + y$, even if $|A| = 1$ and the input graph has diameter 2.*

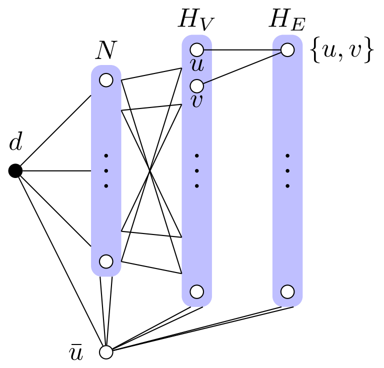

Figure 4: Construction of G' in the proof of Theorem 7.

Proof: *Construction:* Let (G, ℓ) be an instance of Clique. We assume without loss of generality that $\ell \leq n_G$. We construct an instance (G', A, k, x) of CNP-NDV as follows: We start with a single vulnerable vertex d and set $A := \{d\}$. Then, we add $\ell + 1$ neighbor-vertices to d and call this set N. We add a set H_V of n_G vertices to G' that corresponds to the vertex set $V(G)$. We add an edge between every vertex of H_V and every vertex of N. Furthermore, we add for every edge in G a vertex to G' and name that vertex set H_E. For every edge $\{u, v\}$ of G, connect the corresponding vertex of H_E to the two vertices of H_V that correspond to u and v. Finally, we add a universal vertex $\bar{u}$ that is connected to every vertex of G'. The construction of G' is shown in Figure 4.

We set $k := \ell + 1$ and $x := |N| + n_G + m_G + 1 - k - \binom{\ell}{2}$. Then, $y = \binom{\ell}{2} + \ell + 1$ as we are dealing with exactly one vulnerable vertex. Thus, k and y are polynomially bounded in ℓ. Moreover, $|A| = |\{d\}| = 1$ and the diameter of G' is 2 because of the universal vertex $\bar{u}$. The construction can be computed in polynomial time.

Intuition: We constructed G' so that there exists a vertex in G' for every vertex and every edge in G. The vertices of H_V correspond to the n_G vertices of G and the vertices of H_E to the m_G edges of G. As we can add at most k vertices to a critical node cut, we cannot add all vertices of $N \cup \{\bar{u}\}$ to a critical node cut. Consequently, the only vertices that can be separated from d without being in the critical node cut are in the vertex set H_E.

Correctness: Let (G, ℓ) be a yes-instance of Clique. Thus, there is a set of vertices $X \subseteq V(G)$ such that $G[X]$ is a clique of size ℓ. Thus, $G[X]$ contains $\binom{\ell}{2}$ edges. Let Y be the set of these edges. We define the vertex set C that contains the ℓ vertices in H_V that correspond to the vertices of X and the universal vertex $\bar{u}$ of G'. The size of the set C is $\ell + 1 = k$. Then, the $\binom{\ell}{2}$ vertices in H_E that correspond to Y are isolated and thus no longer connected to d. It follows that C is a critical node cut, because $|Y| + |C| = y$ vulnerable connections are deleted from G' by cutting C. Thus, (G', A, k, x) is a yes-instance.

Conversely, let (G', A, k, x) be a yes-instance of CNP-NDV. Let C be a corresponding critical node cut. It follows that $|C| \leq k = \ell + 1$ and cutting C from G' deletes y vulnerable connections. Let $Q \subseteq V(G') \setminus C$ be the set of vertices that are separated from d in $G' - C$. We observe that $Q \subseteq H_E$, because the vertices of $N \cup \{\bar{u}\}$ are adjacent to d and every vertex of H_V is connected to d via more than k vertex-disjoint paths. Furthermore, a vertex of H_E corresponding to $\{u, v\}$ is in Q, if and only if $\{u, v, \bar{u}\} \subseteq C$. Since $|C| + |Q| \geq y = \ell + 1 + \binom{\ell}{2}$ and $|C| \leq \ell + 1$, we have $|Q| \geq \binom{\ell}{2}$. Thus, the critical node cut C consists of $\bar{u}$ and ℓ vertices of H_V that are pairwise adjacent in G. Hence, G has a clique of size ℓ. □

The constructed graph G' does not contain edges within N, within H_V, or within H_E. Thus, removing $\bar{u}$ makes G' bipartite with vertex bipartition $(\{d\} \cup H_V, N \cup H_E)$. Since $\bar{u}$ is in every critical node cut of the constructed instance (as shown in the backward direction of the proof), removing $\bar{u}$ and decreasing k by one gives an equivalent instance.

Corollary 5 CNP-NDV *is* W[1]*-hard with respect to* $k+y$*, even on bipartite graphs and if* $|A| = 1$.

We can also alter the construction of G' in Theorem 7 by making the vertices of $N \cup H_V \cup \{d, \bar{u}\}$ a clique. Then, the size of the neighborhood of A increases by $|H_V|$ and thus is not bounded in ℓ. Then, G' is a split graph (these are the graphs whose vertex sets can be partitioned into a clique and an independent set); the correctness proof is analogous.

Corollary 6 CNP-NDV *is* W[1]*-hard with respect to* $k + y$ *for* $|A| = 1$*, even on split graphs of diameter 2.*

5 Parameters Related to the Vertex Cover Number

5.1 An FPT Algorithm for the Vertex Cover Number

First, we obtain an FPT-algorithm for both problems parameterized by the vertex cover number vc. To unify the presentation, we present the algorithm for the more general CNP-VNDV problem, it uses a combination of branching and dynamic programming. A subroutine in the algorithm will be to compute optimal critical node cuts under the premise that the remaining graph is connected. The following lemma shows that one can efficiently compute critical node cuts that are at least as good as such connectivity-preserving critical node cuts.

Lemma 7 *Given an instance* (G, A, N, k, x) *of* CNP-VNDV*, one can compute in* $\mathcal{O}(m+n)$ *time a set* $D \subseteq (V(G) \setminus N)$ *of size at most* k *such that for every set* $C \subseteq (V(G) \setminus N)$ *of size at most* k *where* $G - C$ *is connected, the* A*-vulnerability of* $G - D$ *is at most the* A*-vulnerability of* $G - C$.

Proof: If $V(G) \setminus N$ contains at most k vertices, then we may simply delete all vertices in $V(G) \setminus N$. Otherwise, consider a set C such that $G' = G - C$ is connected and has minimal A-vulnerability among all such graphs. The graph G' then contains $n' = n - |C|$ vertices and the A-vulnerability of G' is

$$\mathrm{vul}(G') = \binom{n'_A}{2} + n'_A \cdot (n' - n'_A) = n'_A \cdot ((n'_A - 1)/2 + n' - n'_A) = n'_A(-n'_A/2 - 1/2 + n')$$

where n'_A is the number of vertices of A that are contained in G'. Note that this value is minimal when n'_A is minimal: the mapping $n'_A \mapsto n'_A(-n'_A/2 - 1/2 + n')$ is a quadratic function reaching its maximum at $n'_A = n' - 1/2$. Consequently, the function is non-decreasing for increasing integer values $n'_A \in [0, n']$. Altogether, this shows that minimizing n'_A minimizes the A-vulnerability of G'.

Now consider the set D obtained by deleting vertices of A until all remaining vertices of A are also contained in N and thus non-deletable. Formally, this means adding $\min(|A \setminus N|, k)$ vertices of $A \setminus N$ to D and $\max(k - |A \setminus N|, 0)$ vertices of $V(G) \setminus (N \cup A)$ to D. The graph $G - D$ has n' vertices and its A-vulnerability is thus at most

$$\binom{n^*_A}{2} + n^*_A \cdot (n' - n^*_A)$$

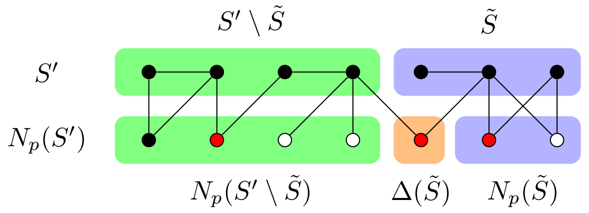

Figure 5: Illustration of the recurrence for a table entry $T[S', k']$. The vertices of N are black and all vertices are vulnerable. An optimal solution for $G' = G[N_p[S']]$ and $k' = 3$ is to delete the red vertices. In the example, $\delta(\tilde{S}) = 1$ because 1) the set $\Delta(\tilde{S})$ contains one vertex and this vertex is not contained in N and 2) there are no edges between $\tilde{S}$ and $S' \setminus \tilde{S}$.

where n_A^* denotes the number of vertices of A contained in $G - D$. By construction of D, $n_A^* \leq n'_A$ and, by the discussion above, the A-vulnerability of $G - D$ thus is at most the A-vulnerability of $G - C$. □

The general approach of the algorithm is to consider all possibilities to delete vertices in the vertex cover. For each such possibility, we then solve the problem of finding an optimal critical node cut when no vertices of the remaining vertex cover may be deleted. In the following, we describe an algorithm for this subproblem and bound its running time.

Lemma 8 *Let $I = (G, A, N, k, x)$ be an instance of* CNP-VNDV *and let S be a given vertex cover of G such that $S \subseteq N$. Then, one can solve I in $3^{|S|} \cdot n^{\mathcal{O}(1)}$ time.*

Proof: We use dynamic programming over subsets of S. The dynamic programming table T contains entries of the type $T[S', k']$ where S' is a subset of S and $k' \in [0, k]$. To define the meaning of a table entry, let $N_p(S')$, for $S' \subseteq S$ denote the private neighbors of S' in the independent set $V(G) \setminus S$, that is, the vertices which have at least one neighbor in S', no neighbor in $S \setminus S'$ and which are not in S. Formally, $N_p(S') = N(S') \setminus (N(S \setminus S') \cup S)$. Moreover, let $N_p[S'] = S' \cup N_p(S')$ denote the union of the private neighborhood of S' and the set S' itself. Finally, let $G' := G[N_p[S']]$ denote the subgraph induced by S' and its private neighborhood. A table entry $T[S', k']$ contains the minimum A-vulnerability of any graph that is obtained from G' by deleting at most k' vertices that are not from N. Since $S' \subseteq N$, the deleted vertices have to belong to $N_p(S') \setminus N$. With this definition, the value of $T[S, k]$ then is the minimum A-vulnerability of any graph that can be obtained from G by deleting at most k vertices which are not in N. Clearly, I is a yes-instance if and only if this number is at most x.

Informally, the recurrence to compute the value of $T[S', k']$ is obtained by the following considerations (see also Figure 5 for an illustration). Recall that $G' = G[N_p[S']]$ is the graph for which an optimal critical node cut C' with at most k' vertices needs to be computed at this table entry. If all vertices of S' end up in the same connected component of $G' - C'$, then $G' - C'$ is connected and we may compute an optimal critical node cut using Lemma 7. Otherwise, there is a nonempty subset $\tilde{S}$ of S' that ends up in a different component of $G' - C'$ than the other vertices of S'. In this case, all vertices that have at least one neighbor in $\tilde{S}$ and one in $S' \setminus \tilde{S}$ have to be deleted to separate $\tilde{S}$ from the other vertices in S'. Formally, this set is defined as

$$\Delta_{S'}(\tilde{S}) := N(\tilde{S}) \cap N(S' \setminus \tilde{S}) \cap N_p(S').$$

To avoid cluttered notation, we write $\Delta(\tilde{S})$ instead of $\Delta_{S'}(\tilde{S})$ since S' will be clear from context. The deletion of $\Delta(\tilde{S})$ splits the instance into two independent parts, one with the vertices from $N_p[\tilde{S}]$, the other part containing the vertices from $N_p[S' \setminus \tilde{S}]$. The two parts can then be solved independently if we consider the optimal way to distribute the deletion budget between the two parts.

To simplify the description, we define $T[S', k'] := +\infty$ for all $k' < 0$. Moreover, we set $T[\emptyset, k'] := 0$ for all $k' \in [0, k]$. This is correct since the empty graph has A-vulnerability 0. Furthermore, we precompute an auxiliary table Q, where an entry $Q[S', k']$ is a lower bound of the A-vulnerability that we get when $G(N_p[S'])$ remains connected after the deletion of k' vertices. More precisely, let $Q[S', k']$ be the A-vulnerability of $G' - D'$ where D' is computed by applying the algorithm from Lemma 7 to G'. Then, any connected graph obtained from G' by deleting at most k' vertices of $N_p(S') \setminus N$ has vulnerability at least $Q[S', k']$.

For nonempty S', we now compute $T[S', k']$ by the recurrence

$$T[S', k'] = \min_{\tilde{S} \subseteq S'} \min_{\tilde{k} \leq k'} \Big(Q[\tilde{S}, \tilde{k}] + T[S' \setminus \tilde{S}, k' - \tilde{k} - \delta(\tilde{S})] \Big)$$

where $\delta(\tilde{S}) = |\Delta(\tilde{S})|$ if

- $\Delta(\tilde{S})$ contains no vertices of N and
- there are no edges with one endpoint in $\tilde{S}$ and one endpoint in $S' \setminus \tilde{S}$,

and $\delta(\tilde{S}) = k + 1$, otherwise. That is, δ counts the number of vertex deletions that are necessary to disconnect $\tilde{S}$ and $S' \setminus \tilde{S}$ in G' if it is at all possible to disconnect the two sets without deleting vertices in N. Otherwise, the value of δ is sufficiently large to ensure that the equation evaluates to $+\infty$.

To show the correctness of the recurrence, we prove the following two inequalities between the right-hand side and left-hand side of the equation.

($\geq$) Consider $T[S', k']$ and let $C' \subseteq V(G') \setminus N$ be a set of size at most k' such that $G' - C'$ has A-vulnerability $T[S', k']$. Let $\tilde{Z}$ be some connected component of $G' - C'$ and let $\tilde{S} := \tilde{Z} \cap S$ be the set of the vertex cover nodes of $\tilde{Z}$. Since $\tilde{Z}$ is a connected component in $G' - C'$, there are no edges between $\tilde{S}$ and $S' \setminus \tilde{S}$ and C' contains every vertex of G' with one neighbor in $\tilde{S}$ and one in $S' \setminus \tilde{S}$, that is, every vertex of $\Delta(\tilde{S})$. In $G' - \Delta(\tilde{S})$, the set $N_p[\tilde{S}]$ is one connected component. Let $\tilde{k}$ denote the number of vertices from $N_p[\tilde{S}]$ that are deleted. Then, $G'[N_p[\tilde{S}]] - C'$ has A-vulnerability at least $Q[\tilde{S}, \tilde{k}]$ since this graph is exactly $G'[\tilde{Z}]$ and therefore connected. Moreover, $G'[N_p[S' \setminus \tilde{S}]] - C'$ has A-vulnerability at least $T[S' \setminus \tilde{S}, k' - \tilde{k} - \delta(\tilde{S})]$, since C' contains at most $k' - \tilde{k} - \delta(\tilde{S})$ vertices of $N_p[S' \setminus \tilde{S}]$ and by the definition of T. Altogether, we have $T[S', k'] \geq Q[\tilde{S}, \tilde{k}] + T[S' \setminus \tilde{S}, k' - \tilde{k} - \delta(\tilde{S})]$ for this particular choice of $\tilde{S}$ and $\tilde{k}$. Therefore, $T[S', k']$ is also at least as large as the minimum over all choices for $\tilde{S}$ and $\tilde{k}$.

($\leq$) Consider a set $\tilde{S}$ and a number $\tilde{k}$ minimizing $Q[\tilde{S}, \tilde{k}] + T[S' \setminus \tilde{S}, k' - \tilde{k} - \delta(\tilde{S})]$ in the right-hand side of the recurrence and observe that the value of the right-hand side is finite. Then, there exist a set $\tilde{C}$ of at most $\tilde{k}$ vertices and a set $\hat{C}$ of at most $k - \tilde{k} - \delta(\tilde{S})$ vertices such that $G'[N_p[\tilde{S}]] - \tilde{C}$ has A-vulnerability at most $Q[\tilde{S}, \tilde{k}]$ and $G'[N_p[S' \setminus \tilde{S}]] - \hat{C}$ has A-vulnerability $T[S' \setminus \tilde{S}, k' - \tilde{k} - \delta(\tilde{S})]$. Now, consider the set $C' = \tilde{C} \cup \Delta(\tilde{S}) \cup \hat{C}$. This set has size at most $\tilde{k} + \delta(\tilde{S}) + (k' - \tilde{k} - \delta(\tilde{S})) = k'$. Then, the A-vulnerability of $G' - C'$ is at most $Q[\tilde{S}, \tilde{k}] + T[S' \setminus \tilde{S}, k' - \tilde{k} - \delta(\tilde{S})]$, since $G' - C'$ is the disjoint union of $G'[N_p[\tilde{S}]] - \tilde{C}$ and $G'[N_p[S' \setminus \tilde{S}]] - \hat{C}$ as 1) there are no edges between $\tilde{S}$ and $S \setminus \tilde{S}$ and 2) all vertices of G' that are not in $N_p[\tilde{S}]$ or in $N_p[S' \setminus \tilde{S}]$ are contained in $\Delta(\tilde{S})$.

The running time for filling the tables can be bounded as follows. Table Q has $\mathcal{O}(2^{|S|} \cdot k)$ entries, each of which can be computed in $n^{\mathcal{O}(1)}$ time by Lemma 7. Table T has $\mathcal{O}(2^{|S|} \cdot k)$ entries as well and the time needed for filling T is dominated by the total number of different terms over which the minimum is taken. This number is $3^{|S|} \cdot n^{\mathcal{O}(1)}$ since each such term corresponds to a 3-partition of S into $S \setminus S'$, $S' \setminus \tilde{S}$, and $\tilde{S}$. □

We can now use this algorithm to obtain a single-exponential FPT-algorithm for the vertex cover number of the input graph.

Theorem 8 CNP-VNDV *can be solved in* $4^{\text{vc}} \cdot n^{\mathcal{O}(1)}$ *time.*

Proof: Let (G, A, N, k, x) be an instance of CNP-VNDV. The first step of the algorithm is to compute a minimum vertex cover S of G. This can be done in $\mathcal{O}(2^{\text{vc}}(n + m))$ time using the standard search tree algorithm [5]. Then, for each set $D \subseteq (S \setminus N)$ of size at most k, we create a branch where we assume that D is the set of vertex deletions in the vertex cover S. Consider one such branch. Let $G' := G - D$ and let $k' := k - |D|$. Observe that $S' := S \setminus D$ is a vertex cover of G'. The question is now whether there is a vertex set C' of size at most k' that avoids $N \cup S'$ such that $G' - C'$ has A-vulnerability at most x. This is equivalent to solving the instance $(G', A, N \cup S', k', x)$ and can be done in $3^{|S'|} \cdot n^{\mathcal{O}(1)}$ time by Lemma 8. If $|D| = i$, then the running time for solving the instance is therefore $3^{\text{vc} - i} \cdot n^{\mathcal{O}(1)}$. Consequently, the overall running time for the branching is $\sum_{i=0}^{\text{vc}} \binom{\text{vc}}{i} \cdot 3^{\text{vc} - i} \cdot n^{\mathcal{O}(1)}$. Using the binomial theorem, the overall running time for all possibilities of D is thus $4^{\text{vc}} \cdot n^{\mathcal{O}(1)}$ time. □

5.2 On Problem Kernelization

Next, we show polynomial-size kernels for CNP-V parameterized by $\text{vc} + x$ and for CNP-NDV parameterized by $\text{vc} + k + x$. To this end, we first make a simple observation on k and the vertex cover number of the input graph. Let (G, A, k, x) be an instance of CNP-V or CNP-NDV. For the rest of the section, we fix a vertex set S which is a 2-approximation of a minimum vertex cover of G, that is, $|S| \leq 2 \cdot \text{vc}$. Note that S can be computed in linear time.

Consider CNP-V. Removing S from G results in an edgeless graph and therefore, there are no vulnerable connections in $G - S$. Thus, we may immediately return yes if k is at least as big as the size of S.

Reduction Rule 5 *Let* (G, A, k, x) *be an instance of* CNP-V*. Return yes, if* $k \geq |S|$*.*

After application of this rule, we may assume $k < 2 \cdot vc$. Recall that we assume that the input instance of CNP-V is reduced with respect to Reduction Rules 1 and 4 and therefore we may assume that there are no isolated vertices and that $|N(v) \cap A| \leq k + \sqrt{2x}$ for every vertex v. In the following, we show that we can use these assumptions to bound the size of A in $\text{vc} + x$.

Lemma 9 *After Reduction Rules 1, 4, and 5 have been applied exhaustively, in an instance* (G, A, k, x) *of* CNP-V*, the set* A *contains less than* $2\,\text{vc} \cdot (2\,\text{vc} + \sqrt{2x} + 1)$ *vertices.*

Proof: Recall that S is a vertex cover of G with $|S| \leq 2 \cdot \text{vc}$. After Reduction Rule 1 has been applied exhaustively, every vertex of G is either in the vertex cover, or a neighbor of S. Then, with Reduction Rules 4 and 5 it follows that

$$|A| = |S \cap A| + |N(S) \cap A| \overset{\text{Red. Rule 4}}{\leq} 2\,\text{vc} + 2\,\text{vc} \cdot (k + \sqrt{2x}) \overset{\text{Red. Rule 5}}{<} 2\,\text{vc} \cdot (2\,\text{vc} + \sqrt{2x} + 1).$$

□

Next, we define a subset B of the vertices. We provide two different definitions for CNP-V or CNP-NDV: For CNP-V, we define $B := A \cup S$. For CNP-NDV, we define $B := S$. We call B the *base*. We then have $|B| \le 2 \cdot \mathrm{vc}$ when we deal with an instance of CNP-NDV and by Lemma 9 we have $|B| \le 2\,\mathrm{vc} \cdot (2\,\mathrm{vc} + \sqrt{2x} + 2)$ when we deal with an instance of CNP-V. It remains to bound the size of the set $Y := V \setminus B$. Note that Y is an independent set because B contains a vertex cover. Moreover, Y does not contain isolated vertices since the instance is reduced with respect to Reduction Rule 1. In the following, we provide a reduction rule that in instances of CNP-NDV helps us to handle vulnerable vertices in the set Y. After the reduction rule has been applied exhaustively, if a vertex v has a neighborhood of size at least $k + x + 1$, all neighbors of v are non-vulnerable.

Reduction Rule 6 *Let (G, A, k, x) be an instance of* CNP-NDV *with base B. If a vertex $v \in B$ has more than $k + x$ neighbors of which one is vulnerable, then do the following:*

1. *If $v \notin A$, then remove v from the graph and decrease k by one.*
2. *If $v \in A$, then return* no.

Lemma 10 *Reduction Rule 6 is safe and can be applied exhaustively in $\mathcal{O}(n^2)$ time.*

Proof: *Safeness: Case 1: $v \notin A$.* We show that, if there is a critical node cut C for the instance (G, A, k, x), then $v \in C$. Let d be a vulnerable neighbor of v. Assume towards a contradiction that $v \notin C$. Consequently, $\{d, w\}$ is a vulnerable connection in $G - C$ for every $w \in N(v) \setminus C$ with $d \neq w$. Also, $\{d, v\}$ is a vulnerable connection in $G - C$. By the requirements of the reduction rule, $|N(v)| > k + x$. It follows that $|(N(v) \setminus \{d\}) \setminus C| \ge x$. Together with $\{d, v\}$ in $G - C$, there are more than x vulnerable connections. We conclude that $v \in C$.

Case 2: $v \in A$. It directly follows from Case 1 that, if $v \in A$, there exists no critical node cut for (G, A, k, x) that is disjoint from A. Hence, there is no critical node cut for (G, A, k, x) for the instance of CNP-NDV. Thus, we can return no.

Running time: Since each application of Reduction Rule 6 removes a vertex, it can be applied at most n times. For every vertex, we compute the size of the neighborhood and check whether one vertex is vulnerable in $\mathcal{O}(n)$ time. Thus, this reduction rule can be applied exhaustively in $\mathcal{O}(n^2)$ time. □

This reduction rule can only be applied on instances of CNP-NDV, because, if $v \notin A$, we know that we have to add v to a critical node cut. However, in CNP-V there remain three options: we can add the vulnerable vertex d, or the vertex v, or both to a critical node cut. Thus, in order to avoid such a decision for instances of CNP-V, we added all vulnerable vertices to the base B.

In the last reduction rule, we use the Expansion Lemma. The Expansion Lemma was introduced by Prieto-Rodríguez [14]. We use the formulation by Cygan et al. [5]. Let H be a bipartite graph with vertex bipartition (R, T). For a positive integer q, a set of edges $M \subseteq E(H)$ is called a *q-expansion of R into T*, if every vertex of R is incident with exactly q edges of M and the edges in M are incident with exactly $q \cdot |R|$ vertices in T.

Lemma 11 (Expansion Lemma [5]) *Let $q \ge 1$ be a positive integer and H be a bipartite graph with vertex bipartition (R, T) such that $|T| \ge q \cdot |R|$ and there are no isolated vertices in T. Then, there exist nonempty vertex sets $P \subseteq R$ and $Q \subseteq T$ such that there is a q-expansion of P into Q and $N_H(Q) \subseteq P$. Furthermore, the sets P and Q can be found in time polynomial in the size of H.*

Since the Expansion Lemma can only be applied to bipartite graphs, in the next reduction rule we define a bipartite graph that is an induced subgraph of G. We apply the Expansion Lemma on the graph G' which contains the vertices $V' := V(G)$ and the set of edges $E' := E(G) \setminus E(G[B])$. This is a bipartite graph, because we do not consider the edges within B and, by definition, Y is an independent set. Thus, G' is a bipartite graph with vertex bipartition (B, Y).

Now, we assume that Reduction Rules 1 and 6 are exhaustively applied.

Reduction Rule 7 *If the set Y contains at least $(k + x + 2) \cdot |B|$ vertices, then, in the graph G' compute non-empty vertex sets $P \subseteq B$ and $Q \subseteq Y$ such that there is a $k + x + 2$-expansion of P into Q. Remove an arbitrary vertex $v \in Q$ from G.*

Lemma 12 *For an instance of* CNP-V *or* CNP-NDV*, Reduction Rule 7 is safe and can be applied exhaustively in polynomial time.*

Proof: *Safeness:* Let (G, A, k, x) be an instance of CNP-V or CNP-NDV with base B for which the inequality $|Y| \geq (k + x + 2) \cdot |B|$ holds.

We start by showing that we can apply the Expansion Lemma. After Reduction Rule 1 has been applied exhaustively, all vertices in Y are adjacent to at least one vertex in B. Thus, all conditions for the Expansion Lemma are fulfilled. From the Expansion Lemma, we know that we can then find non-empty vertex sets $P \subseteq B$ and $Q \subseteq Y$ such that there is a $k + x + 2$-expansion of P into Q in polynomial time. Also, the sets fulfill $N_G(Q) \subseteq P$.

For the rest of the proof, let v be an arbitrary but fixed vertex of Q. We show that (G, A, k, x) is a yes-instance of CNP-V or CNP-NDV, if and only if $(G - \{v\}, A, k, x)$ is a yes-instance of the same problem. Observe that v is non-vulnerable: In an instance of CNP-V we defined $A \subseteq B$ and thus $A \cap Y = \emptyset$ and in particular $A \cap Q = \emptyset$. In an instance of CNP-NDV, after Reduction Rule 6 has been applied exhaustively, a vertex of B with a neighbor in $A \cap Y$ has at most $k + x$ neighbors. Thus, a described $k + x + 2$-expansion of P into Q cannot exist if $A \cap Q \neq \emptyset$.

Because $G - \{v\}$ is an induced subgraph of G, $C \setminus \{v\}$ is a critical node cut for $(G - \{v\}, A, k, x)$ if C is a critical node cut for (G, A, k, x).

Conversely, let $(G - \{v\}, A, k, x)$ be a yes-instance of CNP-V or CNP-NDV and let C be a corresponding critical node cut. From the Expansion Lemma, we know $N(Q) \subseteq P$. In $(G - \{v\}) - C$ there is no vulnerable connection $\{d, u\}$ with $d \in A$ and $u \in P$: Otherwise, for all $w \in (N_G(u) \cap Q) \setminus (\{v\} \cup C)$ also $\{d, w\}$ is a vulnerable connection in $(G - \{v\}) - C$. By the definition of P and Q, the size of $(N_G(u) \cap Q)$ is at least $k + x + 1$ and thus $\{u\} \cup ((N_G(u) \cap Q) \setminus (\{v\} \cup C))$ contains more than x vertices. This is a contradiction to C being a critical node cut. By the same argument, the sets A and $P \setminus C$ are not connected in $(G - \{v\}) - C$. It follows that in $(G - \{v\}) - C$ the sets $P \setminus C$ and $Q \setminus C$ are in connected components that do not contain a vulnerable vertex. Since $N_G(v) \subseteq P$, the A-vulnerability of $(G - \{v\}) - C$ is the A-vulnerability of $G - C$ and C is also a critical node cut for (G, A, k, x).

Clearly, the rule can be performed in polynomial time. □

It remains to give a bound on the size of the computed kernel. To this end, consider the following lemma.

Lemma 13 *An instance (G, A, k, x) of* CNP-V *or* CNP-NDV *contains less than $|B| \cdot (k + x + 3)$ vertices after Reduction Rule 7 has been applied exhaustively.*

Proof: Let (G, A, k, x) be an instance that is reduced exhaustively by Reduction Rule 7. Then, the set Y contains less than $|B| \cdot (k + x + 2)$ vertices. As $V(G) = B \cup Y$, the graph G contains less than $|B| + |B| \cdot (k + x + 2) = |B| \cdot (k + x + 3)$ vertices. □

We next argue why the bound from the previous lemma implies that CNP-V admits a polynomial kernel for parameterization by $\mathrm{vc} + x$ and that CNP-NDV admits a polynomial kernel for parameterization by $\mathrm{vc} + x + k$. Recall that S is a 2-approximation of a vertex cover.

First, consider CNP-V. Recall that $B = A \cup S$ in this case. Hence, $B \leq 2\,\mathrm{vc} \cdot (2\,\mathrm{vc} + \sqrt{2x} + 2)$. This implies that CNP-V admits a problem kernel with less than $(2\,\mathrm{vc}\,(2\,\mathrm{vc} + \sqrt{2x}+2)) \cdot (k+x+3)$ vertices. Furthermore, since the instance is reduced with respect to Reduction Rule 5, we conclude that CNP-V admits a polynomial kernel for parameterization by $\mathrm{vc} + x$.

Second, consider CNP-NDV. Recall that $B = S$ in this case. Hence, $|B| \leq 2\,\mathrm{vc}$ in this case. This implies a problem kernel with less than $2 \cdot \mathrm{vc} \cdot (k+x+3)$ vertices for CNP-NDV. In summary, we obtain the following

Theorem 9 *a)* CNP-V *admits a problem kernel with less than* $2\,\mathrm{vc} \cdot (2\,\mathrm{vc} + \sqrt{2x} + 2) \cdot (2\,\mathrm{vc} + x + 3)$ *vertices.*

b) CNP-NDV *admits a problem kernel with less than* $2 \cdot \mathrm{vc} \cdot (k + x + 3)$ *vertices.*

6 Conclusion

We introduced two new critical node detection problems Critical Node Problem with Vulnerable Nodes (CNP-V) and Critical Node Problem with Non-Deletable Vulnerable Nodes (CNP-NDV), that take into account that we may be only interested in the number of connected pairs for a specified set of vulnerable vertices. We performed a parameterized complexity analysis for some of the most natural parameters and their combinations.

We left open, however, the complexity of several natural parameterizations. For example, is CNP-V FPT with respect to $|A|$? At the moment we only have an XP-algorithm for A and an FPT-algorithm for $|A|+x$. Moreover, does either problem admit a polynomial kernel for the vertex cover number vc?

A further interesting structural parameter is the feedback vertex number (fv) of the input graph. That is the minimum number of vertices that need to be removed to obtain a graph without cycles. Since fv never exceeds the vertex cover size, it is interesting to ask whether our results on parameterization by vc be transferred to this smaller parameter. Moreover, the feedback vertex number also measures the distance to a polynomial-time solvable case, since CNP-V and CNP-NDV can be solved in polynomial time on acyclic graphs [15]. Thus, parameterization by fv is a distance-from-triviality parameterization.

It is also open whether CNP-NDV is FPT or W[1]-hard for parameterization by treewidth (tw). Recall that CNP is W[1]-hard for tw [2, 12] and thus, the more general CNP-V is W[1]-hard for tw as well. However, this does not imply W[1]-hardness for CNP-NDV. Is it possible to adapt the reductions for CNP-NDV [2, 12] so that they work for non-deletable vulnerable vertices or is a fundamentally new approach required?

Besides considering CNP-V and CNP-NDV, it is interesting to introduce vulnerable vertices for other natural critical node problems [13]. For example, instead of defining the A-vulnerability via the number of vulnerable connections, one may consider centrality measures for vertex sets [3, 10] to assess how exposed the vulnerable vertices are. Moreover, one may consider edge deletions instead of vertex deletions. For example, is it possible to adapt existing FPT results for related edge deletion problems [8] to problem versions with vulnerable vertices?

References

[1] B. Addis, M. Di Summa, and A. Grosso. Identifying critical nodes in undirected graphs: Complexity results and polynomial algorithms for the case of bounded treewidth. *Discrete Applied Mathematics*, 161(16-17):2349–2360, 2013. doi:10.1016/j.dam.2013.03.021.

[2] A. Agrawal, D. Lokshtanov, and A. E. Mouawad. Critical node cut parameterized by treewidth and solution size is W[1]-hard. In *Proceedings of the 43rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG '17)*, volume 10520 of *Lecture Notes in Computer Science*, pages 32–44. Springer, 2017. doi:10.1007/978-3-319-68705-6_3.

[3] E. Angriman, A. van der Grinten, A. Bojchevski, D. Zügner, S. Günnemann, and H. Meyerhenke. Group centrality maximization for large-scale graphs. In *Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX '20)*, pages 56–69. SIAM, 2020. doi:10.1137/1.9781611976007.5.

[4] A. Arulselvan, C. W. Commander, L. Elefteriadou, and P. M. Pardalos. Detecting critical nodes in sparse graphs. *Computers & Operations Research*, 36(7):2193–2200, 2009. doi:10.1016/j.cor.2008.08.016.

[5] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

[6] M. Di Summa, A. Grosso, and M. Locatelli. Complexity of the critical node problem over trees. *Computers & Operations Research*, 38(12):1766–1774, 2011. doi:10.1016/j.cor.2011.02.016.

[7] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

[8] J. A. Enright and K. Meeks. Deleting edges to restrict the size of an epidemic: A new application for treewidth. *Algorithmica*, 80(6):1857–1889, 2018. doi:10.1007/s00453-017-0311-7.

[9] F. V. Fomin, P. A. Golovach, and J. H. Korhonen. On the parameterized complexity of cutting a few vertices from a graph. In *Proceedings of the 38th International Symposium on Mathematical Foundations of Computer Science (MFCS '13)*, volume 8087 of *Lecture Notes in Computer Science*, pages 421–432. Springer, 2013. doi:10.1007/978-3-642-40313-2_38.

[10] T. Fushimi, S. Okubo, and K. Saito. Multiple perspective centrality measures based on facility location problem under inter-group competitive environment. *Applied Network Science*, 5(1):80, 2020. doi:10.1007/s41109-020-00326-7.

[11] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976. doi:10.1016/0304-3975(76)90059-1.

[12] D. Hermelin, M. Kaspi, C. Komusiewicz, and B. Navon. Parameterized complexity of critical node cuts. *Theoretical Computer Science*, 651:62–75, 2016. doi:10.1016/j.tcs.2016.08.018.

[13] M. Lalou, M. A. Tahraoui, and H. Kheddouci. The critical node detection problem in networks: A survey. *Computer Science Review*, 28:92–117, 2018. doi:10.1016/j.cosrev.2018.02.002.

[14] E. Prieto-Rodríguez. *Systematic kernelization in FPT algorithm design.* PhD thesis, The University of Newcastle, 2005.

[15] J. Schestag. Critical node problem with vulnerable vertices. Master's thesis, Philipps-Universität Marburg, 2021.

[16] J. Schestag, N. Grüttemeier, C. Komusiewicz, and F. Sommer. On critical node problems with vulnerable vertices. In *Proceedings of the 33rd International Workshop on Combinatorial Algorithms (IWOCA '22)*, volume 13270 of *Lecture Notes in Computer Science*, pages 494–508. Springer, 2022. doi:10.1007/978-3-031-06678-8_36.

[17] Y. Shen, N. P. Nguyen, Y. Xuan, and M. T. Thai. On the discovery of critical links and nodes for assessing network vulnerability. *IEEE/ACM Transactions on Networking*, 21(3):963–973, 2013. doi:10.1109/TNET.2012.2215882.