# On Critical Node Problems with Vulnerable Vertices*

Jannik Schestag, Niels Grüttemeier, Christian Komusiewicz[0000−0003−0829−7032], and Frank Sommer **[0000−0003−4034−525X]

Fachbereich für Mathematik und Informatik, Philipps-Universität Marburg, Germany
{jschestag,niegru,komusiewicz,fsommer}@informatik.uni-marburg.de

**Abstract.** A vertex pair in an undirected graph is called *connected* if the two vertices are in the same connected component. 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 we can transform $G$ via at most $k$ vertex deletions 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 where at least one vertex is vulnerable. 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 case of deletable vulnerable vertices we provide a polynomial kernel for the parameter vc $+k$, where vc is the vertex cover number. In case of non-deletable vulnerable vertices, we prove NP-hardness even when there is only one vulnerable vertex.

## 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 [10]. One concrete 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?

* Most of the results of this work are also contained in the first author's Master's thesis [12].

** Supported by the DFG, project EAGR KO 3669/6-1.

One application of this formulation is to model the influence of vertices in the spreading of viruses in computer networks or social networks [10]. 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. This aspect is missing from the CNP problem. One way to model this aspect is to label some vertices as *vulnerable* and to consider only the number of connected pairs for the vulnerable vertices. In other words, we only count those vertex 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 it is not required that the set $A$ of vulnerable vertices is a subset of the vertex set $V$. Thus, given a graph $G = (V, E)$ with $A \subseteq V$ and a subgraph $G' = (V', E')$ of $G$, we may refer to the $A$-vulnerability of $G'$ 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 $G = (V, E)$, $A \subseteq V$, 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 $x$?

A further complication may be that, for several reasons, vulnerable vertices may not be removed. This is modelled by the following problem.

Critical Node Problem with Non-Deletable Vulnerable Nodes (CNP-NDV)
**Input**:A graph $G = (V, E)$, $A \subseteq V$, 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 $C$ is called a *critical node cut*. We study the parameterized complexity of these two problems.

*Related Work.* Arulselvan et al. [3] 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 [13]. In contrast, CNP can be solved in polynomial time on trees [5]

**Table 1.** Overview of our results.

| Parameter | CNP-V | CNP-NDV |
|---|---|---|
| $x$ | NP-hard for $x = 0$ [9] | W[1]-hard (Thm. 2) |
| | | XP (Prop. 2) |
| $y$ | FPT (Thm. 6) | W[1]-hard (Thm. 7) |
| | No poly kernel [9] | XP (Prop. 3) |
| $k$ | W[1]-hard [9] | 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. 3) | |
| vc | FPT (Thm. 8) | FPT (Thm. 8) |
| vc $+x$ | poly kernel (Cor. 5) | FPT (Thm. 8) |
| vc $+ k + x$ | | poly kernel (Cor. 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 $G$ [9]: On the negative side, CNP is W[1]-hard with respect to $k$ [9] or tw [9], and even with respect to $k + \text{tw}$ [2]. On the positive side, the problem is FPT with respect to $k + x$ and with respect to the parameter $y$ which is defined as $\ell - x$, where $\ell$ is the number of connected pairs in $G$. In other words, $y$ is the number of connected pairs that we want to remove at least by deleting the $k$ 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 [7]. For an overview of different formulations of critical vertex detection, refer to the survey of Lalou et al. [10].

*Our Results.* We study the parameterized complexity of the problems CNP-V and CNP-NDV with respect to a number of natural parameters. 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. We then show that, while being solvable in polynomial time for constant values of $x$, CNP-NDV is W[1]-hard with respect to $x$ even when $|A| = 1$. In contrast, CNP-V is solvable in polynomial time for constant $|A|$ and NP-hard already for $x = 0$. Thus, the complexity of the two problems differs quite drastically with respect to very natural parameters. This can be also observed for the parameter $y$ for which CNP-V has a subexponential FPT algorithm while CNP-NDV is W[1]-hard even with respect to $k + y$. 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 $y$ [9].

Finally, we consider parameterizations using the vertex cover number vc of $G$. This is motivated by the fact that CNP is W[1]-hard with respect to the treewidth tw [2, 9] and thus larger structural parameters need to be considered. We show that both problems are FPT with respect to vc, and provide polynomial kernels for both problems parameterized by $\mathrm{vc}+x$ and $\mathrm{vc}+k+x$, respectively.

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 [12]. Due to lack of space, the proofs of several results (marked with $(\star)$) are deferred to a full version.

*Preliminaries.* For two integers $p$ and $q$, $p \leq q$, we denote $[p, q] := \{p, \ldots, q\}$. We consider undirected simple graphs $G$ and let $V(G)$ denote the vertex set and $E(G)$ the edge set of a graph $G$. We use $n$ to denote the number of vertices of $G$ and $m$ to denote the number of edges. 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. For the relevant definitions of parameterized complexity refer to the standard monographs [4, 6].

## 2 Basic Observations

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

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

For 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.*

Moreover, for CNP-NDV at most $x$ non-vulnerable vertices can be connected to vulnerable vertices in $G-C$. Thus, one may 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 $x$.

**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. The first rule removes trivial components from the input.

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

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

**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*.*

The correctness of Rule 2 can be seen as follows: Since the instance is reduced with respect to Rule 1, every vertex of the graph is in at least one vulnerable connection. If we remove $y$ vertices, we remove at least $y$ vulnerable connections and therefore, the instance is a yes-instance. In case of CNP-NDV, we might not be able to remove $y$ vertices if $|V \setminus A|$ is too small. In this case we can trivially solve the instance by checking if $G - (V \setminus A)$ contains at most $x$ vulnerable connections. Hence, we may assume $y > k$ throughout the rest of this work.

In 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.

**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 his neighborhood belongs to every possible solution.

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

Recall that CNP-V and CNP-NDV can be solved in $\mathcal{O}(n^k \cdot (n + m))$ time due to Proposition 1. Since we can assume $y > k$ due to Rule 2 and, for CNP-V, $|A| > k$ due to Rule 3, 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 is 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 CNP-V when the connected components are small. Let $I := (G, A, k, x)$ be an instance of CNP-V, and let $C_1, \ldots, C_t \subseteq V$ be the connected components of the input graph $G$. The

*component information* $T[i, k']$ of some integers $i \in [1, t]$ and $k' \in [0, \min(k, |C_i|)]$ is defined as the minimal number of vulnerable connections in $G[C_i] - S$ among all subsets $S \subseteq C_i$ of size exactly $k'$. A table $T$ containing all component information $T[i, k']$ is called a *component table of the instance* $I$. We now show that CNP-V 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. [9] for CNP.

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

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.

**Proposition 4.** CNP-V *can be solved in* $\mathcal{O}(n \cdot k^2)$ *time if the input graph has maximum component size* $c$ *for some constant* $c$*.*

*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 ($\star$).** CNP-NDV *is* NP*-hard on planar graphs, even if the input graph has maximum degree* 4*.*

## 3 Parameterization by the Targeted Vulnerability

First, we consider parameterization by $x$ alone. CNP-V is NP-hard for $x = 0$ since it is a generalization of CNP. We now show that, in contrast, CNP-NDV is W[1]-hard with respect to $x$, even if $G$ contains only one vulnerable vertex.

**Theorem 2 ($\star$).** CNP-NDV *is* W[1]*-hard with respect to the parameter* $x$*, even if* $|A| = 1$ *and* diam $= 2$*.*

We now show an FPT algorithm for CNP-V and CNP-NDV parameterized by $k + x$. To this end, we consider the following more general problem.

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 $x$?

Hermelin et al. [9] 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. [9]. The next lemma describes the mechanism of the branching rule.

**Lemma 3 ($\star$).** *Let $I = (G = (V, E), A, N, k, x)$ be an instance of* CNP-VNDV *and let $v \in V \setminus N$. $I$ 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.

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

*Proof. Intuition:* In the algorithm we pick a neighbor $v$ of $N$ and branch into removing $v$ from the graph or making $v$ non-deletable.

*Algorithm:* **Step 0.** If $k < 0$ or the $A$-vulnerability of $G[N]$ is greater than $x$, return no. If the $A$-vulnerability of $G$ is at most $x$, return yes.

**Step 1.** Compute the set $N' \subseteq N$ such that $N'$ contains all vulnerable vertices $A$ and also all vertices that are connected to a vulnerable vertex in $G[N]$.

**Step 2.** If the neighborhood of $N'$ is empty, return yes. Otherwise, pick a neighbor $v$ of $N'$ and branch into the following instances: $I_1 := (G - \{v\}, A, N, k-1, x)$ and $I_2 := (G, A, N \cup \{v\}, k, x)$ of CNP-VNDV.

The correctness of the algorithm follows rather directly from Lemma 3. The running time can be seen as follows: The depth of the search tree is bounded by $k + x$ since in each branch, we either add a vertex to $N$ (which increases the $A$-vulnerability of $G[N]$ by at least one) or delete a vertex (which decreases $k$). Thus, the search tree has size $\mathcal{O}(2^{k+x})$; the steps at each search tree node can be clearly performed in linear time. □

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

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

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

In the following, we provide an algorithm that solves CNP-V in $\mathcal{O}((\frac{4}{3}x + 2)^k \cdot m \cdot x)$ 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 ($\star$).** *An instance $I := (G, A, k, x)$ of* CNP-V *can be solved in $\mathcal{O}((\frac{4}{3}x + 2)^k \cdot m \cdot x)$ time.*

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

**Corollary 3.** 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 $\ell$ is the $A$-vulnerability of the input graph. In other words, $y$ counts how many vulnerable connections shall be removed.

*An FPT Algorithm for Deletable Vulnerable Vertices.* CNP is fixed-parameter tractable with respect to $y$ [9], based on the following observations: If some connected component has at least $y$ vertices, then we have a yes-instance. Afterwards, 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 $C_1, \ldots, C_t$ be the connected components of $G$. Recall that we assume that $I$ is reduced regarding 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 $C_i$ of size at least $y$. Since we assume that every connected component of $G$ contains some vertices from $A$, let $v \in C_i \cap A$. Since $|C_i| \geq y$, we can remove at least $y$ vulnerable connections by deleting $v$. Together with the fact that $k \geq 1$ we conclude that the instance $I$ is a yes-instance. Throughout the rest of the proof, we assume that $|C_i| < y$ 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. With a component table at hand, we can then solve CNP-V in polynomial time due to Lemma 2. 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 $G[C_i]$ that remain after deleting exactly $k'$ vertices in $C_i$.

Let $C_i$ 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. Then, since there are at most $n$ connected components, the statement follows. We first consider the case where $k < \sqrt{y}$. Note that for each $k' \in [0, k]$, there are at most $\binom{|C_i|}{k'} \leq |C_i|^{k'}$ subsets $S \subseteq C_i$ of size $k'$. Since $|C_i| \leq y$ and $k' \leq k < \sqrt{y}$, we can compute all 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. Next, let $k \geq \sqrt{y}$. For this, we first identify a further case, where $I$ is a yes-instance.

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

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

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

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 $C_i$. Then, since $|C_i| \geq \frac{3\sqrt{y}+1}{2}$, the number of vulnerable connections is decreased by at least $\binom{\sqrt{y}}{2} + \sqrt{y} \cdot \left(\frac{3\sqrt{y}+1}{2} - \sqrt{y}\right) = y$. Therefore, $I$ is a yes-instance. ◇

Due to the previous case distinction, we may immediately return *yes* if $C_i$ 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 $|C_i| < \frac{3\sqrt{y}+1}{2}$ or $|C_i \cap A| < \sqrt{y}$. Consider the following cases.

**Case 1: $|C_i| < \frac{3\sqrt{y}+1}{2}$.** We can then compute the component information of the connected component $C_i$ by iterating over all subsets $S \subseteq C_i$ and computing the number of vulnerable connections in $G[C_i] - S$. Since $|C_i| < \frac{3\sqrt{y}+1}{2}$, 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: $|C_i \cap A| < \sqrt{y}$.** Then, since $k \geq \sqrt{y}$, we have $T[i, k'] = 0$ for all $k' \geq |C_i \cap A|$ since one may remove all vulnerable vertices in $C_i$ and afterwards, no vertex of $C_i$ is part of a vulnerable connection anymore. It remains to compute component information $T[i, k']$ with $k' < \sqrt{y}$ by iterating over every $S \subseteq C_i$ of size $k'$. Since there are at most $|C_i|^{k'}$ such subsets, this can be done in $y^{\sqrt{y}} \cdot n^{\mathcal{O}(1)} = 2^{\sqrt{y} \log y} \cdot n^{\mathcal{O}(1)}$ time.

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

*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 $k+y$, even if the input graph only contains one vulnerable vertex. We reduce from Clique which has as input graph $G$ and an integer $\ell$, and asks whether $G$ contains a set of $\ell$ vertices that are pairwise adjacent. It is well-known that Clique is W[1]-hard with respect to $\ell$ [4, 6].

The reduction follows the spirit of a reduction of Fomin et al. [8] that shows W[1]-hardness of the Cutting at most $k$ Vertices with Terminal problem. The reduction of Fomin et al. [8] 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 ($\star$).** CNP-NDV *is* W[1]*-hard with respect to the parameter* $k+y$*, even if* $|A| = 1$ *and the input graph has diameter 2.*

## 5 Parameters Related to the Vertex Cover Number

First, we obtain an FPT algorithm for the vertex cover number vc for the generalization CNP-VNDV of both problems via a combination of branching and dynamic programming.

**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 $G$. Then, we branch into all possible cases for $D := C \cap (S \setminus N)$. In other words, we consider all possible cases for vertex deletions in the vertex cover $S$. Consider one such possibility. 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 set $C'$ of at most $k'$ vertices such that $G' - C'$ has $A$-vulnerability at most $x$ and such that $C'$ contains no vertices of $N$. To answer this question, we use dynamic programming over subsets of $S'$. More precisely, we fill a dynamic programming table $T$ with entries of the type $T[S^*, k^*]$ where $S^*$ is a subset of $S'$ and $k^* \in \{1, \ldots, k'\}$. To define the meaning of a table entry, let $N_p(S^*)$, for $S^* \subseteq S'$ denote the neighbors of $S^*$ that are not neighbors of $S' \setminus S^*$. That is, $N_p(S^*) := N(S^*) \setminus N(S' \setminus S^*)$.

A table entry $T[S^*, k^*]$ contains the minimum $A$-vulnerability of any graph that is obtained from $G'[S^* \cup N_p(S^*)]$ by deleting at most $k^*$ vertices of $N_p(S^*) \setminus N$. 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 from $N(S') \setminus N$. If this number is smaller than $x$, then we have a yes-instance; otherwise, the CNP-VNDV instance has no critical node cut that contains $D$.

Informally, the recurrence to compute the value of $T[S^*, k^*]$ is to consider the possibilities of how one connected component created by the critical node cut may intersect with $S^*$. To simplify the description somewhat, we will define $T[S^*, k^*] = +\infty$ for all $k^* < 0$. The base cases of the recurrence are the $A$-vulnerability values that we get when $S^* \cup N_p(S^*)$ remains connected after the deletion of $k^*$ vertices. More precisely, let $Q[S^*, k^*]$ contain the minimum $A$-vulnerability of any connected graph that is obtained from $G'[S^* \cup N_p(S^*)]$ by deleting at most $k^*$ vertices of $N_p(S^*) \setminus N$. This value can be computed greedily by first deleting as many vertices of $(N_p(S^*) \setminus N) \cap A$ as possible and then deleting up to $k^* - |(N_p(S^*) \setminus N) \cap A|$ vertices of $(N_p(S^*) \setminus N) \setminus A$. Assuming that the values of $Q[S^*, k^*]$ have been precomputed, we may now compute $T[S^*, k^*]$ by the recurrence

$$T[S^*, k^*] = \min_{\tilde{S} \subseteq S^*} \min_{\tilde{k} \leq k^*} Q[\tilde{S}, \tilde{k}] + T[S^* \setminus \tilde{S}, k^* - \tilde{k} - \delta(\tilde{S}, S^* \setminus \tilde{S})]$$

where $\delta(\tilde{S}, S^* \setminus \tilde{S}) = |N(\tilde{S}) \cap N(S^* \setminus \tilde{S}) \cap N_p(S^*)|$ if

- $|N(\tilde{S}) \cap N(S^* \setminus \tilde{S}) \cap N_p(S^*)|$ 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 $\delta(\tilde{S}, S^* \setminus \tilde{S}) = k+1$, otherwise. That is, $\delta$ counts the number of vertex deletions that are necessary to disconnect $\tilde{S^*}$ and $S^* \setminus \tilde{S}$ in $G[S^* \cup N_p(S^*)]$ if it is possible to disconnect the two sets without deleting vertices in $N \cup S^*$. Otherwise, the value of $\delta$ is sufficiently large to ensure that the equation evaluates to $\infty$. . We omit a formal proof of the correctness and now bound the running time of the algorithm. A minimum vertex cover $S$ can be computed in $\mathcal{O}(2^{\mathrm{vc}}(n+m))$ time using the standard search tree algorithm. Afterwards, we consider every subset $D$ of $S$ and fill the table $T$ for the possibility where we delete exactly the vertex set $D$ from $S$. Filling the table needs $3^{\mathrm{vc}-|D|} \cdot n^{\mathcal{O}(1)}$ time since each evaluated term corresponds to a 3-partition of $S \setminus D$. Thus, the overall running time is $\sum_{i=0}^{\mathrm{vc}} \binom{\mathrm{vc}}{i} \cdot 3^{\mathrm{vc}-i} \cdot n^{\mathcal{O}(1)}$. Using the binomial theorem, the overall running time for all possibilites of $D$ is thus $4^{\mathrm{vc}} \cdot n^{\mathcal{O}(1)}$ time. □

Next, we show that CNP-V has a polynomial-size kernel for the parameter $\mathrm{vc} + x$ and that CNP-NDV has a polynomial-size kernel for the parameter $\mathrm{vc}+k+x$. 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 $Z$ which is a 2-approximation of the minimum vertex cover of $G$, that is, $|Z| \leq 2 \cdot \mathrm{vc}$. Note that $Z$ 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 $Z$.

**Rule 5** *Let $(G, A, k, x)$ be an instance of* CNP-V. *Return yes, if $k \geq 2 \cdot \mathrm{vc}$.*

Recall that we assume that the input instance of CNP-V is reduced with respect to Rules 1 and 4 and therefore we might 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 $\mathrm{vc}+x$.

**Lemma 4 (⋆).** *After 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 $(2\,\mathrm{vc}) \cdot ((2\,\mathrm{vc}) + \sqrt{2x} + 1)$ vertices.*

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 $B := A \cup Z$. For CNP-NDV, we define $B := Z$. We call $B$ the *base*. We then have $|B| \leq 2 \cdot \mathrm{vc}$ when we deal with an instance of CNP-NDV and by Lemma 4 we have $|B| \leq (2\,\mathrm{vc}) \cdot ((2\,\mathrm{vc}) + \sqrt{2x} + 2)$ 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 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. This rule should only be applied on instances of CNP-NDV.

**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 5 ($\star$).** *For an instance of* CNP-NDV*, Rule 6 is safe and 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 [11]. We use the formulation by Cygan et al. [4].

**Lemma 6 (Expansion Lemma [4]).** *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 $C$ 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 $T$.*

*Let $q \geq 1$ be a positive integer and $H$ be a bipartite graph with vertex bipartition $(R, T)$ such that $|T| \geq q \cdot |R|$ 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 Rules 1 and 6 are exhaustively applied.

**Rule 7** *If the set $Y$ contains at least $(k + x + 2) \cdot |B|$ vertices, then, in the graph $G'$ that we defined before this reduction rule, 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 7.** *For an instance of* CNP-V *or* CNP-NDV*, 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 $|Y| \geq (k + x + 2) \cdot |B|$ is correct. Let $G'$ be the graph defined before this reduction rule.

We start by showing that we can apply the Expansion Lemma. After 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 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.

**Theorem 9 (⋆).** *An instance $(G, A, k, x)$ of* CNP-V *or* CNP-NDV *contains less than $|B| \cdot (k + x + 3)$ vertices after Rules 1, 6, and 7 have been applied exhaustively.*

Since $|B| \leq |A| + 2 \cdot \mathrm{vc}$ for CNP-V and due to Lemma 4, we obtain the following.

**Corollary 4.** *For an instance $(G, A, k, x)$ of* CNP-V*, we can compute a kernelization with less than $((2\,\mathrm{vc})((2\,\mathrm{vc}) + \sqrt{2x} + 1) \cdot (k + x + 3)$ vertices in polynomial time.*

Since the instance is reduced regarding Rule 5, we obtain the following.

**Corollary 5.** *For an instance $(G, A, k, x)$ of* CNP-V*, we can compute a kernelization with less than $((2\,\mathrm{vc})((2\,\mathrm{vc}) + \sqrt{2x} + 1) \cdot (2\,\mathrm{vc} + x + 3)$ vertices in polynomial time.*

Since $|B| \leq 2 \cdot \mathrm{vc}$ for CNP-NDV, we obtain the following.

**Corollary 6.** *For an instance $(G, A, k, x)$ of* CNP-NDV*, we can compute a kernelization with less than $2 \cdot \mathrm{vc} \cdot (k + x + 3)$ vertices in polynomial time.*

## 6 Conclusion

We introduced two new critical node detection problems problems Critical Node Problem with Vulnerable Nodes (CNP-V) and Critical Node Problem with Non-Deletable Vulnerable Nodes (CNP-NDV), that take into accout that we are 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 a number of 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?

## References


1. Addis, B., Di Summa, M., Grosso, A.: 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)
2. Agrawal, A., Lokshtanov, D., Mouawad, A.E.: 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). Lecture Notes in Computer Science, vol. 10520, pp. 32–44. Springer (2017)
3. Arulselvan, A., Commander, C.W., Elefteriadou, L., Pardalos, P.M.: Detecting critical nodes in sparse graphs. Computers & Operations Research **36**(7), 2193–2200 (2009)
4. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer (2015)
5. Di Summa, M., Grosso, A., Locatelli, M.: Complexity of the critical node problem over trees. Computers & Operations Research **38**(12), 1766–1774 (2011)
6. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science, Springer (2013)
7. Enright, J.A., Meeks, K.: Deleting edges to restrict the size of an epidemic: A new application for treewidth. Algorithmica **80**(6), 1857–1889 (2018)
8. Fomin, F.V., Golovach, P.A., Korhonen, J.H.: 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). pp. 421–432. Springer (2013)
9. Hermelin, D., Kaspi, M., Komusiewicz, C., Navon, B.: Parameterized complexity of critical node cuts. Theoretical Computer Science **651**, 62–75 (2016)
10. Lalou, M., Tahraoui, M.A., Kheddouci, H.: The critical node detection problem in networks: A survey. Computer Science Review **28**, 92–117 (2018)
11. Prieto-Rodríguez, E.: Systematic kernelization in FPT algorithm design. Ph.D. thesis, The University of Newcastle (2005)
12. Schestag, J.: Critical Node Problem with Vulnerable Vertices. Master's thesis, Philipps-Universität Marburg (2021)
13. Shen, Y., Nguyen, N.P., Xuan, Y., Thai, M.T.: On the discovery of critical links and nodes for assessing network vulnerability. IEEE/ACM Transactions on Networking **21**(3), 963–973 (2013)