Faculty for Mathematics and Computer Science
Philipps-University Marburg

Master Thesis

# Critical Node Problem with Vulnerable Nodes

*Author:*
Jannik Schestag

*First Reviewer:*
Prof. Dr. Christian Komusiewicz

submitted: October 25, 2021

*in the subject Computer Science*
*with the working group for Algorithms*

## Selbstständigkeitserklärung

Hiermit versichere ich, Jannik Schestag, dass ich die vorliegende Arbeit mit dem Titel *Critical Node Problem with Vulnerable Nodes* selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Masterarbeit wurde in der jetzigen oder einer ähnlichen Form noch bei keiner anderen Hochschule eingereicht und hat noch keinen sonstigen Prüfungszwecken gedient.

Unterschrift von Jannik Schestag

Ort, Datum

## Abstract

During a pandemic, it is crucial to decide whom to vaccinate first to limit the spread of the virus to a minimum. The CRITICAL NODE PROBLEM (CNP) is applicable for modeling such a situations. The CNP receives a graph and two integers $k$ and $x$ as input. It is asked whether $k$ vertices can be cut from the graph such that in the remaining graph at most $x$ pairs of vertices are connected. In this work, we propose two graph-theoretic problems that can model pandemics more precisely. In addition to the input of the CNP, the new problems receive a set of vertices $A$ as input. After the cut only connected pairs with at least one vertex in $A$ are counted. In one of the new problems, we can cut vertices of $A$ while this is forbidden in the other. We examine the classification of these two problems within the framework of complexity theory. We show that the problems are fixed-parameter tractable with respect to various parameters, as for example $k + x$. Furthermore, we show that the problems are W[1]-hard with regard to several parameters. It is notable that the problem variant in which vertices of $A$ must not be cut is W[1]-hard with respect to $k$ on bipartite graphs and split graphs, even if $A$ contains only one vertex.

## Zusammenfassung

In Situationen wie Pandemien ist es wichtig zu entscheiden, wer zuerst geimpft werden soll, um die Verbreitung des Virus auf ein Minimum zu beschränken. Das CRITICAL NODE PROBLEM (CNP) ist für die Modellierung solcher Situationen geeignet. Das CNP erhält einen Graphen und zwei ganzen Zahlen $k$ und $x$ als Eingabe. Es wird gefragt, ob $k$ Knoten aus dem Graphen gelöscht werden können, sodass im verbleibenden Graphen höchstens $x$ Paare von Knoten verbunden sind. In dieser Arbeit schlagen wir zwei graphentheoretische Probleme vor, die Pandemien genauer modellieren können. Zusätzlich zur Eingabe des CNP erhalten die neuen Probleme eine Menge an Knoten $A$ als Eingabe. Nach der Löschung werden nur verbundene Paare mit mindestens einem Knoten aus $A$ gezählt. In einem der neuen Probleme können wir Knoten von $A$ löschen, während dies im anderen Problem verboten ist. Wir klassifizieren diese beiden Probleme im Rahmen der Komplexitätstheorie. Weiterhin zeigen wir, dass die Probleme in Bezug auf verschiedene Parameter, wie zum Beispiel $k + x$, in der Klasse FPT liegen. Außerdem zeigen wir, dass die Probleme in Bezug auf verschiedene Parameter W[1]-schwer sind. Hervorzuheben ist, dass die Problemvariante, bei der kein Knoten von $A$ gelöscht werden darf, W[1]-schwer bezüglich $k$ auf bipartiten Graphen und Split-Graphen ist, selbst wenn $A$ nur einen Knoten enthält.

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation for the Critical Node Problem

Diseases are a threat for the well-being of all parts of society. Recently, we saw the devastating effect of the Covid-19 pandemic on politics, social life, economics, and, most importantly, it threatened people's lives all over the planet. What worsened the situation was that a vaccination had to be developed over several month and even afterwards the number of doses was limited. It was tough for epidemiologists and politicians to decide in which areas people would have to be vaccinated first, in order to minimize the number of people affected by the coronavirus [CHBA03, ACEP09, HKKN16, NSB16, VHOB18].

This is one of several scenarios that occur in the world in which graph-theoretic concepts can not only model the situation, but also point to a suitable reaction. In a graph, we model objects as vertices and we add a line, which is called an edge, between two objects that have some sort of relationship. One aspect we want to focus on is whether it is possible to take out only a few objects from a scenario so that only a limited number of pairs of the remaining objects are still connected. In the above example about the Covid-19 outbreak, geographic areas could be modeled as vertices of the graph and between two vertices an edge is modeled, if a road connects two geographic areas. The vertices that we want to search are the geographic areas in which we want to vaccinate people such that the spread of the disease is limited. In the following, we will discuss some more scenarios and want to explain why this method could be desired.

Social networks are an important area of modern social interaction. In a social network, the members of the network are modeled as vertices, and an edge exists, if two people know each other. So-called *influencers* know many people within the network. Therefore, if influencers leave a social network, the flow of information in the remaining network is impeded. With the graph theoretic concept that we want to discuss, the owner of a social network can find the influencers of their social network to give these influencers benefits when staying active. Keeping influencers active in a social network is important for the owner to ensure the networks continued stability – which is important for better advertisements, predictions of upcoming elections, and, as already mentioned, prevention of spreading diseases [ACEP09, BC09, AGH16].

In the modern world, it is unimaginable not to have connection to the internet for a prolonged time. In order to keep everyone online, a lot of internet routers have to be operating at the same time. In a communication network, these routers can be modeled as vertices and, if they are physically connected, we model an edge between the two vertices. Losing specific routers may result in the breakdown of the communication network. Therefore, it is important to compute which routers are most crucial to the network. The set of vertices that we want to search are the routers that, when taken out of the communication network, would minimize the remaining network. Thus, when the provider knows which routers are crucial, these routers can be reinforced to strengthen the network against attacks and physical failure [CHBA03, BC09, DSGL12, NSB16, VHOB18].

Special networks with high priority for security agencies are terrorist networks. Within a terrorist network, terrorists can be modeled as vertices and those who know each other as edge. Since catching terrorists and keeping them in prisons is very expensive and can sometimes even cause political tension, only a limited number of terrorists can be caught. With knowledge of the network, we can compute the set of vertices that is the optimal group of terrorists to be targeted so that the network as a whole will have less impact [ACEP09].

Another application of graph-theoretic concepts lies in understanding medical drug design, especially protein-protein interactions. In this scenario, proteins are modeled as vertices. If proteins are interacting, we model an edge between them. In drug design, it is useful to understand which proteins should be treated to have a certain desired biological effect. Knowing the most important proteins may provide medics with a clue about how to proceed with the treatment [BC09, VHOB18].

Finally, we want to mention the field of transportation engineering. Since the difference between roads, rails, conveyor belts, rivers, and so on is marginal in this case, we will only describe how to model transportation over railroads. In this scenario, every switch and station is modeled as vertex. If there is a rail between either switches or stations, we add an edge to the graph. For transportation of goods, it is essential to not run into a shortage because an important switch or station broke down. The set of vertices that we want to compute are some switches and stations that have the capability to break down the transportation system. For reasons of maintenance, these critical switches and stations are to be checked for robustness and may be reinforced in a higher frequence. Therefore, passengers can travel and goods can be delivered as planned. Having an intact rail system is also important for emergency and rescue concepts [ACEP09, BC09, NSB16].

One well-studied graph-theoretic problem that can model all of these scenarios is the CRITICAL NODE PROBLEM (CNP). The problem receives a simple undirected graph as input for which we ask, if it is possible to delete a small node set $C$, called *critical node cut*, in order to minimize the number of pairs of vertices which are connected in the remaining graph. Two vertices $v_0$ and $v_t$ are connected in a graph $G$, if vertices $v_1, \dots, v_{t-1}$ exists such that $\{v_i, v_{i+1}\}$ is an edge of $G$ for every $i < t$. The vertex set $C$ in the above problems would be the areas to be vaccinated, a set of influencers to put more focus on, routers to be reinforced, terrorists to be caught or the stations and switches to receive the most attention for maintenance.

Since the importance of the CNP has now been explained sufficiently, a formal definition is in order:

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? |

**Introduction of Vulnerable Vertices** In some situations, the model of the CNP is not sufficient. Take as an example the Covid-19 pandemic in 2020. In the following, we present a situation that is not necessarily medically correct, but exhibits some similarities to reality. We suppose that the virus is only deadly to a part of the population, such as elderly people, and the rest of the population can spread the virus, but are not effected by it themselves. In order to adjust the model to that situation, we consider a subset of the population for which the connections have to be minimized and we do not care about how many connections the rest of the population has. As the production and distributions of vaccinations took some time, it was only possible to vaccinate few people. Thus, it was not easy to decide who was allowed to be vaccinated. We model every person as vertex and people who are in contact with each other as edges. Elderly people are the subset of people whose connections should be minimized. The question is who should be vaccinated in order to minimize the number of elderly people who die from Covid-19, for a given number of doses.

In the following, consider a simple undirected graph $G = (V, E)$ and a subset of the vertices $A \subseteq V$. We call the vertices of the set $A$ *vulnerable*. In the situation we have described, $A$ is the set of elderly people.

In the above scenario, we have several options to formulate a new problem. We can allow that vertices of $A$ are added to a critical node cut, or we can restrict vulnerable vertices to not be added to a critical node cut. If some elderly people have a lot of social contacts and we assume that the vaccines do not have a side-effect for endangered people, vaccinating at least some of these elderly people could bring the desired effect of removing a lot of danger.

In the following we define *connectivity of a graph* $G$ as the number of connected pairs of vertices in $G$ with at least one node in $A$.

The graph problem variant for the prevention of Covid-19 that we have described above can be defined as follows:

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 connectivity of $G - C$ is at most $x$? |

Informally, the goal in this problem is to detect a small subset $C$ that is removed so that there is a minimal number of connections left which lead to a node in $A$.

We will now describe another scenario. Imagine that the elderly people to whom Covid-19 is the most dangerous might have bad side-effects after the vaccination. Thus, we can not vaccinate elderly people and, in consequence, the subsets $C$ and $A$ have to be disjoint. It follows that $C$ has to be picked from $V \setminus A$. The problem is defined as follows:

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 connectivity of $G - C$ is at most $x$?

For both problems, we demand the set $A$ to be given as input in the form of a list or similar structure. We add to all vertices $v$ a Boolean constant $c$ to indicate whether $v$ is vulnerable. Then, we can iterate over $A$ once and set $c$ true for every vulnerable vertex. This requires $\mathcal{O}(|A|)$ time. Afterwards, we can iterate over $A$ in linear time and check whether a vertex $v$ is vulnerable in constant time.

In the rest of the thesis, we will examine CNP-V and CNP-NDV within the framework of parameterized complexity.

## 1.2 Related Work

In recent years, a lot of research has been done about the Critical Node Problem. We want to give an overview of the research and focus it on parameterized complexity, since we deal with parameterized complexity in this thesis.

It has been shown by Arulselvan et al. [ACEP09] that CNP is NP-complete, as they gave a reduction from Vertex Cover to CNP.

Hermelin et al. [HKKN16] proved that, for several parameters, CNP is fixed-parameter tractable and admits polynomial size kernelizations. The considered parameters are the maximum number of deleted nodes $k$, the number of remaining connected pairs $x$, the number of pairs to delete $y$, and the treewidth $\omega$ of the input graph. The main results of their paper are that CNP is W[1]-hard with respect to the single parameters $k$, $x$, and $\omega$, and that CNP is fixed-parameter tractable with respect to the parameters $y$, $k+x$, $k+\omega$, and $x+\omega$. They showed that CNP is W[1]-hard with respect to $k$ by reducing from Clique. To show that CNP is W[1]-hard with respect to $\omega$, they reduced from W[1]-complete Multicolored Clique. As CNP with parameter $x = 0$ is the NP-complete problem Vertex Cover, we know that CNP is para-NP-hard with respect to the parameter $x$. Also, they showed that there is little hope that CNP could admit a kernelization of polynomial size with respect to $x + \omega$, or $k + y + \omega$.

Addis et al. [ADSG13] showed that a weighted version of CNP (from now on called node-weighted CNP), where every vertex has a non-negative weight and a critical node cut can contain a certain weight at most, is NP-hard on split graphs, on bipartite graphs, and on complements of bipartite graphs. They also showed that the node-weighted CNP can be solved in polynomial time on the family of graphs that have treewidth bounded by a given constant.

Di Summa et al. [DSGL11] showed that the node-weighted CNP is solvable in polynomial time on trees. Also, they showed that the problem is

NP-complete, if every connected pair of vertices has a certain weight and a certain budget can not be exceeded in the remaining graph (from now on we call this version CONNECTION-WEIGHTED CNP). These results have been improved by Lalou et al. [LTK16]. They showed that a related problem 3C-CNP can be solved in linear time on trees. Furthermore, Lalou et al. [LTK16] showed that 3C-CNP can be solved on proper interval graphs in polynomial time.

Guan et al. [GLZ21] showed that CNP can be solved in polynomial time on trees, if the biggest component contains at most $c$ vertices where $c$ is a constant.

The CNP has also been researched within the field of integer linear programming [ACEP09, BC09, DSGL12, NSB16, VHOB18]. In order to solve integer linear programming faster, Di Summa et al. [DSGL12] introduced additional inequalities which can be applied correctly for special appearances of some parts of the graph which are for example cliques and cycles. For a related version of the problem, Ventresca et al. [VHOB18] considered that, for a bi-objective problem definition, an idea is aiming to split the graph into as many connected components as possible while having a low variance in their cardinalities.

Heuristic algorithms have been presented for CNP [ACEP09, BC09, ZH17, AGH16]. Arulselvan et al. [ACEP09] have proposed an efficient combinatorial heuristic for the CNP. Zhou and Hao [ZH17] proposed a heuristic for which they showed empirically that their algorithm is highly competitive with the state of the art results. Aringhieri et al. [AGH16] provided an evolutionary algorithm. Their algorithm ran in ten of sixteen common instances in the best known time.

Ventresca and Aleman [VA14] introduced a randomized rounding-based approximation algorithm which achieves an approximationratio within constant bounds.

Nabli et al. [NCH20] described several versions of the CRITICAL NODE PROBLEM, which are closer to a two-player game, where an attacker tries to *infect* as many vertices as possible and a defender tries to *vaccinate* vertices in a manner that the fewest number of vertices are infected in the end. They showed for which versions the problem the problem is NP-complete or on a higher level of the polynomial hierarchy.

## 1.3 Preliminaries

In this section, we provide the basic graph-theoretic concepts, the basic concepts of the complexity theory, and of some problem definitions used in the rest of this thesis. We define $[\ell, t] := \{i \in \mathbb{N} \mid \ell \leq i \leq t\}$.

For every object that we define with a graph as subscript, we do not write the subscript, if the context makes clear which graph we refer to.

### 1.3.1 Graph Theory

**Graphs** A *graph* $G = (V, E)$ consists of a set of *vertices* $V$ and a set of *edges* $E$. Each edge $e \in E$ is a set of two distinct vertices and exists at most once per pair. We use the notation $V(G)$ for the set of vertices of $G$ and the notation $E(G)$ for the set of edges of $G$. We define $n_G := |V(G)|$, $m_G := |E(G)|$. For vertices $v, w \in V$ an edge $\{v, w\} \in E$ is *incident with* $v$ and $w$ and the vertices $v$ and $w$ are *adjacent*. For a given subset of the vertices $V' \subseteq V$ we denote the *induced subgraph* by $G[V'] = (V', E')$ where the edges of $G[V']$

are $E' := \{\{u,v\} \in E(G) \mid u,v \in V'\}$. For a subset $W \subseteq V$, we define $G - W$ as the induced subgraph $G[V \setminus W]$. For two graphs $G$ and $H$ we call the graph $H$ *isomorphic to* $G$, if a bijection $f : V(G) \to V(H)$ exists such that $\{u,v\}$ is an edge of $G$, if and only if $\{f(u), f(v)\}$ is an edge of $H$. If $H$ is isomorphic to an induced subgraph of $G$, we say that $G$ *contains* $H$. A *path* $P_\ell$ *of length* $\ell$ with $\ell \geq 2$ is the graph with $\ell$ vertices $V := \{v_1, \ldots, v_\ell\}$ and edges set $E := \{\{v_i, v_{i+1}\} \mid i \in [1, \ell-1]\}$. Two vertices $u, v \in V$ are *connected*, if $G$ contains a path $P$ and $u, v \in V(P)$. A graph $G = (V, E)$ is *connected*, if every two vertices $u, v \in V$ are connected. A *connected component* $C = (V', E')$ *of size* $\ell$ of a graph $G = (V, E)$ is a connected induced subgraph of $G$ and for an edge $\{u, v\} \in E(G)$ follows $u, v \in V'$ or $u, v \notin V'$. A vertex $v \in V$ is called *universal*, if $\{v, w\}$ is an edge for every $w \in V \setminus \{v\}$.

**Structure of Graphs** For any vertex $u$, the *degree* of $u$ is the number $\deg(u) := |\{\{u, u'\} \in E \mid u' \in V\}|$. The *maximum vertex degree* of a graph is defined as $\Delta := \max\{\deg(u) \mid u \in V\}$. Two edges $e_1, e_2$ are *adjacent*, if there is a vertex $v \in V$ such that $v$ is incident with $e_1$ and $e_2$. For two vertices $v, w \in V$ of a graph $G$, the *distance* $d(v, w)$ is the smallest integer $d$ such that $G$ contains a path of length $d$ in which $v$ and $w$ are the two vertices with degree 1. If $v$ and $w$ are not connected, we define $d(v, w) = \infty$. The *diameter* diam of a connected graph $G = (V, E)$ is the largest integer such that vertices $v, w \in V$ exist with distance diam. For a graph $G$ that is not connected, we define diam $= \infty$.

If $\{u, v\} \in E$ is an edge, we call $u$ a *neighbor* of $v$. For a set of vertices $S$, the *open neighborhood of* $S$ is the set of vertices $N_G(S)$ that contains all vertices that are a neighbor of at least one vertex of $S$ in the graph $G$ that are not contained in $S$. The *closed neighborhood of* $S$ is $N_G[S] := N_G(S) \cup S$. For a vertex $v \in V$, we write $N_G(v)$ for $N_G(\{v\})$ and $N_G[v]$ for $N_G(v) \cup \{v\}$. The *$\ell$-neighborhood of $S$ is the set of vertices* $N_G^\ell[S]$ that contains all vertices $v \in V$ for which the distance to some vertex $u \in S$ is at most $\ell$.

**Notation for vulnerable vertices** A vertex $v \in A$ is called *vulnerable* and a vertex $w \notin A$ is called *non-vulnerable.* If two vertices $d \in A$ and $v \in V$ are connected in a graph $G$, we say that $\{d, v\}$ is a *vulnerable connection in* $G$. For a graph $G$, by the *connectivity of* $G$, we denote the number of vulnerable connections in $G$. A *trivial connected component* is a connected component of $G$ that does not contain any vulnerable vertex or has a size 1. In instances of CNP-NDV, connected components that do not contain a non-vulnerable vertex are also trivial connected components.

**Graph Classes** A *clique* $K_\ell$ *with* $\ell$ *vertices* is the graph $G = (V, E)$ with $\ell$ vertices $V := \{v_1, \ldots, v_\ell\}$ which are pairwise adjacent. An *independent set with* $\ell$ *vertices* is the graph $G = (V, E)$ with $\ell$ vertices and no edges. A *cycle* $C_\ell$ *of length* $\ell$ is the graph $G = (V, E)$ with vertices $V := \{v_1, \ldots, v_\ell\}$ such that the set of edges is $E := \{\{v_1, v_\ell\}\} \cup \{\{v_i, v_{i+1}\} \mid i \in [1, \ell]\}$. A *$q$-star with center* $u$ is the graph $G = (V, E)$ with $q + 1$ vertices $V := \{u, v_1, \ldots, v_q\}$ and the set of edges $E := \{\{u, v_i\} \mid 1 \leq i \leq q\}$. A *tree with root* $w$ is a connected graph $T = (V, E)$ with a *root* $w \in V$ which does not contain a cycle $C_\ell$ for $\ell > 2$. The *height* $h$ of a vertex $v \in V$ is the distance from $w$ to $v$. A neighbor $u$ of $v \in V$

is called the *parent* of $v$, if the height of $u$ is smaller than $v$. A neighbor $u$ of $v \in V$ is called a *child* of $u$, if the height of $u$ is greater than $v$. A vertex $v \in V$ that is not the root of the tree is a *leaf*, if $v$ has degree 1. The *subtree* $st(T, v)$ is the graph $T$, if $v = w$. Otherwise, $st(T, v)$ is the connected component of $v$ in the graph $T - \{u\}$, where $u$ is the parent of $v$. A *forest* is a graph $G$ in which all connected components are trees. A *planar graph* can be drawn in in the Euclidean plane such that no edges are crossing. A graph $G = (V, E)$ is *bipartite with vertex bipartition* $(V_1, V_2)$, if $V_1 \dot\cup V_2 = V$ and $E(G) \subseteq \{\{v, w\} \mid v \in V_1, w \in V_2\}$. A *complete bipartite graph* $K_{k,\ell}$ describes the bipartite graph $G = (V, E)$ with vertex bipartition $(V_1, V_2)$ where $|V_1| = k, |V_2| = \ell$, and vertices of $V_1$ and $V_2$ are pairwise connected.

For further introduction to graph-theoretic concepts we refer to Diestel [Die17].

### 1.3.2 Complexity Theory

A *language* $L$ is a subset of $\Sigma^*$ where $\Sigma$ is a *finite alphabet*. Every language $L$ defines a *binary decision problem* $P_L : \Sigma^* \to \{\texttt{true}, \texttt{false}\}$ by $P(x) := \texttt{true}$, if and only if $x \in L$. We call $x$ *a yes-instance of* $P_L$, if $x \in L$. Let $F, G : \Sigma^* \to \{\texttt{true}, \texttt{false}\}$ be two binary decision problems. A *polynomial-time reduction* $h : \Sigma^* \to \Sigma^*$ is a computable function that for every $x \in \Sigma^*$ computes the result $h(x)$ in running-time polynomial in $|x|$ such that $x$ is a yes-instance of $F$, if and only if $h(x)$ is a yes-instance of $G$.

P and NP are two classes of binary decision problems. A binary decision problem $T$ is *in* P, if an algorithm $\mathcal{A} : \Sigma^* \to \{\texttt{true}, \texttt{false}\}$ exists so that for every $x \in \Sigma^*$ the algorithm $\mathcal{A}$ has a running-time polynomial in $|x|$ and it is $T(x) = \mathcal{A}(x)$. A binary decision problem $T$ is *in* NP, if an algorithm $\mathcal{A} : \Sigma^* \times \Sigma^* \to \{\texttt{true}, \texttt{false}\}$ and a polynomial $p : \mathbb{N} \to \mathbb{N}$ exist such that for every $x \in \Sigma^*$ the running-time of $\mathcal{A}$ is polynomial in $|x|$ and $T(x) = \texttt{true}$, if and only if a value $u \in \Sigma^*$ exists such that $|u| \leq p(|x|)$ and $\mathcal{A}(x, u) = \texttt{true}$. We call $u$ a *certificate for $x$ with respect to $T$ and $\mathcal{A}$*. Therefore, it follows P $\subseteq$ NP. It is assumed that NP $\neq$ P and some problems which are in NP are therefore assumed not to be in P. A binary decision problem $T$ is called NP-*hard*, if for every binary decision problem $S \in$ NP a polynomial-time reduction $h : S \to T$ exists. A problem $T$ is called NP-*complete*, if $T$ is NP-hard and $T$ is in NP.

### 1.3.3 Parameterized Complexity

**Fixed-Parameter Tractability** A *parameterized language $L$ for the parameter $\lambda$* is a subset of $\Sigma^* \times \mathbb{N}$. A parameterized language $L$ for the parameter $\lambda$ induces a *parameterized problem $P$ for the parameter $\lambda$* as a mapping of $P : \Sigma^* \times \mathbb{N} \to \{\texttt{true}, \texttt{false}\}$ where $P((I, k)) := ((I, k) \in L)$. We call $I$ an *input*, $k$ a *parameter* and $(I, k)$ an *instance* of $P$. We call $(I, k)$ *a yes-instance of $P$*, if $(I, k) \in L$, and a *no-instance* otherwise. For an algorithm $M : \Sigma^* \times \mathbb{N} \to \{\texttt{true}, \texttt{false}\}$, we say $M$ *returns yes on* $(I, k)$, if $M((I, k)) = \texttt{true}$, and otherwise we say $M$ *returns no on* $(I, k)$. In the rest of this thesis we briefly write problem for a paramerized problem. A parameterized problem $P : \Sigma^* \times \mathbb{N} \to \{\texttt{true}, \texttt{false}\}$ for the parameter $\lambda$ is called *fixed-parameter tractable with respect to* $\lambda$, if a deterministic algorithm $\mathcal{A} : \Sigma^* \times \mathbb{N} \to \{\texttt{true}, \texttt{false}\}$, a computable function $f : \mathbb{N} \to \mathbb{N}$, and a constant $c$ exist such that the algorithm $\mathcal{A}$ returns yes on an instance $(I, k) \in \Sigma^* \times \mathbb{N}$, if and only if $P((I, k)) = \texttt{true}$ and $\mathcal{A}$ has a running time bounded by $f(k) \cdot |(I, k)|^c$. The algorithm $\mathcal{A}$ is called *fixed-parameter algorithm*. The complexity class containing all fixed-parameter tractable problems is called *FPT*.

**Slice-wise polynomial** A parameterized problem $P : \Sigma^* \times \mathbb{N} \to \{\texttt{true}, \texttt{false}\}$ for the parameter $\lambda$ is called *slice-wise polynomial with respect to the parameter* $\lambda$, if an algorithm $\mathcal{A}$ and two computable functions $f, g : \mathbb{N} \to \mathbb{N}$ exist such that the algorithm $\mathcal{A}$ on an instance $(I, k) \in \Sigma^* \times \mathbb{N}$ returns yes, if and only if $P((I, k)) = \texttt{true}$ and $\mathcal{A}$ has a running time time bounded by $f(k) \cdot |(I, k)|^{g(k)}$. The complexity class containing all slice-wise polynomial problems is called *XP*.

**para-NP-hardness** We call a parameterized problem $P$ for the parameter $\lambda$ *para-*NP*-hard with respect to the parameter* $\lambda$, if the binary decision problem $P_c : \Sigma^* \to \{\texttt{true}, \texttt{false}\}, P_c(I) \mapsto P(I, c)$ is NP-hard for some constant $c \in \mathbb{N}$.

**Branching Rule** For a problem $P : \Sigma^* \times \mathbb{N} \to \{\texttt{true}, \texttt{false}\}$ a computable function $f : \Sigma^* \times \mathbb{N} \to (\Sigma^* \times \mathbb{N})^t$ is called a *branching rule*, if $f$ maps $((I, k))$ to $((I_1, k_1), \ldots, (I_t, k_t))$ where $(I_i, k_i)$ is an instance of $P$ for every $i \in [1, t]$. A branching rule is called *correct*, if $(I, k)$ is a yes-instance of $P$, if and only if an $i \in [1, t]$ exists such that $(I_i, k_i)$ is a yes-instance of $P$. Applying branching rules builds up a *search tree* where the cumulative number of inputs is called the *size of the search tree*.

**Parameterized reduction** Let $A, B : \Sigma^* \times \mathbb{N} \to \{\texttt{true}, \texttt{false}\}$ be two parameterized problems. A *parameterized reduction from* $A$ *to* $B$ is an algorithm $\mathcal{A}$ that, given an instance $(I, k)$ of $A$, outputs an instance $(I', k')$ of $B$ such that

1. $(I, k)$ is a yes-instance of $A$ if and only if $(I', k')$ is a yes-instance of $B$
2. $k' \leq g(k)$ for some computable function $g$, and
3. the running time of $\mathcal{A}$ is bounded by $f(k) \cdot |x|^c$ for a computable function $f$ and some constant $c \in \mathbb{N}$.

**W[1]-hardness** We call a parameterized problem $P$ for the parameter $\lambda$ W[1]*-hard with respect to* $\lambda$, if a W[1]-hard problem $Q$ for the parameter $\mu$ and a parameterized reduction from $Q$ to $P$ exist. It is assumed that when a parameterized problem $P$ is W[1]-hard with respect to parameter $\lambda$ that $P$ is not fixed-parameter tractable with respect to the parameter $\lambda$.

**Kernelization** Given a parameterized problem $P : \Sigma^* \times \mathbb{N} \to \{\texttt{true}, \texttt{false}\}$ for the parameter $\lambda$, an algorithm $\mathcal{A} : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ is called *kernelization* for the parameter $\lambda$, if for every instance $(J, k)$ of $P$ the algorithm $\mathcal{A}$ computes an equivalent instance $(J', k')$ with $|J'| + k' \leq f(k)$ for some computable function $f$ and $\mathcal{A}$ has a polynomial running time. If the function $f$ is a polynomial, we say that $P$ *admits a kernelization of polynomial size.* A parameterized problem $P$ for the parameter $\lambda$ admits a kernelization for the parameter $\lambda$, if and only if $P$ is fixed-parameter tractable with respect to $\lambda$ [CFK+15].

For a problem $P$, a computable function $f : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ is called a *reduction rule*, if $f((I, k)) = (I', k')$ where $(I', k')$ is an instance of $P$. A reduction rule is called *safe*, if $(I, k)$ is a yes-instance of $P$ if and only if $(I', k')$ is a yes-instance of $P$.

For further introduction to complexity theory and parameterized complexity we refer to Cygan et al. [CFK+15] and Niedermeier [Nie06].

## 1.4 Results

### 1.4.1 Known Results

Notice that CNP is the special case of CNP-V with $A = V$. In this subsection, we summarize hardness results that exist for CNP and are therefore correct for CNP-V. Hermelin et al. [HKKN16] showed that CNP is W[1]-hard with respect to the single parameters solution size $k$ and treewidth $\omega$. As CNP with $x = 0$ is the NP-hard Vertex Cover problem, CNP is para-NP-hard with respect to the parameter $x$. Unless coNP $\subseteq$ NP/poly, the problem CNP has no kernelization of polynomial size for the parameters $x + \omega$ and $k + y + \omega$.

It has been shown by Hermelin et al. [HKKN16] that, if a graph $G$ has no isolated vertices, then $x + y \in \Omega(n)$. If we remove all trivial connected components from a graph, the same is true for CNP-V and CNP-NDV. Thus, CNP-V and CNP-NDV are fixed-parameter tractable with respect to the parameter $x + y$.

### 1.4.2 Our Results

To the best of our knowledge, we are the first to examine the Critical Node Problem with vulnerable vertices. For an overview of all parameters considered in the work, we refer to Table 1.1. Table 1.2 briefly depicts our results.

**The parameter $|\mathbf{A}|$ and the dual parameter $|\mathbf{V} \setminus \mathbf{A}|$.** CNP-NDV is W[1]-hard on bipartite graphs with respect to the parameter $\overline{n} + k + y$, even if $|A| = 1$ (Theorem 4.1). Also, CNP-NDV is W[1]-hard with respect to the parameter $x$, even if $|A| = 1$ (Theorem 4.5). Thus, CNP-NDV is para-NP-hard with respect to the parameter $|A|$. Furthermore, we show that CNP-V is fixed-parameter tractable with respect to the parameter $|A| + x$, and CNP-NDV is fixed-parameter tractable with respect to the parameters $|A| + x + \Delta$ and the number of non-vulnerable vertices $|V \setminus A|$. CNP-NDV admits a kernelization with $\mathcal{O}(|V \setminus A| \cdot \sqrt{x})$ vertices (Theorem 5.6).

**The neighborhood-diversity nd and the vertex cover number vc.** We prove that CNP-V and CNP-NDV are fixed-parameter tractable with respect to the neighborhood-diversity nd (Corollary 3.13) and the vertex cover

Table 1.1: All parameters that we consider in this work.

| Parameter | Meaning |
|---|---|
| $\lvert A\rvert$ | Number of vulnerable vertices |
| $\lvert V \setminus A\rvert$ | Number of non-vulnerable vertices |
| nd | Neighborhood-diversity |
| vc | Vertex cover number |
| $\overline{n}$ | Size of the neighborhood of $A$ |
| $p$ | Number of vulnerable connections |
| $k$ | Maximum size of the vertex cover |
| $x$ | Maximal number of vulnerable connections that can remain in the graph |
| $y$ | Number of vulnerable connections to be deleted |
| $\Delta$ | Maximum vertex degree |
| diam | Diameter of the graph |

number vc (Corollary 3.14). CNP-V and CNP-NDV admit a kernelization with $\mathcal{O}(\text{nd} \cdot (k + x))$ vertices (Theorem 5.14). CNP-V admits a kernelization with less than $(|A|+2\cdot \text{vc})\cdot(k+x+3)$ vertices (Corollary 5.21) and CNP-NDV admits a kernelization with less than $2 \cdot \text{vc}\cdot(k+x+3)$ vertices (Corollary 5.22).

**The parameter $\mathbf{k+x}$ and the parameter $\mathbf{y}$.** We prove that CNP-V and CNP-NDV are fixed-parameter tractable with respect to the parameters $k+x$ and $y$. We put special focus on the parameter $k+x$ by presenting one algorithm for CNP-NDV (Theorem 3.3) and two algorithms for CNP-V (Theorem 3.5 and 3.8), as $k$ and $x$ are explicitly given with the input and $k + x$ is therefore a natural parametrization. Because CNP is a special case of CNP-V, it is somewhat surprising that the fixed-parameter algorithms for CNP-V with respect to $k + x$ and $y$ with running-time $\mathcal{O}(3^{k+x} \cdot (n + m))$ or respectively with running-time $\mathcal{O}(2^y \cdot y^2 \cdot n)$ (Theorem 3.15) are as fast as the (to the best of our knowledge) best currently known fixed-parameter algorithms for CNP with the same paramtrization; the fixed-parameter algorithm with running-time $\mathcal{O}(2^{k+x} \cdot (n + m))$ for CNP-NDV is even faster.

**Results on specific graph classes.** We show that CNP-V and CNP-NDV are solvable within polynomial time on forests (Corollary 2.17). Furthermore, in Theorem 2.28, we show that CNP-V can be solved in linear time, if the biggest non-trivial connected component of the input graph contains at most four vertices. We also present a linear-time algorithm (without correctness proof) for solving CNP-NDV on the same instances as well. CNP-NDV is NP-hard on planar graphs, even if the maximum vertex degree is four (Theorem 2.11) and CNP-NDV is W[1]-hard with respect to the parameter $k + y$ on split graphs, even if there is only one vulnerable vertex (Corollary 4.3). Furthermore, CNP-V is W[1]-hard with respect to the parameter $k$, even on bipartite graphs (Corollary 4.4).

Table 1.2: This table shows with respect to which parameters CNP-V or CNP-NDV are fixed-parameter tractable.

| Parameter | CNP-V | CNP-NDV |
|---|---|---|
| $\lvert A\rvert$ | XP (Cor 3.19) | W[1]-hard (Thm 4.5) |
| $\lvert A\rvert + x$ | FPT (Cor 3.9) | |
| $\lvert A\rvert + x + \Delta$ | | FPT (Thm 4.7) |
| $\lvert V \setminus A\rvert$ | para-NP-hard (Thm 2.12) | FPT (Thm 3.16) |
| $\lvert V \setminus A\rvert + x$ | | poly kernel (Thm 5.6) |
| $\lvert V \setminus A\rvert + \overline{n} + x + \Delta$ | | |
| nd | FPT (Cor 3.13) | FPT (Cor 3.13) |
| $\text{nd} + k + x$ | poly kernel (Thm 5.14) | poly kernel (Thm 5.14) |
| vc | FPT (Cor 3.14) | FPT (Cor 3.14) |
| $\text{vc} + k + x$ | | poly kernel (Cor 5.22) |
| $\lvert A\rvert + \text{vc} + x$ | poly kernel (Cor 5.21) | FPT |
| $\overline{n}$ | para-NP-hard (Thm 2.12) | XP (Cor 3.17) |
| | | W[1]-hard (Thm 4.1) |
| $\overline{n} + k + y$ | FPT (Cor 3.15) | |
| $\overline{n} + x + \Delta$ | FPT (Cor 4.6) | FPT (Thm 4.7) |
| $k$ | W[1]-hard (Cor 4.4) | W[1]-hard (Thm 4.1) |
| | XP (Lem 3.18) | XP (Cor 3.17) |
| $k + x$ | FPT (Cor 3.6, 3.8) | FPT (Cor 3.4) |
| $k + y$ | FPT (Cor 3.15) | W[1]-hard (Cor 4.3) |
| $x$ | para-NP-hard (Thm 2.12) | W[1]-hard (Thm 4.5) |
| | | XP (Cor 4.8) |
| $y$ | FPT (Cor 3.15) | W[1]-hard |
| | No poly kernel [HKKN16] | XP (Cor 3.17) |
| $p$ | FPT (Cor 3.10) | FPT (Cor 3.10) |
| $\Delta$ | para-NP-hard (Thm 2.12) | para-NP-hard (Thm 2.11) |
| diam | para-NP-hard (Lem 2.13) | para-NP-hard (Lem 2.13) |

## 1.5 Some Relevant Graph Problems

In this section, we give an overview over decision problems that we refer to in the rest of the work.

CLIQUE
**Input**: A graph $G = (V, E)$, an integer $\ell \in \mathbb{N}$.
**Question**: Is there a vertex set $C \subseteq V$ of size $\ell$ such that $G[C]$ is a clique?

INDEPENDENT SET
**Input**: A graph $G = (V, E)$ , an integer $\ell \in \mathbb{N}$.
**Question**: Is there a vertex set $I \subseteq V$ of size $\ell$ such that $G[I]$ is an independent set?

VERTEX COVER
**Input**: A graph $G = (V, E)$, an integer $\ell \in \mathbb{N}$.
**Question**: Is there a vertex set $I \subseteq V$ of size $\ell$ such that the graph $G - I$ does not contain an edge?

These three graph-decision problems are classic problems and were shown to be NP-hard by Richard Karp [Kar72]. The problems CLIQUE and INDEPENDENT SET are W[1]-hard with respect to the solution size $\ell$ [CFK$^+$15].

Except for these classic graph decision problems, we will also refer to the following problem several times:

CUTTING AT MOST $k$ VERTICES WITH TERMINAL
**Input**: A graph $G = (V, E)$, a terminal $s \in V$ and integers $k \geq 1, t \geq 0$.
**Question**: Is there a non-empty set $X \subseteq V$ of size at most $X$ such that $s \in X$ such that the neighborhood of $X$ contains at most $t$ vertices?

Fomin et al. [FGK13] showed, that CUTTING AT MOST $k$ VERTICES WITH TERMINAL is NP-hard and W[1]-hard with respect to the parameters $k$ and $t$.

# Chapter 2

# Basic Observations

In the first section of this chapter we introduce parameters that we study in the rest of this work. Furthermore, we show dependencies between the parameters. Because CNP is NP-hard [ACEP09] and CNP is the special case of CNP-V with $A = V$ we already know that CNP-V is NP-hard. A priori, this is not clear for CNP-NDV. In the second section of this chapter we show that CNP-NDV is NP-hard, even if the maximum vertex degree is 4 and the input graph is a planar graph. In the last section of this chapter, we provide some polynomial-time algorithms to solve CNP-V and CNP-NDV in special cases, for example forests.

## 2.1 Interesting Parameters

In this section we present the parameters that we study in this work. These parameters can be categorized in parameters that are based on the structure of the input graph and parameters that are given by the problem definition. In the second subsection, we show that the parameters we consider are not redundant by showing that the parameters are independent or whether there are dependencies.

### 2.1.1 Overview of Parameters

We start to give an overview over parameters that are given by the problem definition. A natural parameter arising from the problem definition is the size of the set of vulnerable vertices $|A|$. We also consider the dual parameter $|V \setminus A|$ that is the number of non-vulnerable vertices. Another parameter that comes with the vulnerable nodes is $\overline{n} := |N(A)|$ which is the size of the neighborhood of $A$. Additionally, three parameters that we consider have already been presented by Hermelin et al. [HKKN16] in their paper about parameterized complexity of CNP: the maximum size of a critical node cut $k$, the number of remaining vulnerable connections $x$, the number of vulnerable connections to delete $y$.

Observe, that in CNP-V and CNP-NDV we define the parameters $x$ and $y$ with regard to vulnerable connections, while in CNP both parameters were only dependent on vertices and edges of the graph, as all vertices are vulnerable. Especially, if a connected graph contains $n$ vertices then unlike in CNP the equation $y = \binom{n}{2} - x$ does not hold in CNP-V and CNP-NDV, unless $A = V$ and $x \leq n(n-1)$. Thus, we define the number $p$ of vulnerable connections of an input graph $G$, $p := |\{\{u, v\} \mid u \in A \text{ and } v \in A \text{ are connected}\}|$. Then we have $y := \max\{0, p - x\}$.

We also consider structural parameters. These parameters include the vertex cover number vc, the diameter diam of the graph and the maximum vertex degree $\Delta$. For the relation $\sim$ on vertices of $V$ with $u \sim v :\iff N[u] = N[v]$ or $N(u) = N(v)$ we define the *neighborhood-diversity* nd of a graph $G$ as the number of equivalence-classes $\text{nd} := |\{[u]_\sim \mid u \in V\}|$. An equivalence-class given by $\sim$ is called a *neighborhood-class.* A neighborhood-class that is an independent set is called a *critical independent set* and a neighborhood-class that is a clique is called a *critical clique.* The parameter neighborhood-diversity nd is a structural parameter, which was introduced by Michael Lampis [Lam12].

Summarizing, all parameters that we consider in this work are the parameters that arise from the problem definition $|A|, |V \setminus A|, \overline{n}, p, k, x,$ and $y$ and the problems that are defined by the structure of the input graph alone $\text{nd}, \text{vc}, \Delta,$ and diam. All parameters are listed in Table 1.1.

## 2.1.2 Dependencies of Parameters

In this subsection we show whether two parameters bounded by each other, or whether they are independent. A similar work has been done by Sorge and Weller [SW13]. For sake of completeness we prove the dependencies between all pairwise combinations of the parameters considered in this work. Let $\lambda_1$ and $\lambda_2$ be two parameters. We call $\lambda_1$ *bounded in* $\lambda_2$, if a computable function $f$ exists such that $\lambda_1 \leq f(\lambda_2)$ in every instance. We call $\lambda_1$ *independent from* $\lambda_2$, if $\lambda_1$ is not bounded in $\lambda_2$ and $\lambda_2$ is not bounded in $\lambda_1$.

In order to show the parameters used in this work are not redundant, we show for every pair of parameters $\lambda_1$ and $\lambda_2$ studied in this work, that if $\lambda_1$ is bounded in $\lambda_2$, then $\lambda_2$ is not bounded in $\lambda_1$. Observe, that every structural parameter $\lambda_1$ is independent from every parameter $\lambda_2$ that is specific from the problem definition, as we can adjust the input graph such that one parameter can be held on the same size while the other is not. However, at the end of the subsection we also consider some instances that are trivially solvable, if parameters have a certain size in comparison to another.

We observe that the parameters $k$ and $x$ can be chosen independently from the input graph. Hence, they are independent from all other parameters. Later, we show that some instances are trivial, if $k$ is bigger than specific parameters.

Recall that we defined $y = \max\{0, p - x\} \leq p$. It follows, that $x + y \geq p$ because $x + y = \max\{x, p\} \geq p$. Furthermore, because every neighbor of $A$ is in a vulnerable connection, $\overline{n} \leq p$.

**Observation 2.1** *1. The parameters $k$ and $x$ are independent from all other parameters presented in Table 1.1.*

*2. The parameters $\overline{n}$ and $y$ are bounded in $p$.*

As we have categorized $k$ and $x$, the remaining parameters that are from the problem definitions are $|A|, |V \setminus A|, \overline{n}, p$ and $y$.

**Observation 2.2** *1. The parameter $p$ is not bounded in $y$.*

2. *The parameter $p$ is not bounded in $\overline{n}$.*

3. *The parameter $y$ is independent from $|A|$, $|V \setminus A|$ and $\overline{n}$.*

4. *The parameter $|V \setminus A|$ is bounded in $\overline{n}$, but the parameter $\overline{n}$ is bounded in $|V \setminus A|$.*

**Proof** *1. and 2.* In order to show that the parameter $p$ is not bounded in $y$ or $\overline{n}$, we give a sequence of instances $I_{n\in\mathbb{N}} = (G, A, k, x)_{n\in\mathbb{N}}$ such that an integer $c$ exists such that $y$ and $\overline{n}$ are at most $c$ in $I_i$ for all $i \in \mathbb{N}$, but $p$ is divergent in the sequence.

Let $G_\ell$ be a clique with $\ell$ vertices and let $A_\ell = V(G_\ell)$ and define the integers $k_\ell := 5, x_\ell := {}^{\ell(\ell-1)}/_2$. Then, in the instance $(G_\ell, A_\ell, k_\ell, x_\ell)$ the value of $\overline{n} = y = 0$ and the value of $p$ is ${}^{\ell(\ell-1)}/_2$ and therefore divergent.

*3. to 4.* Let $G_\ell$ be a path of length $\ell$ and let $A_\ell$ contain exactly one vertex. When $k_\ell = x_\ell = 0$. Then, in the instance $I_\ell$ the value of $|A| = 1, \overline{n} \in \{1, 2\}$ and $p = y = |V \setminus A| = \ell - 1$.

Let $G_\ell$ be a given graph with vulnerable vertices $A_\ell$. Set $x_\ell$ to a number that is greater than the number of vulnerable connections $p$. In this instance it is $y = 0$. Thereby, $|A|, |V \setminus A|$ and $\overline{n}$ are not bounded by $y$, which shows independence of $y$ from the parameters $\overline{n}$ and $|A|$.

Consider a connected graph $G_\ell$ with $\ell$ vertices and the sequence of instances $I_\ell = (G_\ell, V(G_\ell), 0, 0)$. In $I_\ell$ we have $|V \setminus A| = 0$ and $y = {}^{\ell\cdot(\ell-1)}/_2$, which shows independence of $y$ and $|V \setminus A|$.

Because $\overline{n} = |N(A)|$ and $N(A) \subseteq V \setminus A$, we follow $\overline{n} \leq |V \setminus A|$. □

For the parameters that arise from the problem definition, we have to show the following two combinations: $|A|$ with the dual parameter $|V \setminus A|$ and the size of the neighborhood $\overline{n}$. Furthermore, we have not yet analyzed the dependencies between $p$ and $|A|$ respectively $|V \setminus A|$. We leave that case for the end of the section, because we can find instances in which $|A|$ or $|V \setminus A|$ are not dependent on $p$, but these can be reduced to instances in which $|A| \leq p$ and $|V \setminus A| \leq p$.

**Observation 2.3** *The parameter $|A|$ is independent from $|V \setminus A|$ and from $\overline{n}$.*

**Proof** Let $G_\ell$ be a graph with $\ell$ vertices. We consider the two sequences of instances $I_\ell = (G_\ell, V(G_\ell), k, x)$ and $J_\ell = (G_\ell, \emptyset, k, x)$ for fixed $k, x \in \mathbb{N}$. In $I_\ell$ we have $|A| = \ell$ but $|V \setminus A| = 0$ and in $J_\ell$ we have $|A| = 0$ but $|V \setminus A| = \ell$. Thus, $|A|$ and $|V \setminus A|$ are independent from each other.

To proof the independence of $\overline{n}$ from $|A|$ consider the graph $G_q$ which is a $q$-star with center $u$. According to this, we consider the two sequences of instances $I_q = (G_q, \{u\}, k, x)$ and $J_\ell = (G_q, V(G_q) \setminus \{u\}, k, x)$ for fixed $k, x \in \mathbb{N}$. Then, in $I_q$ we have $|A| = 1, \overline{n} = q$ and in $J_q$ we have $|A| = q, \overline{n} = 1$. Thus, $\overline{n}$ is independent from $|A|$. □

Next, we consider dependencies between structural parameters.

**Observation 2.4** 1. *The parameter $\Delta$ is independent from the parameters neighborhood-diversity* nd*, vertex cover number* vc *and diameter of the input graph* diam*.*

2. *The parameter* nd *is not bounded in* diam*, but* diam $\leq$ nd*.*

3. *The parameter* vc *is not bounded in* nd*, but* $\mathrm{nd} \in \mathcal{O}(2^{\mathrm{vc}})$.

4. *The parameter* vc *is not bounded in* diam*, but* $\mathrm{diam} \leq 2 \cdot \mathrm{vc} + 1$.

**Proof** In this proof we only give a sequence of graphs as example that are defined by giving the instance $G_\ell$ for an integer $\ell \in \mathbb{N}$. This is sufficient, as a sequence of graphs $G_\ell$ can be considered as sequence of instances $(G_\ell, V(G_\ell), k, x)$ for a fixed $k, x \in \mathbb{N}$.

*The parameters* $\mathrm{nd}, \mathrm{vc}$ *and* diam *are not bounded in* $\Delta$*:* Considering with $T_\ell$ a tree of depth $\ell$ in which every vertex of height smaller than $\ell$ has exactly two children and vertices with height $\ell$ are leaves. Then, in $T_\ell$ the maximum node degree is $\Delta = 3$, however $\mathrm{diam} = 2 \cdot \ell$ and $\mathrm{vc} \geq 2^{\ell-2}$. Every vertex in $T_\ell$ is in its own neighborhood-class, only the leaves are together with the sibling-leaf. It follows that $\mathrm{nd} = |V| - \frac{1}{2} \cdot 2^\ell = (2^{\ell+1} - 1) - 2^{\ell-1} = 3 \cdot 2^{\ell-1} - 1$.

*The parameters* $\mathrm{nd}, \mathrm{vc}$ *and* $\Delta$ *are not bounded in* diam*:* Consider the split graph $G_\ell$ that contains $2 \cdot \ell$ vertices that are a clique of size $\ell$ for which every vertex has a personal one-degree neighbor. Then in $G_\ell$ the diameter is $\mathrm{diam} = 3$, but $\Delta = \ell$, $\mathrm{nd} = 2 \cdot \ell$ and $\mathrm{vc} = \ell$.

*The parameters* vc *and* $\Delta$ *are not bounded in* nd*:* In a clique of size $\ell$ the neighborhood-diversity is $\mathrm{nd} = 1$, but $\mathrm{vc} = \Delta = \ell - 1$.

*The parameter* $\Delta$ *is not bounded in* vc*:* In a $q$-star the vertex cover number is $\mathrm{vc} = 1$, but $\Delta = q$.

diam *is bounded in* nd*:* Let a graph $G$ be a connected graph with neighborhood-diversity nd. Without loss of generality, the vertices $w_1, w_2 \in V$ have the maximum distance within the graph. Let $P$ be a shortest path from $w_1$ to $w_2$ in $G$ that also contains the vertices $v_1, \ldots, v_t$. Recall that we defined the neighborhood-diversity over a relation $\sim$ on vertices which on vertices $u$ and $v$ is defined by $u \sim v \iff N(u) = N(v)$ or $N[u] = N[v]$. It is $v_\ell \notin [w_1]_\sim$, because otherwise $v_{\ell+1} = w_1$ or $v_{\ell+1}$ is neighbor of $w_1$ and $w_1, v_{\ell+1}, \ldots, v_t, w_2$ or $w_1, v_{\ell+2}, \ldots, v_t, w_2$ is a path from $w_1$ to $w_2$ in $G$. This is a contradiction to the fact that $P$ is a shortest path. With the same argument we follow that $v_\ell \notin [w_1]_\sim \cup \bigcup_{i=1}^{\ell-1} [v_i]_\sim$. Hence, the vertices $w_1, v_1, \ldots, v_t, w_2$ are all in unique neighborhood-classes and $\mathrm{diam} \leq \mathrm{nd}$.

nd *is bounded in* vc*:* Let $G$ be a graph and vc the vertex cover number of $G$. Let $u_1, \ldots, u_{\mathrm{nd}}$ be representative vertices of all unique neighborhood-classes of $G$. For the induced subgraph $G' := G[\{u_1, \ldots, u_{\mathrm{nd}}\}]$ let $\overline{v}$ be the vertex cover number. Obviously, $\overline{v} \leq \mathrm{vc}$. Without loss of generality, let $\{u_1, \ldots, u_{\overline{v}}\}$ be a vertex cover of $G'$ and thus removing $u_1, \ldots, u_{\overline{v}}$ from $G'$ removes all edges of $G'$. This implicates that there are no edges in the induced subgraph $G[\{u_{\overline{v}+1}, \ldots, u_{\mathrm{nd}}\}]$. The nodes $u_{\overline{v}+1}, \ldots, u_{\mathrm{nd}}$ are from different neighborhood-classes in $G$. Thus, $u_p$ and $u_q$ for $p, q \in \{\overline{v}+1, \ldots, \mathrm{nd}\}$ are not neighbors of the same vertices of $u_1, \ldots, u_{\overline{v}}$ and because $G[\{u_{\overline{v}+1}, \ldots, u_{\mathrm{nd}}\}]$ does not contain edges, $u_p$ and $u_q$ are not neighbors in $G$. Thus, as there are $2^{\overline{v}}$ options to be adjacent to a vertex of $u_1, \ldots, u_{\overline{v}}$. Thus, $\mathrm{nd} \leq \overline{v} + 2^{\overline{v}} \leq \mathrm{vc} + 2^{\mathrm{vc}} \in \mathcal{O}(2^{\mathrm{vc}})$.

diam *is bounded in* vc*:* Let $G = (V, E)$ be a graph and vc the vertex cover number of $G$. Let $Z$ be a vertex cover of $G$ and let $v_1, \ldots, v_{\mathrm{diam}}$ be a shortest path in $G$. Because $Z$ is a vertex cover, $G - Z$ does not contain edges. It follows, that for every $i \in [1, \mathrm{diam} - 1]$ at most one vertex of $v_i$ and $v_{i+1}$ is in $V \setminus Z$. It follows, that $\mathrm{diam} \leq 2 \cdot \mathrm{vc} + 1$. □

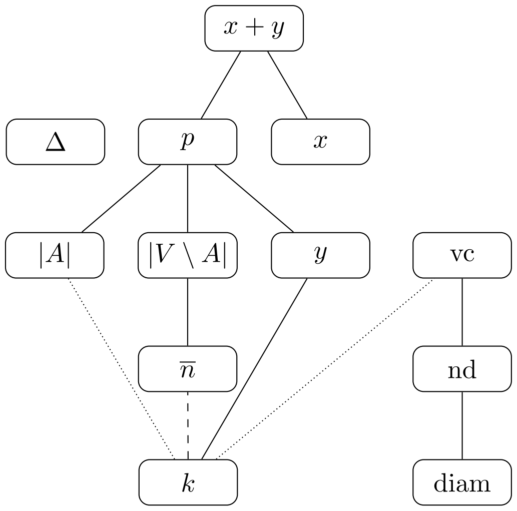


Figure 2.1: The hierarchy of dependencies of the parameters, where the higher listed parameters are greater. Dotted lines only in CNP-V. Dashed lines only in CNP-NDV.

As we have mentioned before, structural parameters and problem parameters are independent from each other. Nevertheless, some instances are trivial if two parameters have a certain size in relation to each other. In the rest of the subsection we focus on these cases. In Figure 2.1 we can see the hierarchies that account on both problems in solid lines, while the dotted lines are only for instances of CNP-V and dashed lines only for instances of CNP-NDV.

The first example is that, if in an instance of CNP-V or CNP-NDV the parameter $k$ is greater than $y$, we add $y$ arbitrary non-vulnerable vertices of non-trivial connected components to a critical node cut.

**Reduction Rule 2.5** *Return yes, if the parameter $y$ is at most $k$ in an instance $(G, A, k, x)$ of* CNP-V *or* CNP-NDV*.*

Notice, that in the explanation before, we left a case out. If an instance of CNP-NDV contains less than $y < k$ non-vulnerable vertices in non-trivial components, we can not add $y$ non-vulnerable vertices to a critical node cut. Instead, we add all non-vulnerable vertices to a potential critical node cut and then all vulnerable connections are within $G[A]$. The same is already true if $|V \setminus A| \leq k$ and does not need $|V \setminus A| < y < k$. As an instance of CNP-NDV can be a no-instance when $|V \setminus A| < y < k$, it is important to execute the following before Reduction Rule 2.5.

**Reduction Rule 2.6** *If in an instance $(G, A, k, x)$ of* CNP-NDV *the parameter $|V \setminus A|$ is at most $k$, then return yes if and only if the connectivity of $G[A]$ is at most $x$.*

Reduction Rule 2.6 can be specified even more, as it is sufficient to add the neighborhood of $A$ to a critical node cut. Thus, we can follow the following.

**Reduction Rule 2.7** *If in an instance $(G, A, k, x)$ of* CNP-NDV *the parameter $\overline{n}$ is at most $k$, then return yes if and only if the connectivity of $G[A]$ is at most $x$.*

It is easy to observe that an instance $(G, A, k, x)$ is equivalent to an instance $(G', A', k, x)$ in which $G'$ is the graph $G$ after all trivial components are removed from $G$. Thus, every vulnerable vertex is in at least one vulnerable connection and because every non-vulnerable is in a connected component with at least one vulnerable vertex.

**Reduction Rule 2.8** *Remove all trivial connected components from an instance* $(G, A, k, x)$ *of* CNP-V *or* CNP-NDV*. Afterwards the parameter* $p$ *is greater than* $|A|$ *or* $|V \setminus A|$.

An instance of CNP-V is a trivial yes-instance, if $|A| \leq k$, as we can add all vulnerable vertices to a critical node cut and have no more vulnerable connections.

**Reduction Rule 2.9** *Return yes, if in instance* $(G, A, k, x)$ *of* CNP-V *the parameter* $|A|$ *is at most* $k$.

If an instance of CNP-V has a vertex cover number that is at most $k$, the vertex cover is a critical node cut. We can directly return yes, even though we maybe do not find the vertex cover in polynomial time.

**Reduction Rule 2.10** *Return yes, if in an instance* $(G, A, k, x)$ *of* CNP-V *the parameter* vc *is at most* $k$.

## 2.2 NP-hardness

In this section we show that both problems CNP-V and CNP-NDV are NP-hard. We give even stronger results by showing that CNP-NDV is NP-hard on planar graphs, even if the maximum degree is 4. Because CNP-V generalizes Vertex Cover, CNP-V is NP-hard, even if $|V \setminus A| = \overline{n} = x = 0$ and $\Delta = 3$.

To show, that CNP-NDV is NP-hard, even if the maximum degree is 4, we reduce from Independent Set. Recall that in Independent Set, a graph and an integer $\ell$ are given as input and the question is whether there exists a subset $X$ of $\ell$ vertices, which are pairwise non adjacent. Garey et al. [GJS76] showed that Independent Set is NP-hard on planar graphs, even if the maximum degree is 3.

**Theorem 2.11** CNP-NDV *is* NP-*hard on planar graphs, even if* $\Delta = 4$.

**Proof** *Construction:* Let $(G, \ell)$ 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: 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 $k := n_G - \ell$ and $x := \ell$. 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.

*Correctess:* Let $(G, \ell)$ be a yes-instance of Independent Set. Then, there exists an independent set $I \subseteq V(G)$ of size $\ell$. We show that the vertex set $V \setminus I$ is a critical node cut: The size of $V \setminus I$ is $n_G - \ell$ 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 $\ell$ 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 connectivity of $G' - (V \setminus I)$ is $\ell$. 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 $n_G - \ell$. The graph $G' - C$ contains $\ell$ vertices that are not vulnerable and by the definition, 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, $(G, \ell)$ is a yes-instance of INDEPENDENT SET. $\square$

It has been shown by Arulselvan et al. [ACEP09] that the CNP is NP-complete. Since CNP is the special case of CNP-V with $A = V$, we already know, that CNP-V is NP-hard. However, we observe an even stronger result by reducing from VERTEX COVER.

Recall, that in VERTEX COVER, a graph $G$ and an integer $k$ are given as input and the question is, whether by deleting $k$ vertices of $G$ there remains no edge in $G$. VERTEX COVER is one of Richard Karp's famous 21 combinatorial NP-complete problems and Karp [Kar72] showed, that VERTEX COVER is NP-hard, even if $\Delta = 3$. Since VERTEX COVER is a special case of CNP-V with $A = V$ and $x = 0$, we can directly follow that CNP-NDV is NP-hard, even if $|V \setminus A| + x + \overline{n} + \Delta$ is constant.

**Theorem 2.12** CNP-V *is* NP*-hard, even if* $|V \setminus A| = x = \overline{n} = 0$ *and* $\Delta = 3$.

So far we saw that CNP-V and CNP-NDV are NP-hard problems. Consider the following: An instance $(G, A, k, x)$ of CNP-V or CNP-NDV is equivalent to an instance $(G', A, k + 1, x)$ where $G'$ is the graph $G$ where a universal vertex. Then, $G'$ has a diameter of two. Thus, both problems are NP-hard, even if diam $= 2$.

**Lemma 2.13** CNP-V *and* CNP-NDV *are NP-hard, even if* diam $= 2$.

## 2.3 Polynomial-time Algorithms

In this section we present some algorithms that run in polynomial time. First, we show that we can compute the connectivity of a graph in $\mathcal{O}(n + m)$ time. This information will be used in the rest of the work because the computation of the connectivity is a subroutine of almost every other algorithm.

Furthermore, we show that we can solve CNP-V and CNP-NDV in polynomial time on forests and that we can solve CNP-V in linear time on graphs, where the maximum size of a connected component is 4.

**Lemma 2.14** *The connectivity of a graph $G$ can be computed in $\mathcal{O}(n+m)$ time.*

**Proof** In $\mathcal{O}(n + m)$ time all connected components can be computed with breadth-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 connectivity of the connected component $K$ is

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

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

**Trees and Forests** In the following, we prove that CNP-V and CNP-NDV can be solved in polynomial time on trees and later we generalize this result to forests. This algorithm shows us that at least some non-trivial algorithms exist that run in polynomial time. With the help of dynamic programming, Di Summa et al. [DSGL11] proved that CNP can be solved in polynomial time on trees. Our algorithm uses a similar technique but we need to remember the number of vulnerable and non-vulnerable vertices as well.

**Theorem 2.15** *CNP-V and CNP-NDV can be solved in $\mathcal{O}(k \cdot n^2)$ on trees.*

**Proof** *Definition of the table:* We provide a dynamic programming algorithm. Let $T := (V, E)$ be a tree with root $w$, which can be an arbitrary vertex $w \in V$. Furthermore, let $(T, A, k, x)$ be an instance of CNP-V or CNP-NDV. In this algorithm, we define two four-dimensional tables, $F$ and $H_i$ where $i$ is an integer. A table entry $F[v, \overline{k}, d, b]$ stores the smallest number of vulnerable connections in the subtree $st(T, v)$ after exactly $\overline{k}$ vertices are removed under the condition that $b$ vertices (including $v$) are connected to $v$ of which $d$ are vulnerable. That means that if $b = 0$, the vertex $v$ is removed. Recall, that $st(T, v) = T$ if $v = w$ and otherwise $st(T, v)$ is the connected component of $v$ after the parent of $v$ is removed from $T$. Let $v$ be a child with children $u_1, \dots, u_\ell$. For $i \in [1, \ell]$ we define with $T_{v,i}$ the subtree of $T$ that contains the vertex $v$ and the vertices of the subtrees $st(T, u_i), \dots, st(T, u_\ell)$. A table entry $H_i[v, \overline{k}, d, b]$ stores the smallest number of vulnerable connections in $T_{v,i}$ after exactly $\overline{k}$ vertices are removed under the condition that $b$ vertices (including $v$) are connected to $v$ of which $d$ are vulnerable. If it is impossible that with the deletion of $k$ vertices from $st(T, v)$ or respectively $T_{v,i}$ exactly $b$ vertices are connected to $v$ of which $d$ are vulnerable, then we define $F[v, k, d, b] = \infty$ or respectively $H_i[v, k, d, b] = \infty$. At the end the algorithm returns yes if and only if there exist $d \in [0, |A|]$, $b \in [0, |V|]$ such that $F[w, k, d, b] \leq x$.

*Algorithm:* Let $T = (V, E)$ be a tree with root $w$ and let $(T, A, k, x)$ be an instance of CNP-V or CNP-NDV. Let $v$ be a vertex with children $u_1, \dots, u_\ell$. For the cases $d > b$ and $d < 0$ and $b < 0$ and $k < 0$, we do not define $F[v, k, d, b]$ or $H_i[v, k, d, b]$. For a leaf $v$, we define $F$ if no vertex can be removed as $k = 0$

$$F[v,0,0,0] = \infty \tag{2.1}$$

$$F[v,0,0,1] = \begin{cases} 0 & v \notin A \\ \infty & v \in A \end{cases} \tag{2.2}$$

$$F[v,0,1,1] = \begin{cases} \infty & v \notin A \\ 0 & v \in A \end{cases} \tag{2.3}$$

For a leaf $v$, we define $F$ if one vertex can be removed as $k = 1$

$$F[v,1,0,0] \overset{\text{CNP-V}}{=} 0 \tag{2.4}$$

$$F[v,1,0,0] \overset{\text{CNP-NDV}}{=} \begin{cases} 0 & v \notin A \\ \infty & v \in A \end{cases} \tag{2.5}$$

$$F[v,1,0,1] = \infty \tag{2.6}$$

$$F[v,1,1,1] = \infty \tag{2.7}$$

$$F[v,k,d,b] = \infty \quad \text{if } k > 1 \text{ or } d > 1 \text{ or } b > 1 \tag{2.8}$$

Let $v$ be a vertex of $T$ with children $u_1, \dots, u_\ell$. We define the abbreviation

$$F[v,k,*,*] := \min\{F[v,k,d,b] \mid 0 \le d < b \le n_{st(T,v)}\}.$$

First, we define $H_i$ for the case that $b = 0$, which means that $v$ is removed. Observe that $v$ has exactly $\ell$ children. For the entire algorithm we define for every integer $z \in \mathbb{N}$ that $\infty + z = z + \infty = \infty$. For all $\overline{k} \in [0,k]$ we define:

$$H_\ell[v,\overline{k},0,0] \overset{\text{CNP-V}}{=} \begin{cases} F[u_\ell,\overline{k}-1,*,*] & \overline{k} > 0 \\ \infty & \overline{k} = 0 \end{cases} \tag{2.9}$$

$$H_\ell[v,\overline{k},0,0] \overset{\text{CNP-NDV}}{=} \begin{cases} F[u_\ell,\overline{k}-1,*,*] & v \notin A, \overline{k} > 0 \\ \infty & v \notin A, \overline{k} = 0 \\ \infty & v \in A \end{cases} \tag{2.10}$$

Now we define $H_i$, where $i \in [1, \ell-1]$. Thus, we have already regarded the subtree $T_{v,i+1}$ and we have to combine the solution to have a solution for $T_{v,i}$.

$$H_i[v,\overline{k},0,0] \overset{\text{CNP-V}}{=} \begin{cases} \min\limits_{k' \in [0,\overline{k}]} \big( \; H_{i+1}[v,k',0,0] \\ \qquad + F[u_i, \overline{k}-k', *, *] \; \big) \end{cases} \tag{2.11}$$

$$H_i[v,\overline{k},0,0] \overset{\text{CNP-NDV}}{=} \begin{cases} \min\limits_{k' \in [0,\overline{k}]} \big( \; H_{i+1}[v,k',0,0] & \\ \qquad + F[u_i, \overline{k}-k', *, *] \; \big) & v \notin A \\ \infty & v \in A \end{cases} \tag{2.12}$$

Once we define $H_1$, we can migrate the information to $F$. So we set the value of $F[v,\overline{k},0,0] = H_1[v,\overline{k},0,0]$. So now we define the case in which $v$ is not removed, that means $b > 0$.

$$H_\ell[v,\overline{k},d,b] = \begin{cases} F[u_\ell,\overline{k},d,b-1] + d & v \notin A \\ F[u_\ell,\overline{k},d-1,b-1] + b - 1 & v \in A, d \neq 0 \\ \infty & v \in A, d = 0 \end{cases} \tag{2.13}$$

$$H_i[v,\overline{k},d,b] = \begin{cases} \min\limits_{\substack{k' \in [0,\overline{k}] \\ d' \in [0,d] \\ b' \in [0,b]}} \big( \; H_{i+1}[v,k',d',b'] \\ \quad + F[u_i, \overline{k}-k', d-d', b-b'] \\ \quad + d' \cdot (b-b') + (d-d') \cdot (b'-d') \; \big) \end{cases} \tag{2.14}$$

Jut like in the other case, we define $F[v,\overline{k},d,b] = H_1[v,\overline{k},d,b]$.

If $d$ and $b$ exist such that $F[w,k,d,b] \le x$, then we return yes. Otherwise return no.

*Correctness:* The correctness of the algorithm follows by easy observation of all cases. We only observe the correctness of Equation 2.13 and 2.14.

We start with Equation 2.13. So if $b > 0$, then the vertex $v$ is not removed from $T_{v,\ell}$. Thus, we have the vertex $v$ that is connected to $v$. It follows, that in the table entry $H_\ell[v, k, d, b]$ we can copy the best solution in the subtree $st(T, v)$, where $b - 1$ vertices are connected to $v$, of which $d$ are vulnerable, if $v$ is non-vulnerable and $d-1$ otherwise. But then $v$ is in new vulnerable connections with all $d$ vulnerable vertices, if $v$ is non-vulnerable. Otherwise, if $v$ is vulnerable, the vertex $v$ is in vulnerable connections with all $b - 1$ vertices. These vulnerable connections have to be added to the vulnerable connections within $st(T, v)$. If $b > 0$ and $v$ is vulnerable at least one vulnerable vertex is connected to $v$. Hence, if $v$ is vulnerable it is $H_\ell[v, \overline{k}, 0, b] = \infty$.

Now we consider Equation 2.14. Because $b > 0$, the vertex $v$ is not removed from $T_{v,\ell}$. Thus, for every selection of $k' \in [0, \overline{k}], d' \in [0, \overline{d}], b' \in [0, \overline{b}]$ the subtree $T_{v,i+1}$ contains $b'$ vertices that are connected to $v$, of which $d'$ are vulnerable and the subtree $st(T, u_i)$ contains $b-b'$ vertices that are connected to $u_i$ the child of $v$, of which $d - d'$ are vulnerable. It follows, that new vulnerable connections are $d' \cdot (d - d')$ vulnerable connections between two vulnerable vertices, $d' \cdot ((b - b') - (d - d'))$ vulnerable connections between vulnerable vertices of $T_{v,i+1}$ and non-vulnerable vertices of $st(T, u_i)$ and $(b' - d') \cdot (d - d')$ vulnerable connections between non-vulnerable vertices of $T_{v,i+1}$ and vulnerable vertices of $st(T, u_i)$. Altogether the number of new vulnerable connections adds up to $d' \cdot (b - b') + (b' - d') \cdot (d - d')$ new vulnerable connections, that have to be added.

*Running time:* The table $F$ and $H_i$ contains $(n + 1)^3 \cdot (k + 1)$ entries and $i \in [1, \Delta]$. Every entry is computed in $\mathcal{O}(k \cdot n^2)$ time, when the look-up of a table cell is considered to take constant time. Hence, the algorithm has polynomial running time. □

In the next lemma, we prove that if we have optimal solutions for connected components of a graph as input, we can find a solution for the entire graph within polynomial time.

**Lemma 2.16** *Let $(G, A, k, x)$ be an instance of* CNP-V *or* CNP-NDV *and we consider with let $C_1, \dots, C_\ell$ be the connected components of $G$. If a solution for the instances $(C_i, A \cap C_i, k', x')$ for all $0 \le k' \le k$, $0 \le x' \le x$ and $i \in \{1, \dots, \ell\}$ are given, we can compute a solution for $(G, A, k, x)$ in polynomial time.*

**Proof** *Algorithm:* We define a direct programming algorithm. We define values of the table entries $F[i, k']$ and $H[i, k']$ by

$$F[i, k'] \quad = \quad \min\{x \in \mathbb{N} \mid (C_i, A \cap C_i, k', x) \text{ is a yes-instance}\} \tag{2.15}$$

$$H[1, k'] \quad = \quad F[1, k'] \tag{2.16}$$

$$H[i + 1, k'] \quad = \quad \min_{0 \le k'' \le k'} (\; G[i, k''] + F[i + 1, k'' - k'] \;) \tag{2.17}$$

We return yes, if and only if $H[\ell, k] \le x$.

*Correctness:* The correctness follows by observation of the three cases.

*Running time:* By the assumption, for all $i \in [1, \ell]$, and $k' \in [0, k]$ we can compute $F[i, k']$ in polynomial time. The table $F$ has $\ell \cdot (k + 1)$ entries. Thus, all entries of $F$ can be computed in polynomial time.

We can compute $H[i, k']$ for $i > 1$ and $k' \in [1, k]$ by picking $2 \cdot (k' + 1)$ table-entries and adding two. Thus, we can compute $H[i, k']$ in $\mathcal{O}(k')$ time.

Thus, for every $i \in [1, \ell]$, we can compute all $H[i, k']$ with $k' \in [0, k]$ in $\mathcal{O}\left(\binom{k}{2}\right) = \mathcal{O}(k^2)$ time. Altogether, we can compute all entries of $H$ in $\mathcal{O}(\ell \cdot k^2)$ time. □

We proved in Theorem 2.15 that we can solve CNP-V and CNP-NDV in polynomial time on trees. Lemma 2.16 thus now implies the following.

**Corollary 2.17** CNP-V *and* CNP-NDV *can be solved in polynomial time on forests.*

**Small connected components** Let $c$ be a constant. Observe, that we can solve CNP-V and CNP-NDV on a connected component $K$ with $c$ vertices within constant time, by checking whether one of the $2^c$ subsets of vertices of $K$ is a critical node cut for $(G[K], A \cap K, k, x)$ for any integers $k, x$. Observe that $2^c$ is a constant as $c$ is a constant. With Lemma 2.16 the following is a direct result.

**Corollary 2.18** *Let $c$ be a constant integer.* CNP-V *and* CNP-NDV *can be solved in polynomial time on graphs in which the biggest connected component contains at most $c$ vertices.*

The rest of the section we improve the result of Corollary 2.18 by providing a linear-time algorithm that computes a critical node cut for an instance of CNP-V, if the biggest non-trivial connected component of the input graph has size at most 4. We can use this algorithm later in the thesis whenever we have a set of vertices $C$ such that $G - C$ contains only connected components with at most four vertices. After the correctness proof of this algorithm, we shortly present an adjustment for instances of CNP-NDV without a proof.

For the rest of the section, we only consider graphs, in which every connected component contains at most four vertices and at least one vulnerable vertex. Notice, that we can generalize the algorithm easily to graphs in which the largest non-trivial connected component contains at most four vertices by removing all connected components with more than four vertices.

For this algorithm, we first observe which structure a connected component of size at most four can have. That includes how the underlying graph looks like and we also need to specify, which vertices are vulnerable. There are nine options for connected underlying graphs with at most four vertices $\{v_1, \ldots, v_4\}$. Furthermore, there are 15 options such that the connected component contains at least one vulnerable vertex. Let $b \in [1, 15]$ be a number with binary representation $b = b_4 b_3 b_2 b_1$. We define $A_b := \{v_{b_i} \mid i \in [1, 4], b_i = 1\}$. Notice, that $A_b$ is a subset of $\{v_1, \ldots, v_4\}$.

In the following we define the structures $S_{a,b}$ where $a \in [1, 9]$ defines to which underlying graph a connected component is isomorphic to and $b \in [1, 15]$ implicates that the vertices of the set $A_b$ are vulnerable.

1. A $P_4$ with edges $E(P_4) := \{\{v_i, v_{i+1}\} \mid i \in [1, 3]\}$. We call this structure $S_{1,b}$.

2. The connected component with four vertices $\{v_1, \dots, v_4\}$ and with the set of edges $\{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}\}$. We call this structure $S_{2,b}$. The underlying graph is called *claw*.

3. The connected component with four vertices $\{v_1, \dots, v_4\}$ and with the set of edges $\{\{v_1, v_3\}\} \cup E(P_4)$. We call this structure $S_{3,b}$. The underlying graph is called *paw*.

4. A $K_4$. We call this structure $S_{4,b}$.

5. The connected component with four vertices $\{v_1, \dots, v_4\}$ and with the set of edges $\{\{v_1, v_3\}, \{v_1, v_4\}\} \cup E(P_4)$. We call this structure $S_{5,b}$. The underlying graph is called *diamond*.

6. A $C_4$. We call this structure $S_{6,b}$.

7. An isolated $P_3$. We call this structure $S_{7,b}$, where $b \in [1, 7]$.

8. An isolated $C_3$. We call this structure $S_{8,b}$, where $b \in [1, 7]$. The underlying graph is called *triangle*.

9. An isolated $P_2$. We call this structure $S_{9,b}$, where $b \in [1, 3]$.

In the rest of the section, we say that a connected component $K$ of an input-graph $G$ of an instance of CNP-V or CNP-NDV is *isomorphic* to a structure $S_{a,b}$, if a bijection $f : \{v_1, \dots, v_4\} \to \{v_1, \dots, v_4\}$ exists such that $\{u, v\}$ is an edge of $G[K]$, if and only if $\{f(u), f(v)\}$ is an edge of $S_{a,b}$ and a vertex $v$ of $K$ is vulnerable if and only if $f(v)$ is vulnerable in $S_{a,b}$.

Observe that some structures, for example $S_{1,1}$ and $S_{1,8}$, are equal up to isomorphism since in both cases exactly one vertex is vulnerable and this vertex has degree one.

Also, we define *removal vectors* for every structure. A removal vector is a tuple of integers. For a connected component $K$, the value on position $i$ of the removal vector of $K$ implicates how many vulnerable connections can be removed from $K$, when $i$ vertices are removed from $K$ and the $i$ vertices that are removed are or respectively can be extended to the best option for removing two vertices from $K$. We only define removal vectors as tuples of size two and consider two removes from $K$. The reason is that the only connected component of size in which we can not remove all vulnerable connections by removing two vertices is an isolated $K_4$ with at least three vulnerable vertices.

To give an example for a removal vector: A connected component that is isomorphic to $S_{3,12}$ is a paw in which the vertex with degree 1 and the vertex with degree 3 are vulnerable. $S_{3,12}$ has a removal vector of $(5, 5)$. This is because with the first vertex that can be removed we remove the degree three vertex which removes five vulnerable connections and the remaining vertices are in trivial components. Tables 2.1 and 2.2 depicts all structures with their removal vector.

Now we explain why it is important that we define removal vectors in the matter that two vertices are removed from a connected component. In that definition for a connected component $K$ the first entry of the removal vector is not automatically $|K| - 1$. For example consider complete bipartite graph $K_{2,3}$ with vertex bipartition $(P, Q)$ and $|Q| = 3$ and we consider an instance $I := (K_{2,3}, A := Q, k := 2, x := 0)$ of CNP-V. $I$ is a yes-instance, as $P$ is a critical node cut. However, if we ask how many vulnerable connections can be removed, if one vertex is added to a critical node cut, the answer is four. This creates a set that can not be extended to the best option for removing two vertices, which would be $P$. Notice, that we had to consider a connected component with five vertices to give an example. Thus, if we compute all removal vectors of connected components of size at most four we can observe the following.

**Observation 2.19** *In an instance of* CNP-V *every connected component $K$ has a removal vector $(i, j)$ with $j \geq i$ and $i = |K| - 1$.*

We continue with a small Lemma to prove that for a given removal vector, we can compute all connected components of a graph with that removal vector in linear time.

**Lemma 2.20** *Let $i, j \in [1, 6]$ be two integers. In a graph $G$ in which the biggest connected component has at most four vertices, we can find all connected components of $G$ with removal vector $(i, j)$ in $\mathcal{O}(n)$ time.*

**Proof** We iterate over all vertices to check whether a connected component has removal vector of $(i, j)$. For every vertex we can compute its connected component in constant time, as every connected component has at most four vertices. As there are less than $9 \cdot 15$ (which is a constant amount) possibilities for the structure of connected components, we can find the removal vector of an instance in constant time. □

Now, we come to the first reduction rule that includes removal vectors. We first consider the case that all vulnerable connections are removed by removing one vertex of the connected component, such as it is the case in $S_{3,12}$ that we gave as example above. Observe that then the second integer in the removal vector is the same as the first.

**Reduction Rule 2.21** *Let $i \in [1, 6]$ be an integer. If $k \geq 1$ and $G$ contains a connected component $K$ that has a removal vector of $(i, i)$ but $G$ does not contain a connected component that has a removal vector of $(j, t)$ with $j > i$ or $t - j \geq i$, then remove $K$ from $G$ and decrease $k$ by one.*

**Lemma 2.22** *Reduction Rule 2.21 is safe and for a fixed $i \in [1, 6]$ can be applied exhaustively in $\mathcal{O}(n)$ time.*

**Proof** *Safeness:* Let $(G, A, k, x)$ be an instance of CNP-V for which $C$ is a critical node cut and let $G'$ be the graph after Reduction Rule 2.21 is applied. Let $K$ be the connected component described in Reduction Rule 2.21 and let $v$ be a vertex which, when removed, removes all vulnerable connections from $K$. We show that $(G, A, k, x)$ is a yes-instance of CNP-V if and only if the instance $(G', A \setminus \{v\}, k - 1, x)$ is a yes-instance of CNP-V.

If $v \in C$ we are done. Assume that $v \notin C$. By the requirements to $G$, we know that $G$ does not contains a connected component in which the first or second vertex that can be removed, removes at most $i$ vulnerable connections from $G$. Let $w$ be a vertex of $C$. We define $C' := (C \setminus \{w\}) \cup \{v\}$. It follows, that the connectivity of $G - C'$ is as most as high as the connectivity

Table 2.1: Removal vectors of Structures $S_{a,b}$ with $a < 4$. Black vertices are vulnerable.

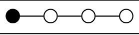
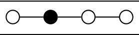
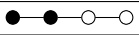
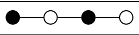
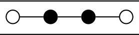
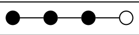
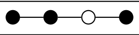
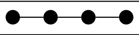
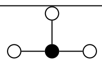
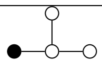
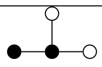
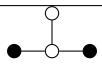
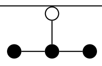
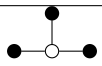
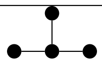
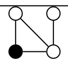
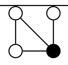
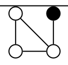
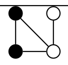
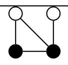
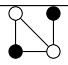
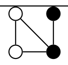
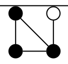
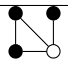
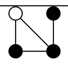
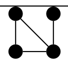
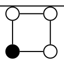
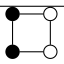
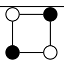
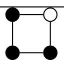
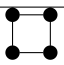
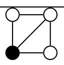
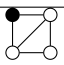
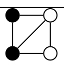
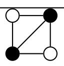
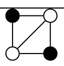
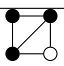
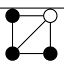
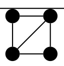
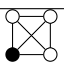
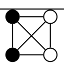
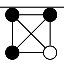
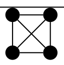
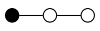
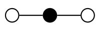
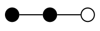
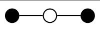
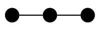
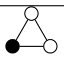
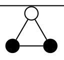
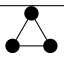
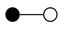
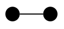
| Component | Name | removal vector | |
|---|---|---|---|
| | | CNP-V | CNP-NDV |
|  | $S_{1,1} \cong S_{1,8}$ | $(3,3)$ | $(3,3)$ |
|  | $S_{1,2} \cong S_{1,4}$ | $(3,3)$ | $(2,3)$ |
|  | $S_{1,3} \cong S_{1,12}$ | $(5,5)$ | $(4,4)$ |
|  | $S_{1,5} \cong S_{1,10}$ | $(3,4)$ | $(3,4)$ |
|  | $S_{1,6}$ | $(4,5)$ | $(2,4)$ |
|  | $S_{1,7} \cong S_{1,14}$ | $(5,6)$ | $(3,-)$ |
|  | $S_{1,11} \cong S_{1,14}$ | $(5,6)$ | $(5,-)$ |
|  | $S_{1,15}$ | $(5,6)$ | $(-,-)$ |
|  | $S_{2,1}$ | $(3,3)$ | $(1,2)$ |
|  | $S_{2,2} \cong S_{2,4} \cong S_{2,8}$ | $(3,3)$ | $(3,3)$ |
|  | $S_{2,3} \cong S_{2,5} \cong S_{2,9}$ | $(5,5)$ | $(2,4)$ |
|  | $S_{2,6} \cong S_{2,10} \cong S_{2,12}$ | $(5,5)$ | $(5,5)$ |
|  | $S_{2,7} \cong S_{2,11} \cong S_{2,13}$ | $(6,6)$ | $(3,-)$ |
|  | $S_{2,14}$ | $(6,6)$ | $(3,-)$ |
|  | $S_{2,15}$ | $(6,6)$ | $(-,-)$ |
|  | $S_{3,1} \cong S_{3,2}$ | $(3,3)$ | $(2,3)$ |
|  | $S_{3,4}$ | $(3,3)$ | $(1,2)$ |
|  | $S_{3,8}$ | $(3,3)$ | $(3,3)$ |
|  | $S_{3,3}$ | $(4,5)$ | $(4,4)$ |
|  | $S_{3,5} \cong S_{3,6}$ | $(4,5)$ | $(2,4)$ |
|  | $S_{3,9} \cong S_{3,10}$ | $(4,5)$ | $(4,5)$ |
|  | $S_{3,12}$ | $(5,5)$ | $(2,4)$ |
|  | $S_{3,7}$ | $(5,6)$ | $(3,-)$ |
|  | $S_{3,11}$ | $(5,6)$ | $(5,5)$ |
|  | $S_{3,13} \cong S_{3,14}$ | $(5,6)$ | $(3,-)$ |
|  | $S_{3,15}$ | $(5,6)$ | $(-,-)$ |

Table 2.2: Removal vectors of Structures $S_{a,b}$ with $a \geq 4$. Black vertices are vulnerable.

| Component | Name | removal vector | |
|---|---|---|---|
| | | CNP-V | CNP-NDV |
|  | $S_{4,1} \cong S_{4,2} \cong S_{4,4} \cong S_{4,8}$ | $(3,3)$ | $(1,3)$ |
|  | $S_{4,3} \cong S_{4,6} \cong S_{4,9} \cong S_{4,12}$ | $(3,5)$ | $(2,4)$ |
|  | $S_{4,5} \cong S_{4,10}$ | $(3,5)$ | $(2,5)$ |
|  | $S_{4,7} \cong S_{4,11} \cong S_{4,13} \cong S_{4,14}$ | $(3,6)$ | $(3,-)$ |
|  | $S_{4,15}$ | $(3,6)$ | $(-,-)$ |
|  | $S_{5,1} \cong S_{5,4}$ | $(3,3)$ | $(1,2)$ |
|  | $S_{5,2} \cong S_{5,8}$ | $(3,3)$ | $(1,3)$ |
|  | $S_{5,3} \cong S_{5,6} \cong S_{5,9} \cong S_{5,12}$ | $(3,5)$ | $(2,4)$ |
|  | $S_{5,5}$ | $(3,5)$ | $(2,4)$ |
|  | $S_{5,10}$ | $(3,5)$ | $(2,5)$ |
|  | $S_{5,7} \cong S_{5,13}$ | $(3,6)$ | $(3,-)$ |
|  | $S_{5,11} \cong S_{5,14}$ | $(3,6)$ | $(3,-)$ |
|  | $S_{5,15}$ | $(3,6)$ | $(-,-)$ |
|  | $S_{6,1} \cong S_{6,2} \cong S_{6,4} \cong S_{6,8}$ | $(3,3)$ | $(1,2)$ |
|  | $S_{6,3} \cong S_{6,5} \cong S_{6,6}$ $\cong S_{6,9} \cong S_{6,10} \cong S_{6,12}$ | $(3,5)$ | $(2,4)$ |
|  | $S_{6,7} \cong S_{6,11} \cong S_{6,13} \cong S_{6,14}$ | $(3,5)$ | $(3,-)$ |
|  | $S_{6,15}$ | $(3,5)$ | $(-,-)$ |
|  | $S_{7,1} \cong S_{7,4}$ | $(2,2)$ | $(2,2)$ |
|  | $S_{7,2}$ | $(2,2)$ | $(1,2)$ |
|  | $S_{7,3} \cong S_{7,6}$ | $(3,3)$ | $(2,-)$ |
|  | $S_{7,5}$ | $(3,3)$ | $(3,-)$ |
|  | $S_{7,7}$ | $(3,3)$ | $(-,-)$ |
|  | $S_{8,1} \cong S_{8,2} \cong S_{8,4}$ | $(2,2)$ | $(1,2)$ |
|  | $S_{8,3} \cong S_{8,5} \cong S_{8,6}$ | $(2,3)$ | $(2,-)$ |
|  | $S_{8,7}$ | $(2,3)$ | $(-,-)$ |
|  | $S_{9,1} \cong S_{9,2}$ | $(1,1)$ | $(1,-)$ |
|  | $S_{9,3}$ | $(1,1)$ | $(-,-)$ |

of $G - C$. Thus, $C'$ is a critical node cut for $(G, A, k, x)$. Hence, $C' \setminus \{v\}$ is a critical node cut for $(G - \{v\}, A \setminus \{v\}, k - 1, x)$.

Let $C$ be a critical node cut for $(G', A \setminus \{v\}, k-1, x)$. It follows that $C \cup \{v\}$ is a critical node cut for $(G, A, k, x)$.

The running time follows directly from Lemma 2.20. □

Before we continue with the next reduction rule, we observe that in an instance of CNP-V there is no connected component with at most four vertices with a removal vector $(i, j)$ with $j > 2 \cdot i$. Furthermore, all connected components with a removal vector $(i, j)$ with $j = 2 \cdot i$ have a removal vector of $(3, 6)$. This observation can be proven by computing all removal vectors for possible connected components as it has been done in Tables 2.1 and 2.2. In consequence, the first deletion in a connected component removes at least as many vulnerable connections as the second. This leads to the following observation.

**Observation 2.23** *In an instance of* CNP-V *there is no connected component with at most four vertices with a removal vector $(i, j)$ where $j > 2 \cdot i$ and if the equation $j = 2 \cdot i$ is fulfilled, then $i = 3$.*

At first of connected components in which not all vulnerable connections are removed by removing one vertex, we consider the connected components with removal vector $(3, 6)$. One example of a connected component with removal vector $(3, 6)$ is a cycle with four vertices in which are at least three vulnerable.

**Reduction Rule 2.24** *If $G$ contains a connected component $K$ that has a removal vector of $(3, 6)$ but $G$ does not contain a connected component that has a removal vector of $(p, q)$ with $p > 3$, then:*

- *If $k \geq 2$ decrease $k$ by two and remove $K$ from $G$.*
- *If $k = 1$ return yes if and only if the connectivity of $G$ is at most $x + 3$*

**Lemma 2.25** *For instances of* CNP-V*, Reduction Rule 2.24 is safe and can be applied exhaustively in $\mathcal{O}(n)$ time.*

**Proof** *Safeness:* By the requirements of Reduction Rule 2.24 the graph $G$ does not contain a connected component with removal vector $(p, q)$ with $p > 3$. Let $G$ contain a connected component $K$ with removal vector $(3, 6)$. We observe the two cases for $k$ separately.

*Case $k = 1$:* If $k = 1$ then at most one vertex of $G$ can be added to a critical node cut. It follows from Observation 2.19 that we can add any vulnerable vertex $v$ from $K$ to a critical node cut by which we reduce the connectivity of $G$ by three. So, with $\{v\}$ a critical node cut for the instance exists if and only if the connectivity of $G - \{v\}$ is at most $x$ if and only if the connectivity of $G$ is at most $x + 3$.

*Case $k \geq 2$:* Just like in the first case we observe that if we remove a single vertex we can remove at most three vulnerable connections. It follows from Observation 2.23 that with two vertices, at most six vulnerable connections can be removed. As $K$ has removal vector $(3, 6)$, $K$ does not have removal vector $(3, 5)$ and we follow that $K$ is not a clique. Thus, if we remove two vertices from $K$, the other two vertices are not connected anymore and because $K$ has a removal vector of $(3, 6)$ six vulnerable connections are removed by that. Thus, it is the best option to remove $K$ and decrease $k$ by two.

The running time follows directly from Lemma 2.20. $\square$

Now, we consider the other case, that we can not remove all vulnerable connections of $K$ by removing one vertex like it is the case with $S_{1,15}$.

**Reduction Rule 2.26** *Let $i \in [1, 6]$ be an integer. If $k \geq 1$ and $G$ contains a connected component $K$ that has a removal vector of $(i, j)$ where $j \in [i+1, 2 \cdot i - 1]$ but $G$ does not contain a connected component that has a removal vector of $(p, q)$ with $p > i$ or $q = 2 \cdot i$, then decrease $k$ by one and remove $v$ from $G$ where $v$ is the vertex from $K$ which has to be removed first such that with second removal $j$ vulnerable connections are removed.*

Notice, that if $q = 2 \cdot i$, we can follow from Observation 2.23 that $i = 3$ and there exists no connected component with removal vector $(3, 6)$.

**Lemma 2.27** *Reduction Rule 2.26 is safe and for every fixed $i \in [1, 6]$ can be applied exhaustively in $\mathcal{O}(n)$ time.*

**Proof** *Safeness:* We show that $(G, A, k, x)$ is a yes-instance of CNP-V if and only if $(G - \{v\}, A \setminus \{v\}, k - 1, x)$ is a yes-instance of CNP-V.

Let $C$ be a critical node cut for $(G, A, k, x)$. Nothing is to show if $v \in C$. So we assume that $v \notin C$. Because of the requirements of Reduction Rule 2.26 the graph $G$ does not contain a connected component with removal vector $(p, q)$ where $p > i$ or $q \geq 2 \cdot i$. It follows from Observation 2.19 that by removing $t$ vertices from $G$, we can remove at most $t \cdot i$ vulnerable connections. Thus, for every $w \in C$ also $C' := (C \cup \{v\}) \setminus \{w\}$ is a critical node cut for $(G, A, k, x)$. Hence, $C' \setminus \{v\}$ is a critical node cut for $(G - \{v\}, A \setminus \{v\}, k - 1, x)$.

Let $C$ be a critical node cut for $(G - \{v\}, A \setminus \{v\}, k - 1, x)$. It follows that $C \cup \{v\}$ is a critical node cut for $(G, A, k, x)$.

The running time follows directly from Lemma 2.20. $\square$

Now, we can show that we can solve CNP-V on graphs in which every non-trivial connected component contains at most four vertices in linear time: For this, we only need to exhaustively apply the Reduction Rules 2.21, 2.24 and 2.26 until $k = 0$ or no more vulnerable vertices are left. Then, return yes if and only if the connectivity of the remaining graph is at most $x$. Observe, that for every $i \in [1, 6]$ the Reduction Rules 2.21 and 2.26 as well as Reduction Rule 2.24 can only be applied exhaustively once. Thus, as applying all these reduction rules exhaustively takes only linear time, the overall running time is linear. It directly follows that:

**Theorem 2.28** CNP-V *can be solved in $\mathcal{O}(n)$ time on graphs, in which every connected component contains at most four vertices.*

**Adaption to CNP-NDV** We provided this algorithm only for instances of CNP-V. We only prove this version, as we do not need a similar algorithm for CNP-NDV in the rest of the thesis. Nevertheless, the algorithm behind Theorem 2.28 can easily be adjusted to CNP-NDV. Therefore, the removal vectors of all components have to be recomputed. Observe, that Observation 2.19 is not true on instances of CNP-NDV in general, as it is the case for a $P_4$, where one of the vertices with degree two is vulnerable.

Removal vectors for instances of CNP-NDV have to be adjusted a bit, as it is not possible to remove two vertices in every connected component $K$, as $K$ could have less than two non-vulnerable vertices. In that case we define the removal vector $(i, -)$ if $K$ contains exactly one non-vulnerable vertex and $i = |K| - 1$ and we define the removal vector $(-, -)$ if $K$ only consists of vulnerable vertices. Tables 2.1 and 2.2 depicts also all removal vectors for components when given as input for CNP-NDV.

Observe, that some observations we made for CNP-V in this subsection are not correct for CNP-NDV. Observation 2.19 is incorrect for every connected component that does not contain exactly one non-vulnerable vertex. Observe, that Observation 2.23 is not correct: The connected component which has a paw as underlying graph in which the vertex with degree 1 and the vertex with degree 3 are vulnerable has removal vector $(2, 4)$ and a connected component that has a cycle of length 4 with one vulnerable vertex has removal vector $(1, 3)$. Thus, we adjust Observation 2.19 to the following.

**Observation 2.29** *In an instance of* CNP-V *there is no connected component with at most four vertices with a removal vector* $(i, j)$ *where* $j > 2 \cdot i + 1$ *and:*

- *if the equation* $j = 2 \cdot i$ *is fulfilled, then* $i \in \{1, 2\}$.
- *if the equation* $j = 2 \cdot i + 1$ *is fulfilled, then* $i \in \{1, 2\}$.

Observe that with this observation we can still focus on the first number of the removal vector in the decision which connected component we we want to treat next: Let $K$ be a connected component with removal vector $(i + 1, j)$ where $j \geq i + 1$ and $K'$ be a component with removal vector $(i, 2 \cdot i + 1)$. Removing one vertex from $K$ and one vertex from $K'$ removes $2 \cdot i + 1$ vulnerable connections and is thereby as good as removing two vertices from $K'$.

We will now provide an algorithm for solving CNP-NDV in linear time on graphs in which the biggest non-trivial connected component contains at most four vertices. We will not prove the correctness of this algorithm however.

**Step 0.** Let $(G, A, k, x)$ be an instance of CNP-NDV. Set $i = 5$. Until $k = 0$ perform Steps 1 to 3. Once $k = 0$ at any point of the algorithm, then return yes if and only if the connectivity of $G$ is at most $x$.

**Step 1.** For every connected component $K$ with removal vector $(i, 2 \cdot i + 1)$ execute:

- If $k \geq 2$, remove $K$ from $G$ and reduce $k$ by two.
- If $k = 1$, return yes if and only if the connectivity of $G$ is at most $x + i$.

**Step 2.** For every connected component $K$ with removal vector $(i, 2 \cdot i)$ execute:

- If $k \geq 2$, remove $K$ from $G$ and reduce $k$ by two.
- If $k = 1$, return yes if and only if the connectivity of $G$ is at most $x + i$.

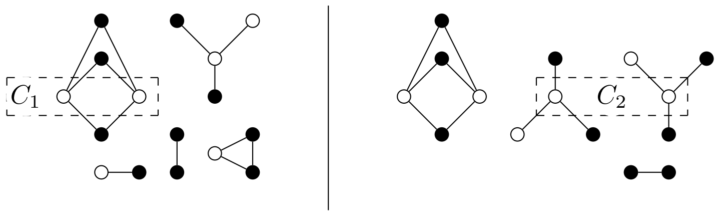


Figure 2.2: A counter example that with connected components of size 5, we can not find a greedy algorithm, that only considers the removal vector and $k$. Vulnerable vertices are pictured in black.

**Step 3.** For every connected component $K$ with removal vector $(i, i)$ or $(i, -)$ remove $K$ from $G$ and reduce $k$ by one. For every connected component $K$ with removal vector $(i, j)$ for some $j > i$ remove one vertex $v$ in $K$ such that together with the second removal of $K$ exactly $j$ vulnerable connections are removed and decrease $k$ by one. If both cases are not possible anymore, decrease $i$ by one and continue with Step 1.

**Theorem 2.30** CNP-NDV *can be solved in $\mathcal{O}(n)$ time on graphs, in which every non-trivial connected component contains at most four vertices.*

**Limits of our algorithm** We can wonder, if the connected component size 4 is somehow magic, or whether we also can also solve the instance $(G, A, k, x)$ of CNP-V or CNP-NDV in linear time, if the size of the biggest component is bounded by any integer $c$. In fact, the greedy algorithm that we provided can not be used to compute a solution for an instance $(G, A, k, x)$ of CNP-V or CNP-NDV, if $G$ contains connected components with five vertices. Consider the following example:

We define a connected component $\overline{K}$ that as underlying graph has the complete bipartite graph $K_{2,3}$ with vertex bipartition $(P, Q)$ with $|Q| = 3$. The only vulnerable vertices in $\overline{K}$ are the vertices of $P$.

Let $G_1$ contain $\overline{K}$, and furthermore $G_1$ contains connected components that are isomorphic to $S_{2,6}$, $S_{8,3}$, $S_{9,1}$ and $S_{9,3}$. Let $G_2$ contain $\overline{K}$, as well as two connected components that are isomorphic to $S_{2,6}$ and $S_{9,3}$. Figure 2.2 depicts the graphs $G_1$ and $G_2$.

The only critical node cut $C_1$ for the instance $(G_1, A_1, k := 2, x := 15)$ is $P$, while the only critical node cut for the instance $(G_2, A_2, k := 2, x := 15)$ consists of the two centers of the stars in $G_2$ that are isomorphic to $S_{2,6}$. It follows, that for a graph with connected components with at most five vertices it is not sufficient to consider the removal vector and the size of $k$. We need more information to do a correct greedy-step. One idea could be to check how many connected components with the same removal vector are in the graph.

# Chapter 3

# Fixed-Parameter Algorithms

In the previous chapter we have proven that CNP-V and CNP-NDV are NP-hard. In this chapter we present parameters for which the problems are "fast" to solve, if the parameters are small. In Section 3.1 we provide fixed-paramter algorithms to solve the problems CNP-V and CNP-NDV with respect to the parameter $k+x$ and in Section 3.2, with respect to the neighborhood-diversity nd of the input graph. Also, in Section 3.3 we prove that CNP-V is fixed-parameter tractable with respect to $y$, the number of vulnerable connections that need to be deleted, and in Section 3.4 we prove that CNP-NDV is fixed-parameter tractable with respect to the parameter $|V \setminus A|$, the number of non-vulnerable vertices.

For Sections 3.1 and 3.2 we consider a problem that generalizes CNP-V and CNP-NDV.

Critical Node Problem with Vulnerable and Non-Deletable Vertices (CNP-VNDV)

**Input**: Simple graph $G = (V, E)$, two vertex sets $A, N \subseteq V$, 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 connectivity of $G - C$ is at most $x$?

We call the set $C$ in the question a *critical node cut* of the instance $(G, A, N, k, x)$ of CNP-VNDV and a vertex $v \in N$ is called *non-deletable* while a vertex $w \notin N$ is called a *deletable*. Observe that CNP-V and CNP-NDV are special cases of CNP-VNDV with either $N = \emptyset$ or $N = A$ respectively. Thus, if we prove that CNP-VNDV is fixed-parameter tractable with respect to a parameter $\lambda$, then also CNP-V and CNP-NDV are fixed-parameter tractable with respect to the parameter $\lambda$.

## 3.1 Parametrization by $\mathbf{k + x}$

In this section we show that CNP-V and CNP-NDV are fixed-parameter tractable with respect to the parameter $k + x$. Hermelin et al. [HKKN16] showed that a solution for CNP can be found in $\mathcal{O}(3^{k+x} \cdot (x^{k+2} + n))$ time and thus, CNP is fixed-parameter tractable with regard to the parameter $k+x$. The algorithm that we provide has a shorter running time than their algorithm. We start by proving a fact about vertices of CNP-VNDV, that we use later as branching rule.

**Lemma 3.1** *Let $G = (V, E)$ be a graph and let $v \in V \setminus N$ be an arbitrary vertex. An instance $(G, A, N, k, x)$ of* CNP-VNDV *is a yes-instance, if and only if $(G - \{v\}, A \setminus \{v\}, N, k-1, x)$ or $(G, A, N \cup \{v\}, k, x)$ is a yes-instance of* CNP-VNDV*.*

**Proof** Let $(G, A, N, k, x)$ 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 connectivity of at most $x$. If $v \in C$, then the instance $(G - \{v\}, A \setminus \{v\}, N, k-1, x)$ is also a yes-instance, because the set $C' = C \setminus \{v\}$ is a critical node cut. Otherwise, if $v \notin C$, then $(G, A, N \cup \{v\}, k, x)$ is a yes-instance, as $C$ is also a critical node cut for $(G, A, N \cup \{v\}, k, x)$.

Conversely, let $(G - \{v\}, A \setminus \{v\}, N, k-1, x)$ or $(G, A, N \cup \{v\}, k, x)$ be a yes-instance of CNP-VNDV. First, suppose there exists a critical node cut $C$ for the instance $(G - \{v\}, A \setminus \{v\}, N, k-1, x)$. The set $C' := C \cup \{v\}$ is a critical node cut for $(G, A, N, k, x)$, for these three reasons: the size of $C'$ is $|C|+1 \leq (k-1)+1 = k$, since $G - C' = (G - \{v\}) - C$ every vulnerable connection in $G - C'$ is also a vulnerable connection in $(G - \{v\}) - C$, and since $v \notin N$ we conclude that $C' \subset V \setminus N$. For the other case let $(G, A, N \cup \{v\}, k, x)$ 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 $(G, A, N, k, x)$. □

We want to point out that we required $A \subseteq V$ as input for CNP-VNDV. Thus, if $v \in A$, the input $(G - \{v\}, A, N, k-1, x)$ would not fulfill the requirement $A \subset V(G - \{v\})$. So we write $(G - \{v\}, A \setminus \{v\}, N, k-1, x)$.

We can also show a similar observation for every pair of vertices. However, for the branching we do later we only need edges. So we will restrict the next observation to edges.

**Lemma 3.2** *Let $G = (V, E)$ be a graph and $\{u, v\} \in E$ an arbitrary edge of $G$. An instance $(G, A, N, k, x)$ of* CNP-VNDV *is a yes-instance, if and only if*

1. *$u \in V \setminus N$ and $(G - \{u\}, A \setminus \{u\}, N, k-1, x)$ is yes-instance of* CNP-VNDV*,*
2. *$v \in V \setminus N$ and $(G - \{v\}, A \setminus \{v\}, N, k-1, x)$ is yes-instance of* CNP-VNDV *or*
3. *$(G, A, N \cup \{u, v\}, k, x)$ is yes-instance of* CNP-VNDV*.*

**Proof** Let $(G, A, N, k, x)$ be a yes-instance of CNP-VNDV and let $C$ be a critical node cut. In the case that $w \in C$ for $w \in \{u, v\}$, we conclude that $w \notin N$ and the vertex set $C \setminus \{w\}$ is a critical node cut for the instance $(G - \{w\}, A \setminus \{w\}, N, k-1, x)$ and thus $C \setminus \{w\}$ is a yes-instance. Otherwise, if $u, v \notin C$, then $(G, A, N \cup \{u, v\}, k, x)$ is a yes-instance, as $C$ is also a critical node cut for $(G, A, N \cup \{u, v\}, k, x)$.

Conversely, let $(G - \{u\}, A \setminus \{u\}, N, k-1, x)$ or $(G - \{v\}, A \setminus \{v\}, N, k-1, x)$ or $(G, A, N \cup \{u, v\}, k, x)$ be a yes-instance of CNP-VNDV. If one of the two first instances is a yes-instance, we know from Lemma 3.1 that $(G, A, N, k, x)$ of CNP-VNDV is a yes-instance. Suppose, $(G, A, N \cup \{u, v\}, k, x)$ is a yes-instance of CNP-VNDV and let $C$ be the corresponding critical node cut. Then, $C$ is also a critical node cut for $(G, A, N, k, x)$. □

Now we show that CNP-NDV is fixed-parameter tractable with respect to the parameter $k + x$. For this, we provide an even stronger result and show that an instance $(G, A, N, k, x)$ of CNP-VNDV can be solved in $\mathcal{O}(2^{k+x} \cdot (n + m))$ time, if $A \subseteq N$. We closely relate to the algorithm Hermelin et al. [HKKN16] provided to show that CNP is fixed-parameter tractable with respect to the parameter $k + x$. For a graph $G$, the algorithm of Hermelin et al. picked an arbitrary edge $\{u, v\}$ from $G$ and branched into the options of removing $u$ or $v$ from the graph $G$, or keeping $\{u, v\}$ in the graph $G$. In an instance of CNP-VNDV we cannot add non-deletable vertices to a critical node cut. Thus, if we pick an edge $\{d, v\}$ that is incident with a vulnerable vertex $d$, we only have two options: adding $v$ to a critical node cut or keeping $\{d, v\}$ in the graph.

**Theorem 3.3** *An instance $(G, A, N, k, x)$ of* CNP-VNDV *with $A \subseteq N$ can be solved in $\mathcal{O}\left(2^{k+x} \cdot (n + m)\right)$ time.*

**Proof** *Intuition:* In the algorithm we pick neighbors $v$ of $N$ and branch into removing $v$ from the graph or make $v$ non-deletable and therefore increase the number of vulnerable connections in $G[N]$.

*Algorithm:* Let $(G, A, N, k, x)$ be an instance of CNP-VNDV with $A \subseteq N$.

**Step 0.** If $k < 0$ or the connectivity of $G[N]$ is greater than $x$, return no in this branch of the decision-tree. If the connectivity of $G$ is at most $x$, return yes.

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

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

*Correctness:* The returns in Step 0 are correct by easy observations. If in Step 2 the neighborhood of $N'$ is empty, then observe:

1. The connectivity of $G[N]$ is at most $x$, because we did not return no in Step 0.

2. In $G[N]$ every vertex that is connected to a vulnerable vertex is contained in $N'$ and $A \subseteq N'$.

Because of the second point and because the neighborhood of $N'$ is empty, every vertex that is connected to a vulnerable vertex in $G$ is contained in $N'$. Thus, the connectivity of $G$ is the connectivity of $G[N]$. Therefore, this return is correct.

Let $v$ be a neighbor of the set $N'$ computed in Step 1. The vertex $v$ is not in $N$: Assume towards a contradiction $v \in N$. Because, $v$ is a neighbor of $N'$, we can assume $\{u, v\}$ with $u \in N'$. Because $u \in N'$ we follow $u \in A$ or $u$ is connected to a vulnerable vertex. In both cases $v$ is connected to a vulnerable vertex. Thus, $v \in N'$ contradicting $v$ being a neighbor of $N'$.

It follows that the vertex $v$ in Step 2, which is a neighbor of $N'$, is in $V \setminus N$. We can apply Lemma 3.1 to prove that the branching in Step 2 is correct.

*Running time:* If a vertex $v$ with $\{u, v\} \in E(G), u \in N'$ is added to $N$, the connectivity of $G[N]$ increases: The vulnerable vertex that is connected to $u$ is then also connected to $v$, or $u \in A$ and $v$ is adjacent to a vulnerable vertex. It follows that the algorithm can choose the instance $(G, A, N \cup \{v\}, k, x)$ at most $x$ times and the instance $(G - \{v\}, A \setminus \{v\}, N, k-1, x)$ at most $k$ times. Afterwards, the algorithm returns in Step 0 or Step 2. It follows that the decision tree has size of $2^{k+x}$.

In every step of the decision tree, we have to compute the connectivity for several graphs, which can be done in $\mathcal{O}(n+m)$ time, due to Lemma 2.14. With breadth-first search, we can compute the set $N'$ in Step 1 within $\mathcal{O}(n+m)$ time. Hence, the algorithm is done in $\mathcal{O}(2^{k+x} \cdot (n+m))$ time. □

Because CNP-NDV is the special case of CNP-VNDVwith $A = N$ we conclude:

**Corollary 3.4** CNP-NDV *can be solved in* $\mathcal{O}\left(2^{k+x} \cdot (n+m)\right)$ *time.*

The algorithm that we provided in the proof of Theorem 3.3 cannot be applied to an instance of CNP-V, because the condition $A \subseteq N$ is not satisfied with $N = \emptyset$. Informally, the reason is that we do not know what to do with a vulnerable vertex, other than in CNP-NDV where we cannot remove vulnerable vertices It is obvious that we have to search for another strategy when we consider CNP, which is the special case of CNP-V with $A = V$. Therefore, we provide another algorithm that shows us that CNP-V is fixed-parameter tractable with respect to the parameter $k + x$. Like the algorithm of Theorem 3.3, the algorithm we provide next follows the pattern of the algorithm by Hermelin et al. [HKKN16] that solves CNP in $\mathcal{O}(3^{k+x} \cdot (x^{k+2} + n))$ time.

**Theorem 3.5** CNP-VNDV *can be solved in* $\mathcal{O}\left(3^{k+x} \cdot (n+m)\right)$ *time.*

**Proof** *Algorithm:* Let $(G, A, N, k, x)$ be an instance of CNP-VNDV.

**Step 0.** If $k = 0$, then return yes if and only if the connectivity of $G$ is at most $x$. If the connectivity of $G[N]$ is greater than $x$, then return no.

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

**Step 2.** Let $\{u, v\} \in E(G)$ be an edge in $G$ that fulfills

- $u \in A \cup N'$ and $v \notin N$ or
- $u \in N \setminus N'$ and $v \in A$.

If such an edge does not exist in $G$, then return yes. Otherwise, pick such an edge $\{u, v\} \in E(G)$ arbitrarily and branch into one of the following three cases:

1. $(G - \{v\}, A \setminus \{v\}, N, k-1, x)$.
2. $(G, A, N \cup \{u, v\}, k, x)$.
3. If $u \in A \setminus N$, then $(G - \{u\}, A \setminus \{u\}, N, k-1, x)$ is an option. If $u \in N$, then this option is not available.

*Correctness:* If $k = 0$, the only potential critical node cut is $C = \emptyset$, and we can return yes if and only if the connectivity of $G$ is at most $x$. If the connectivity of $G[N]$ is greater than $x$, we cannot find a critical node cut $C \subseteq V \setminus N$. Thus, the returns in Step 0 is correct.

*Claim 3.5a:* The return in Step 2 is correct.
*Proof;* If in Step 2 no edge $\{u, v\}$ exists with $u \in A \cup N'$ and $v \notin N$, then we observe:

- The neighborhood of $N'$ is empty, because if a vertex $v$ of $N$ is a neighbor of $N'$, $v \in N'$.
- The neighborhood of $A \setminus N'$ is a subset of $N \setminus N'$.

Thus, if additionally no edge $\{u, v\}$ with $u \in N \setminus N'$ and $v \in A$ exists, all vulnerable vertices are included in the set $N'$, and the neighborhood of $N'$ is empty. Thus, the connectivity of $G$ is the connectivity of $G[N]$, which after Step 0 is at most $x$. It follows, that the return in Step 2 is correct. $\diamond$

We want to use Lemma 3.2 to show the correctness of the branching in Step 2. Let $v$ be a neighbor of $A \cup N'$. It is required for $v$ to not be in $N$ in the first option for $\{u, v\}$ that is $u \in A \cup N'$ and $v \notin N$. In the second option, which is $u \in N \setminus N'$ and $v \in A$, observe: if $u \in N \setminus N'$ and $v \in A$, we conclude $v \notin N$, because otherwise $u \in N'$. In option 3, we require $u \in N$. Thus, all requirements of Lemma 3.2 are correct.

*Running time:* In order to find a suitable edge, we iterate over all edges. For every edge $e$ that we have to check if the vertices are deletable or vulnerable that $e$ is incident with, in order to to check in which option we branch to. Because we saved that in a Boolean flag, for every edge this can be done in constant time. Therefore, finding a suitable edge can be done in $\mathcal{O}(m)$ time. Notice that when in the branching step we pick the first or the third option, we decrease $k$ by one. It follows that we can chose these options at most $k$ times. Next we show that whenever we chose option two, we increase the connectivity of $G[N]$ and thus we can chose this option at most $x$ times.

*Claim 3.5b:* Every time we return the result of this algorithm with input $(G, A, N \cup \{u, v\}, k, x)$, the connectivity of $G[N]$ is increased.
*Proof; Case 1:* We took the decision concerning an edge $\{u, v\}$ with $u \in A \cup N'$ and $v \notin N$. If $u \in A \setminus N'$, then $\{u, v\}$ is a vulnerable connection in $G[N \cup \{u, v\}]$. If $u \in N'$, then by the definition of $N'$, a vulnerable vertex $d$ exists such that $d$ and $u$ are connected in $G[N]$. It follows that over $u$, the vertices $d$ and $v$ are connected in $G[N \cup \{u, v\}]$.

*Case 2:* We took the decision concerning an edge $\{u, v\}$ with $u \in N \setminus N'$ and $v \in A$. It follows, that $\{u, v\}$ is a vulnerable connection in $G[N \cup \{u, v\}]$. $\diamond$

This bounds the height of the search-tree to $k + x$. In every step we have at most three options, such that the size of the search tree is at most $3^{k+x}$.

We can compute the Steps 0 and 1 in $\mathcal{O}(n + m)$ time with breadth-first search. We can find a suitable edge in Step 2 within $\mathcal{O}(m)$ time by iterating over $E(G)$ and having the information $v \in A$, $v \in N'$ and $v \in N$ saved in flags for every vertex $v$. Thus, the algorithm is done in $\mathcal{O}(3^{k+x} \cdot (n+m))$ time. $\square$

**Corollary 3.6** CNP-V *can be solved in* $\mathcal{O}\left(3^{k+x} \cdot (n+m)\right)$ *time.*

The result of the previous theorem improves the fixed-parameter algorithm provided by Hermelin et al. [HKKN16], which ran in $\mathcal{O}(3^{k+x} \cdot (x^{k+2} + n))$ time. The algorithm by Hermelin et al. [HKKN16] performs a brute-force on minimal solutions of an auxiliary problem, which increases the running time for every leaf. Recall, that CNP is the special case of CNP-V with $A = V$.

**Corollary 3.7** CNP *can be solved in* $\mathcal{O}\left(3^{k+x} \cdot (n+m)\right)$ *time.*

In the following, we provide another algorithm to show that CNP-V can be solved in $\mathcal{O}\left(\left(\frac{4}{3}x+2\right)^k \cdot m \cdot x\right)$ time. If $k < x \cdot \ln(3) / \ln(\frac{4}{9} \cdot x^2)$, then an algorithm with running time $\mathcal{O}\left(\left(\frac{4}{3}x\right)^k\right)$ is faster than an algorithm with running time $\mathcal{O}(3^{k+x})$. This formula marks the bound but notice that the formula is already true if $k < \frac{1}{10} \cdot x^{0.9}$. In the algorithm that we provide in the next prove, we make connected components of size at least 4 smaller by adding vertices to a critical node cut. Recall that due to Theorem 2.28 CNP-V can be solved in $\mathcal{O}(n)$ time on a graph, in which every connected component has at most three vertices. We use this algorithm in the end of the following algorithm:

**Theorem 3.8** CNP-V *can be solved in* $\mathcal{O}\left(\left(\frac{4}{3}x+2\right)^k \cdot m \cdot x\right)$ *time.*

**Proof** *Intuition:* The idea of the algorithm is that we search a subset of vertices $B$, which consists of at most $\frac{4}{3}x + 2$ vertices such that the connectivity of $G[B]$ already exceeds $x$. Thus, if there exists a critical node cut $C$, at least one vertex of $B$ is in $C$. We only consider connected components of size at least 4. If at the end of the algorithm there are only connected components of size at most 3, then return the result of the algorithm of Theorem 2.28.

*Algorithm:* Let $(G, A, k, x)$ be an instance of CNP-V.

**Step 0.** If $k < 0$, then return no in this branch of the tree. Otherwise, if the connectivity of $G$ is at most $x$, then return yes. If all remaining connected components have size of at most 3, return the result of the algorithm presented in Theorem 2.28.

**Step 1.** Pick an arbitrary vulnerable vertex $v$ that is in a connected component of size at least 4. Set $B = \{v\}$ and go to Step 2.

**Step 1b.** Pick an arbitrary vulnerable vertex $v$ that is not in $B$ and that is in a connected component of size at least 5. Add $v$ to $B$ and go to Step 2. If there are no more non-trivial connected components of size at least 4, go to Step 3.

**Step 2.** If the connectivity of $G[B]$ is greater than $x$, go to Step 3. If the connected components of $B$ are connected components in $G$, go to Step 1b. Until the connectivity of $G[B]$ is greater than $x$, or $B$ is isolated in $G$, add an arbitrary neighbor of $B$ to $B$.

**Step 3.** For every vertex $v$ of $B$, branch and return the result of this algorithm with input $(G - \{v\}, A \setminus \{v\}, k - 1, x)$.

*Correctness:* Step 0 is correct by the definition of the problem. It remains to show that the branching in Step 3 is correct. In this case this means that we want to show the following:

*Claim 3.8a:* Let $B$ be a vertex set computed according to Step 1 and Step 2. An instance $(G, A, k, x)$ of CNP-V is a yes-instance if and only if there exists a

vertex $v \in B$ such that $(G - \{v\}, A \setminus \{v\}, k-1, x)$ is a yes-instance of CNP-V.
*Proof;* Suppose that $(G, A, k, x)$ is a yes-instance with a corresponding critical node cut $C$. It is sufficient to show that $C$ contains at least one vertex of $B$. For this, we consider these two different options: One option is that $B$ is computed in Step 1 and Step 2 such that the connectivity of $G[B]$ is greater than $x$. The other option is that the connectivity of $G[B]$ is at most $x$ even after all connected components of $G$ that contain at least four vertices are added to $B$.

*Case 1:* We start with the case, that $B$ is computed in Step 1 and Step 2 such that the connectivity of $G[B]$ is greater than $x$. Suppose $B \cap C = \emptyset$. The connectivity of $G - C$ is at least as large as the connectivity of $G[B]$. However, because the connectivity of $G - C$ is at most $x$, this is a contradiction and we conclude $B \cap C \neq \emptyset$.

*Case 2:* The other case is that the connectivity of $G[B]$ is at most $x$ when all non-trivial connected components of size at least 4 are added to $B$. We observe that, because we continued after Step 0, the connectivity of $G$ is greater than $x$. Thus, $C$ is not empty. Assume that $C$ contains no vertex of $B$. It follows that all the vertices that are in $C$ are in connected components of size at most 3. We observe that cutting any amount of vertex from a connected component of size at most 3 removes at most three vulnerable connections from the graph, while cutting a vulnerable vertex from a connected component of size at least 4 removes at least three vulnerable connections. Thus, we can replace at least one vertex of $C$ with a vertex of $B$ to have a vertex set $C'$ that is still a critical node cut. It follows that there exists a critical node cut that contains at least one vertex of $B$.

Conversely, assume that $I := (G - \{v\}, A \setminus \{v\}, k-1, x)$ is a yes-instance of CNP-V for some vertex $v \in B$. Let $C \subseteq V(G - \{v\})$ be a critical node cut of $I$. We define $C' = C \cup \{v\}$. The instance $(G, A, k, x)$ is a yes instance, because $C'$ is a critical node cut for $(G, A, k, x)$. $\diamond$

We proved, that the branching that we perform in Step 3 is correct and thus the algorithm returns yes, if and only if a yes-instance of CNP-V is given as input.

*Running time:* In Step 0 we compute the connectivity of $G$, which can be done in $\mathcal{O}(n + m)$ time, according to Lemma 2.14. In Step 1 we search a vulnerable vertex that is in a connected component of size at least 4. For that we can iterate over $E$ and find all edges that are incident with a vulnerable vertex $v$ and compute whether $v$ is in a connected component of size at least 4 by iterating at most 4 times over $E$. Thus, Step 1 is done in $\mathcal{O}(m)$ time.

Finding a neighbor of $B$ in Step 2 can be done in $\mathcal{O}(m)$ time. Because we increase the connectivity of $B$ every time Step 2 is called, we start Step 2 at most $x$ times. Altogether, Step 2 is done in $\mathcal{O}(m \cdot x)$ time, which is called in every branch. In order to compute the size of $B$, consider the following: Generally, a non-trivial connected component $C$ of size $c$, has a connectivity of at least $c - 1$. Thus, $B$ contains at most $\frac{1}{c-1}x$ components of size $c$ which are altogether $\frac{c}{c-1}x$ vertices, when the connectivity of $B$ is at most $x$. The number $\frac{c}{c-1}x$ is decreasing, if $c$ is increasing. Thus, $\frac{c}{c-1}x$ is the highest for $c = 4$. If $x$ is divisible by 4 and $\frac{x}{3}$ connected components of size four are in $B$, the connectivity of $G[B]$ is exactly $x$. Thus, we need to add another two vertices such that the connectivity of $B$ exceeds $x$. Altogether, the size of $B$ is at most $\frac{4}{3}x + 2$. In Step 3 we branch into $|B|$ options, which are at most $\frac{4}{3}x + 2$, as proven above. This can be done at most $k$ times, as $k$ is decreased every time by exactly one. Hence, the size of the search tree is at most $\mathcal{O}\left((\frac{4}{3}x+2)^k\right)$. The steps at every vertex of the search tree can be done in $\mathcal{O}(m \cdot x)$ time. Altogether, the algorithm runs in $\mathcal{O}\left((\frac{4}{3}x+2)^k \cdot m \cdot x\right)$ time. □

We proved that CNP-V and CNP-NDV are fixed-parameter tractable with respect to the parameter $k + x$. It directly follows that CNP-V and CNP-NDV are fixed-parameter tractable for some other parameters as well. After Reduction Rule 2.9 is applied, we can assume $|A| > k$ for instances of CNP-V.

**Corollary 3.9** CNP-V *is fixed-parameter tractable with respect to the parameter* $|A| + x$*.*

Recall that, after Reduction Rule 2.5 is applied, we can assume $y > k$. Recall that the parameter $p$ is the number of vulnerable connections. By the definition of $p$, which is number of vulnerable connections, we know that $p \geq x + y$ if $x$ is chosen to be at most $p$. We have the following result:

**Corollary 3.10** CNP-V *and* CNP-NDV *are fixed-parameter tractable with respect to the parameter* $p$*.*

## 3.2 Parametrization by the Neighborhood-Diversity

In the following we show that both problems CNP-V and CNP-NDV are fixed-parameter tractable when parameterized by the parameter nd. Recall, that nd is the number of neighborhood-classes of the input graph. A neighborhood-class is defined as a subset of vertices that all have the the same open or closed neighborhood.

We start by an observation that we can refer to in the next algorithm.

**Observation 3.11** *Let $G$ be a graph that can not be separated by removing one vertex and let $C_1, \ldots, C_\ell$ be the connected components in $G$.*

- *If for $i \in [1, \ell]$ there are two vertices $u$ and $v$ in the set $A \cap C_i$, then the graphs $G - \{u\}$ and $G - \{v\}$ have the same connectivity.*
- *If for $i \in [1, \ell]$ there are two vertices $u$ and $v$ in the set $(V \setminus A) \cap C_i$, then the graphs $G - \{u\}$ and $G - \{v\}$ have the same connectivity.*
- *If for $i \in [1, \ell]$ there are two vertices $u \in A \cap C_i, v \in (V \setminus A) \cap C_i$, then the connectivity of the graph $G - \{v\}$ is higher than the connectivity of the graph $G - \{v\}$.*

In the following algorithm we show that CNP-VNDV is fixed-parameter tractable with respect to the parameter nd. As CNP-V and CNP-NDV are special cases of CNP-VNDV, we can directly follow that CNP-V and CNP-NDV are fixed-parameter tractable with respect to the parameter nd.

**Theorem 3.12** CNP-VNDV *can be solved in* $\mathcal{O}(2^{\text{nd}} \cdot k \cdot n \cdot (n + m))$ *time.*

**Proof** *Intuition:* In this algorithm we consider every subset $\mathcal{Z}$ of neighborhood-classes that do not contain a non-deletable vertex. For a given subset $\mathcal{Z}$, every neighborhood-class $K \in \mathcal{Z}$ shall be a subset of a critical node cut $C$. Each other neighborhood-class $K'$ is not a subset of the critical node cut $C$. Hence, at least one vertex of $K'$ remains in the graph $G - C$. In consequence, a vulnerable connection $\{u, v\}$ is only removed from $G - C$, if $u$ or $v$ is added to $C$.

*Algorithm:* Let $(G, A, N, k, x)$ be an instance of CNP-VNDV. We define $\mathcal{C} := \{C_1, \dots, C_{\mathrm{nd}}\}$ to be the set of all different neighborhood-classes of $G$. We define

$$\mathcal{C}' := \{K \in \mathcal{C} \mid K \cap N = \emptyset\}.$$

Do Steps 0 to 3 for every subset $\mathcal{Z} \subseteq \mathcal{C}'$. Return yes, if and only if for any subset $\mathcal{Z}$ the algorithm returns yes.

**Step 0.** If the union of the neighborhood-classes of $\mathcal{Z}$ contains more than $k$ vertices, do not consider $\mathcal{Z}$ and continue with the next set. Otherwise, initialize the set $C := \emptyset$.

**Step 1.** Add all vertices in the classes $\mathcal{Z} = \{C_{i_1}, \dots, C_{i_t}\}$ to a potential critical node cut $C$.

**Step 2.** Compute the number of vulnerable vertices of every connected component $L$ of the remaining graph $G - C$ and assign that number to the non-vulnerable vertices of $L$. For every connected component $L$ in the remaining graph $G - C$ assign the number $|L| - 1$ to every vulnerable vertex $v \in L$. Assign the value $-1$ to every non-deletable vertex $v \in N$, even if another value was assigned before.

**Step 3.** Until the size of $C$ is $k$ or there is no vertex that is deletable and not the last vertex of its neighborhood-class do the following: Add the vertex with the highest assigned number to $C$, when only vertices are considered that are deletable and not the last vertex of its neighborhood-class. If a vulnerable vertex $v$ is added to $C$, then decrease the assigned number to $w$ by one for every vertex $w$ that is in the same connected component with $v$. If a non-vulnerable vertex $v$ is added to $C$, then decrease the assigned number to $w$ by one for every vulnerable vertex $w$ that is in the same connected component with $v$. Once $C$ has $k$ vertices or there are no more vertices that are deletable and not the last one of their neighborhood-class, or there are only vertices left with a negative assigned number, return yes if the connectivity of $G - C$ is at most $x$. Otherwise, continue with the next subset of neighborhood-classes.

*Correctness:* Let $(G, A, N, k, x)$ be a yes-instance of CNP-VNDV with a critical node cut $K$ for $(G, A, N, k, x)$. We show that the algorithm returns yes. We show that the set $C$ that the algorithm computes is also a critical node cut and the algorithm returns yes. Without loss of generality, the neighborhood-classes $\mathcal{Z} = \{C_1, \dots, C_t\}$ are a subset of $K$ and the neighborhood-classes $C_{t+1}, \dots, C_{\mathrm{nd}}$ have at least one vertex which is in $G - K$. It follows that $C_i \in \mathcal{C}'$ for $i \in [1, t]$. Recall what we observed in the intuition: Adding a vertex $v$ of $C_{t+i}$ with $i \in [1, \mathrm{nd} - t]$ to a critical node cut $F$ does not separate $G - F$, because another vertex of the neighborhood-class remains in $G - F$. Observe that a vulnerable vertex in a connected component of size $c$ is in $c - 1$ vulnerable connections and a non-vulnerable vertex in a connected component with $d$ vulnerable vertices is in $d$ vulnerable connections. By definition, a critical node cut $C$ does not contain any vertex $v \in N$. Thus, we assigned to every vertex the number of vulnerable connections that are removed from the graph $G - C$, when a vertex is added to $C$. Let $C$ be the set that the algorithm computes, when $\mathcal{Z}$ is considered. We can make this assumption, as the algorithm tries every subset $\mathcal{Z}$ of $\mathcal{C}'$. We assume, that $|C| = |K| = k$, as otherwise we can add any non-deletable vertices to $K$ or $C$. Let $v_1, \ldots, v_k$ be the order in which the vertices $v_1, \ldots, v_k$ are added to $C$. Assume towards a contradiction that $C$ is not a critical node cut of $(G, A, N, k, x)$, especially $C \neq K'$ for every critical node cut $K'$ for $(G, A, N, k, x)$. For the rest of the proof, fix such a critical node cut $K'$. It follows, that an $i \in [1, k]$ exists such that $\{v_1, \ldots, v_{i-1}\} \subseteq K'$ and $v_i \notin K'$. We bring this proof to a contradiction by showing that a critical node cut exists which contains all vertices $\{v_1, \ldots, v_i\}$ and especially all vertices of vertex sets in $\mathcal{Z}$. Let $L$ be the connected component of $v_i$.

*Case 1:* $K' \setminus \{v_1, \ldots, v_{i-1}\}$ does not contain any vertex from the connected component of $v_i$ in $G - \{v_1, \ldots, v_{i-1}\}$. Because the algorithm always picks the vertex which results in the highest number of removed vulnerable connections, adding $v_i$ to a critical node cut removes at least as many vulnerable connections as any other vertex. Because adding another vertex to a critical node cut does not remove more vulnerable connections and even adding several vertices to a critical node cut does not separate the graph, the set $(K' \cup \{v_i\}) \setminus \{w\}$ for any $w \in K' \setminus \{v_1, \ldots, v_{i-1}\}$ is a critical node cut.

*Case 2:* $K' \setminus \{v_1, \ldots, v_{i-1}\}$ contains exactly one vertex $w$ from the connected component $L$ of $v_i$ in $G - \{v_1, \ldots, v_{i-1}\}$. It follows from Observation 3.11 that the vertex $v_i$ is vulnerable if and only if $L$ contains vulnerable deletable vertices that are not the last vertex of their neighborhood-class. So, if $w$ is vulnerable, then $v_i$ is vulnerable as well. Furthermore, in $K'$ we can exchange $w$ for $v_i$ in any case, because all vulnerable vertices in a connected component have the same assigned value and non-vulnerable vertices in a connected component have the same assigned value and the value is always less than the assigned value to a vulnerable vertex.

*Case 3:* $K' \setminus \{v_1, \ldots, v_{i-1}\}$ contains $j$ vertices from the connected component $L$ of $v_i$ in $G - \{v_1, \ldots, v_{i-1}\}$ with $j > 1$.

*Case 3.1:* By Observation 3.11 we know that if $v_i$ is non-vulnerable, all $j$ vertices are non-vulnerable as well. Thus, in $K'$ we can exchange any of these $j$ vertices with $v_i$, because all non-vulnerable vertices of a connected component have the same assigned value.

*Case 3.2:* If at least one of these $j$ vertices $w$ is vulnerable, $v_i$ is vulnerable as well and we can exchange $w$ with $v_i$ in $K'$, because all vulnerable vertices of a connected component have the same assigned value.

*Case 3.3:* It remains the case that all $j$ vertices are non-vulnerable, except for $v_i$ which is vulnerable. Let $w$ be any of these $j$ vertices. Let $A_L$ be the set of all $d$ vulnerable vertices in $L$ and set $q := |L|$. It follows $q > d$, because $w$ is non-vulnerable. We define $K'' := (K' \cup \{v_i\}) \setminus \{w\}$. The graph $G - K'$ in comparison to $G - K''$ does not contain the vulnerable connections $\{u, w\}$ for all $u \in A_L \setminus \{v_i\}$, but contains the vulnerable connections $\{v_i, u\}$ for all vertices $u \in L \setminus \{w\}$. Thus, $G - K'$ contains $(q - 1) - (d - 1) > 0$ vulnerable connections more than $G - K''$ and $K''$ is also a critical node cut.

It follows, that for every $i \in [1, k]$ we can find a critical node cut that contains $\{v_1, \ldots, v_k\}$ and thus $C$ is a critical node cut.

Conversely, let the algorithm return yes when the instance $(G, A, N, k, x)$ of CNP-VNDV is given as input. The algorithm only returns yes, if a set $C$ is defined with at most $k$ vertices and the connectivity of $G - C$ is at most $x$. It follows that $C$ is a critical node cut of $(G, A, N, k, x)$, which then is a yes-instance of CNP-VNDV.

*Running time:* Computing all neighborhood-classes can be done in linear time [HM91]. For each of the at most $2^{\mathrm{nd}}$ subsets of $\mathcal{C}'$, we have to do the following: In the Steps 0 to Step 2, we have to compute the number of vulnerable vertices of the connected component or the size of the connected component which takes $\mathcal{O}(n + m)$ time by breadth-first search. Then, we have to add vertices to $C$ greedily, at most $k$ times.

Finding the vertex with the highest assigned number is done in $\mathcal{O}(n)$ time by iterating over all vertices. Remark that when we add a vertex $v$ to $C$, we have to decrease the assigned number for all or some vertices that are in the same connected component as $v$ by one. For that, we have to find all vertices that are in the same connected component which is done in $\mathcal{O}(n + m)$ time by breadth-first search. Altogether, this leads us to an algorithm running in $\mathcal{O}(2^{\mathrm{nd}} \cdot k \cdot n \cdot (n + m))$ time. □

Because CNP-V and CNP-NDV are special cases of CNP-VNDV, we conclude the following:

**Corollary 3.13** CNP-V *and* CNP-NDV *are fixed-parameter tractable with respect to the neighborhood-diversity of the input graph.*

By Observation 2.4 we know that the parameter nd can be bounded by $\mathcal{O}(2^{\mathrm{vc}})$, where vc is the vertex cover number. It follows, that:

**Corollary 3.14** CNP-V *and* CNP-NDV *are fixed-parameter tractable with respect to the vertex cover number of the input graph.*

## 3.3 Parametrization by the Number of Connections to be removed

In this section, we consider a parametrization of the problem CNP-NDV with the parameter $y$ which is the number of vulnerable connections that have to be removed from the graph $G$. Hermelin et al. [HKKN16] showed that CNP is fixed-parameter tractable with respect to the parameter $y$. Their algorithm finds a solution for every connected component with brute-force and then computes the solution for the entire graph. We can use a similar algorithm for CNP-V, but need to alter the algorithm in two points. First, their argument for $y > k$ is not sufficient for our problem. However, after Reduction Rule 2.5 is applied, we can assume $y > k$. Also, we can assume that a connected component in $G$ has at most $y$ vertices. For that, we remove all connected components in the graph that do not have vulnerable vertices. Then, if a non-trivial connected component has more than $y$ vertices, we are dealing with a trivial yes-instance, as we can just cut a vulnerable vertex from the connected component that has at most $y$ vertices.

**Theorem 3.15** *A solution for* CNP-V *can be computed in* $\mathcal{O}(2^y \cdot y^2 \cdot n)$ *time.*

**Proof** *Algorithm:* Let $(G, A, k, x)$ be a given instance and let $C_1, \ldots, C_t$ be the connected components of $G$. We proceed as follows:

**Step 1.** For every $i \in \{1, \ldots, t\}$, check if $C_i$ contains at least a vulnerable vertex. If $C_i$ does not contain a vulnerable vertex, then remove $C_i$ from the graph $G$.

**Step 2.** Return yes, if a connected component $C_i$ of size at least $y$ exists. Return yes, if $k \geq y$.

**Step 3.** For each connected component $C_i$ of $G$ and each $k' \in [1, k]$, compute by brute-force the maximum number of vulnerable connections in $C_i$ that can be removed by deleting exactly $k'$ vertices in $C_i$. Let $T[i, k']$ denote this number.

**Step 4.** For increasing $i$, compute the maximum number of vulnerable connections that can be removed by deleting exactly $k'$ vertices in the connected components $C_1, \ldots, C_i$. Let $Q[i, k']$ denote this number. For the value $i = 1$, we set $Q[1, k'] := T[1, k']$. For $i > 1$, we set

$$Q[i, k'] := \max_{k'' < k'} Q[i-1, k''] + T[i, k' - k''].$$

**Step 5.** If $Q[t, k] < y$ return no. Otherwise, return yes.

As noted by Hermelin et al. [HKKN16], the correctness of the algorithm is obvious: optimal solutions for different connected components can be combined since the connected pairs are only contained within connected components.

*Running time:* All connected components can be computed by breadth-first search. In Step 1 and Step 2 we iterate over all vertices of the connected components and thus the steps are performed in linear time. Afterwards, the size of a connected component is at most $y$ and $k < y$. It follows that for every connected component there are $\mathcal{O}(2^y)$ possibilities to check in Step 3. The dynamic programming in Step 4 of the algorithm is then performed for $t \leq n$ different values of $i$. For each value of $i$, $k^2 \leq y^2$ possible combinations of $k'$ and $k''$ are considered. Altogether, the algorithm has a running time of $\mathcal{O}(2^y \cdot y^2 \cdot n)$. □

## 3.4 Parametrization by the Number of Non-Vulnerable Vertices

In this section we show that CNP-NDV is fixed-parameter tractable with respect to the parameter $|V \setminus A|$. This is a difference in comparison to CNP-V, which by Theorem 2.12 is NP-hard, even if $|V \setminus A| = 0$.

**Theorem 3.16** CNP-NDV *can be solved in* $\mathcal{O}\left(\binom{|V \setminus A|}{k} \cdot (n+m)\right)$ *time.*

**Proof** *Algorithm:* If $|V \setminus A| < k$, return yes, if and only if the connectivity of $G[A]$ is at most $x$. Otherwise, iterate over all subsets $K$ of $V \setminus A$ of size $k$. If for such a subset $K$ the graph $G - K$ has a connectivity of at most $x$, return yes. If no such subset exists, return yes.

*Correctness:* The correctness of the first condition is obvious.

Let $(G, A, k, x)$ be a yes-instance of CNP-NDV. Thus, there exists a critical node cut $C$. If $|C| = k$, the algorithm generates $C$ at one point and returns yes. Otherwise, let $C'$ be a subset of $V \setminus A$ of size $k$ and $C \subset C'$. The connectivity of $G - C'$ is at most the connectivity of $G - C$. Thus, $C'$ is generated by the algorithm and yes is returned.

If the algorithm returns yes, a critical node cut is generated by the algorithm and $(G, A, k, x)$ is a yes-instance.

*Running time:* There are exactly $\binom{|V \setminus A|}{k}$ subsets of $V \setminus A$ with exactly $k$ vertices. For every such set $K$, we have to compute the connectivity of the graph $G - K$. By Lemma 2.14, the connectivity of a graph can be computed in $\mathcal{O}(n + m)$ time. □

For $k \neq 0$, we remark at this point that $\binom{|V \setminus A|}{k} < 2^{|V \setminus A|}$. Thus, Theorem 3.16 shows us that CNP-NDV is fixed-parameter tractable with respect to the parameter $|V \setminus A|$.

Because $V \setminus A \subseteq V$, in Theorem 3.16 we have an algorithm that is done in at most $\mathcal{O}(n^k \cdot n \cdot m)$ time. After Reduction Rules 2.7 and 2.5 are applied, we can assume that $\overline{n} > k$ and $y > k$. We have the following results.

**Corollary 3.17** CNP-NDV *is slice-wise polynomial with respect to the parameters $k$, $\overline{n}$ and $y$.*

In the algorithm in the proof of Theorem 3.16, we iterate over all subsets of $V \setminus A$ of size $k$ to find a solution. This is enough for CNP-NDV, because a critical node cut must be disjoint from the vulnerable vertices. This is not correct for CNP-V. However, iterating over all vertex sets of size $k$ in an instance of CNP-V and performing the rest of the algorithm just like in Theorem 3.16 solves an instance of CNP-V in $\mathcal{O}(n^k)$ time. It follows:

**Lemma 3.18** CNP-V *is slice-wise polynomial with respect to the parameter $k$.*

Because in non-trivial instances of CNP-V the number of vulnerable vertices is greater than $k$, we also have the following result:

**Corollary 3.19** CNP-V *is slice-wise polynomial with respect to the parameter $|A|$.*

# Chapter 4

# Hardness for One Non-Deletable Vulnerable Vertex

In this chapter, we show that CNP-NDV is NP-hard, even if the input graph $G$ contains only one vulnerable vertex. In contrast it is apparent that CNP-V cannot be NP-hard with only one vulnerable vertex: If $k = 0$, we return yes if and only if the connectivity of $G$ is at most $x$ and if $k \geq 1$, we can return yes, because $A$ is a critical node cut.

To give a stronger result than the NP-hardness we show that CNP-NDV is W[1]-hard with respect to the parameters $\overline{n} + k + y$ and with respect to $x$, even if there is only one vulnerable vertex. This is in fact a stronger result, as in this chapter we only present reductions that can be computed in polynomial time. Recall, that $\overline{n}$ is the size of the open neighborhood of $A$ and, that $y$ is the number of vulnerable connections that need to be removed from an instance.

We also show that, even if the input graph contains only one vulnerable vertex, CNP-NDV is both W[1]-hard with respect to the parameter $\overline{n} + k + y$ on bipartite graphs and W[1]-hard with respect to $k + y$ on split graphs. Also, we prove that even on split graphs CNP-V is W[1]-hard with respect to the parameter $k$.

## 4.1 Parameterized Hardness for $\mathbf{k + y}$

Now we show that CNP-NDV is W[1]-hard with respect to the parameter $\overline{n} + k + y$, even if the input graph only contains one vulnerable vertex and diam = 2. We reduce from Clique. Recall that in Clique, a graph and an integer $\ell$ are given as input, and the question is whether there exists a subset $X$ of $\ell$ vertices that are pairwise adjacent. Clique is well-known W[1]-hard with respect to the parameter $\ell$ which is the size of the clique [CFK+15]. The reduction that we provide follows the spirit of the reduction that Fomin et al. [FGK13] provided to show that Cutting at most $k$ Vertices with Terminal is W[1]-hard with respect to the parameter $t$, which is the size of the cut. Thus, we can directly follow that CNP-NDV is W[1]-hard with respect to the param-

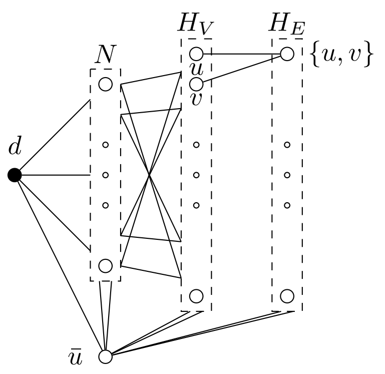


Figure 4.1: Construction of $G'$ in the proof of Theorem 4.1.

eter $k$, even if $|A| = 1$. We write an own reduction, because we want to show the stronger result that CNP-NDV is W[1]-hard with respect to the parameter $\overline{n}+k+y$, even if $|A| = 1$ and diam $= 2$. Recall that in Cutting at Most $k$ Vertices with Terminal a graph $G$, a vertex $s \in V(G)$ and two integers $k$ and $t$ are given as input and the question is, whether a vertex set $X \subseteq V(G)$ exists such that $s \in X$, $|X| \leq k$ and $|N(X)| \leq t$.

**Theorem 4.1** CNP-NDV *is* W[1]*-hard with respect to the parameter* $\overline{n}+k+y$*, even if* $|A| = 1$ *and* diam $= 2$.

**Proof** *Construction:* Let $(G, \ell)$ 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 correspond 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 4.1. We set $k := \ell + 1$ and $x := |N| + n_G + m_G + 1 - \binom{\ell}{2}$. Observe, $\overline{n} = \ell + 2 = |N \cup \{\bar{u}\}|$. Then, $y = \binom{\ell}{2} + \ell + 1$, as we are dealing with exactly one vulnerable vertex. Thus, $\overline{n}$, $k$ and $y$ are polynomial bounded by $\ell$. It is $|A| = |\{d\}| = 1$ and because of the universal vertex $\bar{u}$ the diameter of the graph $G'$ is 2. The construction is computed in polynomial time.

*Intuition:* We constructed $G'$ so that there exists a vertex in $G'$ for every vertex and for 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. In consequence, the only vertices that can be separated from $d$ without being in the critical node cut are in the vertex set $H_E$. We want to exploit the structure of a clique to prove the equivalence of the instances.

*Correctness:* Let $(G, \ell)$ 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 $\ell$. 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 $\ell$ 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 $\ell+1$. 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 $|C| \leq \ell+1$ and cutting $C$ from $G'$ deletes $y$ vulnerable connections. Because $|A| = 1$, $y$ vertices are no longer connected to the vertex $d$. 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 vertices of $N$ are adjacent to $d$ and every vertex of $H_V$ has more than $k$ vertex-distinct-paths from $d$. 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 $\ell$ vertices of $H_V$ that are pairwise adjacent in $G$. Hence, in $G$ exists a clique of size $\ell$. $\square$

Observe, that, by the construction of $G'$ in the previous proof, the graph $G'$ does not contain edges within $N$, within $H_V$, or within $H_E$. It follows that, if we remove the vertex $\bar{u}$, the constructed graph $G'$ is bipartite with vertex bipartition $(\{a\} \cup H_V, N \cup H_E)$. Because $\bar{u}$ is in every critical node cut, we can remove $\bar{u}$ and decrease $k$ by one to have an equivalent instance. Then, every vertex in $G$ has a path of length at most 2 to every vertex in $N$. It follows:

**Corollary 4.2** CNP-NDV *is* W[1]*-hard with respect to* $\overline{n} + k + y$, *even on bipartite graphs and if* $|A| = 1$ *and* $\text{diam} = 4$.

We can also alter the construction of $G'$ in Theorem 4.1 by making the vertices of $N$, $H_V$, and the vertices $d$ and $\bar{u}$ a clique. By this, the neighborhood of $A$ increases by $|H_V|$ and thus is not bounded in $\ell$. The rest of the proof however is analogue. Observe, that the graph $G'$ that we obtained consists of a clique that we just constructed and an independent set $H_E$. Thus, $G'$ is a split graph.

**Corollary 4.3** CNP-NDV *is* W[1]*-hard with respect to* $k + y$, *even on split graphs and if* $|A| = 1$ *and* $\text{diam} = 2$

We emphasize that the only vulnerable vertex is in the clique of the split graph. We can use the knowledge of Corollary 4.3 to prove that also CNP-V is W[1]-hard with respect to parameter $k$ even on split graphs. However, after Reduction Rule 2.9 is applied, we can assume that $|A| > k$ and thus in contrast to CNP-NDV, the hardness is not correct if the input graph only contains one vulnerable vertex.

**Corollary 4.4** CNP-V *is* W[1]*-hard with respect to the parameter* $k$, *even on split graphs.*

**Proof** *Construction:* Let $(G, A := \{a\}, k, x)$ be an instance of CNP-NDV with $G$ being a split graph with clique $K$ and independent set $I$. Furthermore, $N[d] = K$. All these requirements are given by the construction for Corollary 4.3. We construct an instance $(G, A', k, x')$ by setting $A' := K$ and $x' := (|K| - k) \cdot x - \binom{k}{2}$. This transformation can be computed in polynomial time.

*Correctness:* Assume that $(G, A, k, x)$ is a yes-instance of CNP-NDV. If $|K| \leq k$ then $K$ is a critical node cut for $(G, A', k, x')$. Otherwise, there exists a critical node cut $C$ for $(G, A, k, x)$. If $C \cap I \neq \emptyset$, we define a vertex set $C'$ in which every vertex of $C$ that is in the independent set is replaced with an arbitrary vertex of the clique. Because the removal of a vertex of the independent set deletes at most $|A|$ vulnerable connections and it is impossible to separate the graph by removing vertices from the independent set, $C'$ is also a critical node cut of $(G, A, k, x)$. If $|C'| < k$, add arbitrary vertices of the clique until $|C'| = k$. This is possible, because $|K| > k$.

Because $C'$ is a critical node cut of $(G, A, k, x)$, there are at most $x$ vulnerable connections in $G - C'$. Because the vulnerable vertices $A$ are the clique of $G$, there are $(|K| - k) \cdot x - \binom{k}{2}$ vulnerable connections in $G - C'$. The term $-\binom{k}{2}$ is because we may count the connections within $C - C'$ only once.

Conversely, assume that there is a critical node cut $C$ for $(G, A', k, x')$. We can assume that $|C| = k$ and that $C$ only consists of vulnerable vertices. Because $N[d] = K$, we can also assume $d \notin C$ (the vulnerable vertex in $G$). Otherwise, replace $C$ or add vulnerable vertices to $C$. If $\{u, v\}$ is a vulnerable connection in $G - C$ with $u \in A$, then all vulnerable vertices $w \in A \backslash \{v\}$ in $G - C$ are connected to $v$. Because the $|K| - k$ vulnerable vertices in $G - C$ are all connected to the same vertices and the connections within $C$ are not counted twice, there are at most $\frac{1}{|K|-k} \cdot \left(x' + \binom{k}{2}\right) = x$ vulnerable connections in $G - C$. Thus, $C$ is a critical node cut for $(G, A, k, x)$. □

## 4.2 Parameterized Hardness for x

In this section, we consider the parameter $|A| + x$ for the problem CNP-NDV. In Corollary 3.9, we proved that CNP-V is fixed-parameter tractable with respect to the parameter $|A| + x$. In contrast, we show in this section that CNP-NDV is W[1]-hard with respect to the parameter $x$, even if there exists only one vulnerable vertex.

Recall that in Cutting at Most $k$ Vertices with Terminal a graph $G$, a vertex $s \in V(G)$ and two integers $k$ and $t$ are given as input. The question is, whether there exists a vertex set $X \subseteq V(G)$ of size at most $k$ such that $s \in X$ and $|N(X)| \leq t$. Fomin et al. [FGK13] showed that Cutting at Most $k$ Vertices with Terminal is W[1]-hard with respect to $k$, the maximal number of vertices that can remain in the connected component of the terminal in the graph $G - N(X)$. We reduce from Cutting at Most $k$ Vertices with Terminal to prove that CNP-NDV is W[1]-hard with respect to the parameter $x$, even if $|A| = 1$. In our construction, the terminal is the only vulnerable vertex in our construction.

**Theorem 4.5** CNP-NDV *is* W[1]*-hard with respect to the parameter $x$, even if* $|A| = 1$ *and* diam $= 2$.

**Proof** *Construction:* Let $(G, s, k, t)$ be an instance of Cutting at Most $k$ Vertices with Terminal. We construct a graph $G'$ by adding a vertex $\bar{u}$ to $G$ that is connected to every vertex of $G$. 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 $\bar{u}$, every pair of vertices is connected with a path with length of at most 2 and so diam $= 2$.

*Correctness:* Let $(G, s, k, t)$ be a yes-instance of Cutting at Most $k$ Vertices with Terminal. Let $S$ 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 $|S| - 1 = x$ 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. $\square$

In Theorem 4.5 we reduce from Cutting at Most $k$ Vertices with Terminal. When we have a look in the reduction of Fomin et al. [FGK13], we observe that the vertex $s$ has a high degree. Thus, a natural question is what happens, if we also consider the parameter $\Delta$. Observe that, if we prove that CNP-NDV is fixed-parameter tractable with respect to $|A| + \Delta$, we know the same with respect to $\overline{n} + \Delta$ where $\overline{n}$ is the size of the neighborhood of $A$ since $\overline{n} \leq |A| \cdot \Delta$ and $|A| \leq \overline{n} \cdot \Delta$.

To compare this with CNP-V we remember that CNP-V is fixed-parameter tractable with respect to $|A| + x$, due to Corollary 3.9. Thus, CNP-V is fixed-parameter tractable with respect to $|A| + x + \Delta$ and with respect to $\overline{n} + x + \Delta$.

**Corollary 4.6** CNP-V *is fixed-parameter tractable with respect to the parameter* $\overline{n} + x + \Delta$.

To show that CNP-NDV is fixed-parameter tractable with respect to the parameter, we observe that a vertex $v$ that in the input graph is not in $N^x[A]$ should be separated from every vulnerable vertex. Thus, the set of vertices that are in a vulnerable connection is a subset of $N^x[A]$. We use these observations in the proof of the algorithm.

**Theorem 4.7** CNP-NDV *is fixed-parameter tractable with respect to the parameter* $|A| + x + \Delta$ *and with respect to the parameter* $\overline{n} + x + \Delta$.

**Proof** *Algorithm:* Let $(G, A, k, x)$ be an instance of CNP-NDV. Iterate over all subsets $X$ of $N^x_G[A]$ that have a size of at most $2 \cdot x$. For every vertex set $X$, define a vertex set $K := N_G(A \cup X)$. Return yes if $K$ contains at most $k$ vertices and the connectivity of $G - K$ is at most $x$. Return no, if we have not found a critical node cut at the end of the iteration.

Recall, that $N^x_G[A]$ contains all vertices that in $G$ can be reached in at most $x$ steps from a vulnerable vertex. Also, $N_G(A \cup X)$ contains no vertices of $A \cup X$.

*Correctness:* Consider an instance $I = (G, A, k, x)$ of CNP-NDV. We show that $I$ is a yes-instance of CNP-NDV if and only if the algorithm returns yes.

Suppose $I$ is a yes-instance and $C$ a critical node cut. Define

$$S := \{u, v \in V(G) \mid \{u, v\} \text{ is a vulnerable connection in } G - C\}.$$

Because $G - C$ has at most $x$ vulnerable connections, $S$ contains at most $2 \cdot x$ vertices. Assume towards a contradiction that there is a vertex $v \in S \setminus N^x_G[A]$ and there exists a vulnerable connection $\{d, v\}$ in $G - C$. Because $v \notin N^x_G[A]$, there exists a path $d, u_1, \ldots, u_t, v$ in $G - C$ with $t \geq x$. Consequently, there exist at least $t + 1 > x$ vulnerable connections in $G - C$. To be more precisely these are $\{d, v\}$ and $\{d, u_\ell\}$ for $\ell \in [1, t]$. This however contradicts the assumption that $C$ is a critical node cut and it follows that $S \subseteq N^x_G[A]$. Thus, $S$ is considered by the above algorithm.

Now we want to show that $A \cup S$ is isolated in $G - C$, which is the case if $N(A \cup S) \subseteq C$. Let $w$ be a neighbor of $v \in A \cup S$ and $w \notin S$. If $v \in A$, then $\{v, w\}$ is a vulnerable connection in $G$ and because $w \notin S$, $\{v, w\}$ is not a vulnerable connection in $G - C$. We conclude $w \in C$. If $v \in S \setminus A$, there exists a vulnerable connection $\{d, v\}$ in $G - C$ by the definition of $S$. Because $w$ is a neighbor of $v$, $\{d, w\}$ is a vulnerable connection in $G$ and because $w \notin S$, $\{v, w\}$ is not a vulnerable connection in $G - C$. We conclude $w \in C$. It follows that the neighborhood of $A \cup S$ is a subset of $C$. Thus, because $C$ is a critical node cut, the algorithm returns yes when regarding $S$.

Conversely, suppose the algorithm returns yes. The defined set $K$ is a critical node cut and thus $(G, A, k, x)$ is a yes-instance of CNP-NDV.

*Running time:* $A$ contains $|A|$ or at most $\overline{n} \cdot \Delta$ vertices, while the neighborhood of $A$ contains $\overline{n}$ or at most $|A| \cdot \Delta$ vertices. If for $i \geq 1$ $N^i_G(A)$ contains $c$ vertices, then $N^{i+1}_G[A]$ contains at most $c \cdot (\Delta - 1)$ vertices. It follows that $N^x_G[A]$ contains at most

$$|A| + \sum_{i=0}^{x-1} |A| \cdot \Delta \cdot (\Delta - 1)^i \text{ respectively } \overline{n} \cdot \Delta + \sum_{i=0}^{x-1} \overline{n} \cdot (\Delta - 1)^i$$

vertices. We define $z := |N^x_G[A]|$ and remember that $z$ is bounded by a function depending on $|A| + x + \Delta$ or a function depending on $\overline{n} + x + \Delta$.

The algorithm iterates over all $\sum_{i=0}^{2 \cdot x} \binom{z}{i}$ subsets of $N^x_G[A]$ that have at most $2 \cdot x$ vertices. Because $\binom{z}{i} \in \mathcal{O}(z^i)$, we conclude $\sum_{i=0}^{2 \cdot x} \binom{z}{i} \in \mathcal{O}(z^{2 \cdot x})$.

For every considered set $X$, we check whether the set $K$ is a critical node cut. For that we have to compute the connectivity of the graph $G - K$. By Lemma 2.14, this can be computed in $\mathcal{O}(n + m)$ time. Altogether, the algorithm has a running time of $\mathcal{O}(z^{2 \cdot x} \cdot (n + m))$. □

We showed that CNP-NDV can be solved in $\mathcal{O}(|N_G^x[A]|^{2 \cdot x} \cdot (n + m))$. The set $N_G^x[A]$ is a subset of $V$, and so by the previous algorithm we have also proven the following.

**Corollary 4.8** CNP-NDV *is slice-wise polynomial with respect to the parameter $x$*

# Chapter 5

# Polynomial Kernelizations

In this chapter, we present some kernelizations of polynomial size. Recall that a problem $P$ admits a kernelization for a parameter $\lambda$, if and only if the problem $P$ is fixed-parameter tractable with respect to $\lambda$ [CFK⁺15]. By that, we already know that CNP-V and CNP-NDV admit a kernelization for parameters $k + x$ and the neighborhood-diversity nd. In the following, we want to improve this positive result for some parameters.

First, we show that for every instance of CNP-NDV we can find an equivalent instance in which all vulnerable vertices are neighbors of non-vulnerable vertices. Using this fact, we show that CNP-NDV admits a kernelization with $\mathcal{O}(|V \setminus A| \cdot x)$ vertices. However, it is unlikely that CNP-V admits a kernelization for the parameter $|V \setminus A| + x$, because by Theorem 2.12 CNP-V is NP-hard, even if $x = |V \setminus A| = 0$.

In another approach, we bound the size of all neighborhood-classes. By that, we show that both problems CNP-V and CNP-NDV admit a kernelization with $\mathcal{O}(\mathrm{nd} \cdot (k + x))$ vertices with is the neighborhood-diversity nd. At last, we show that CNP-V admits a kernelization with $\mathcal{O}((|A| + \mathrm{vc}) \cdot (k + x))$ vertices and CNP-NDV admits a kernelization with $\mathcal{O}(\mathrm{vc} \cdot (k + x))$ vertices.

## 5.1 Kernelization based on Non-Vulnerable Vertices

We know by Theorem 3.16 that CNP-NDV is fixed-parameter tractable with respect to $|V \setminus A|$ and thus also with respect to the bigger parameter $|V \setminus A| + x$. In this section, we strengthen this positive result and show that CNP-NDV admits a kernelization with $\mathcal{O}(|V \setminus A| \cdot \sqrt{x})$ vertices. Unless W[1] = FPT (which we do not expect), CNP-V does not admit a kernelization with regard to the combined parameter $|V \setminus A| + x$, because due to Theorem 2.12 CNP-V is NP-hard, even if $|V \setminus A| = x = 0$.

The idea is that for a given instance $(G, A, k, x)$ of CNP-NDV we compute an equivalent instance of CNP-NDV, in which every vulnerable vertex is a neighbor of a non-vulnerable vertex. If $C$ is a critical node cut for an instance $(G, A, k, x)$ of CNP-NDV, then in $G - C$ a non-vulnerable vertex is adjacent to less than $x$ vulnerable vertices. Thus, we can limit the number of

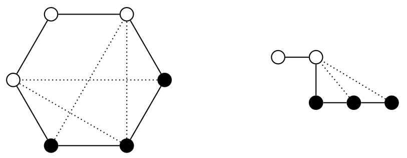


Figure 5.1: An application of Reduction Rule 5.2. The dotted lines are inserted by Reduction Rule 5.2, when this instance is given as input. The black vertices are vulnerable.

vulnerable vertices and therefore the number of vertices in the graph. To this end, we present three rules.

We start with a rule that removes connected components that are a subset of $A$ from the graph. This ensures that every vulnerable vertex is connected to a non-vulnerable vertex afterwards.

**Reduction Rule 5.1** *If there exists a connected component $K$ that is a subset of $A$, then decrease $x$ by $\binom{|K|}{2}$ and remove $K$ from the graph.*

With breadth-first search, we can compute all connected components of a graph within $\mathcal{O}(n+m)$ time. If we save in a Boolean constant which vertex is vulnerable, we can check directly in the breadth-first search, if a component contains only vulnerable vertices. Thus, Reduction Rule 5.1 can be applied exhaustively in $\mathcal{O}(n+m)$ time.

It is easy to prove that Reduction Rule 5.1 is safe. As we cannot add a vulnerable vertex to a critical node cut, we cannot add a single vertex from a connected component $K$ that consists only of vulnerable vertices to a critical node cut. Thus, we can remove $K$ from the graph, but have to decrease $x$ by the number of vulnerable connections that exist in $K$. This number is $\binom{|K|}{2}$.

Next, we present a rule that connects every vulnerable vertex $v$ to the non-vulnerable vertices which are reachable from $v$ over a path that consists of vulnerable vertices. In consequence, every vulnerable vertex is a neighbor of a non-vulnerable vertex and we only have to bound the neighborhood of every non-vulnerable vertex.

**Reduction Rule 5.2** *For every connected component $K$ in $G[A]$, connect every vertex of $K$ with every vertex of $N_G(K)$.*

Observe that, because $K$ is isolated in $G[A]$, the neighborhood of $K$ in $G$ is a subset of $V \setminus A$. Figure 5.1 depicts an application of Reduction Rule 5.2.

**Lemma 5.3** *For every instance of* CNP-NDV*, Reduction Rule 5.2 is safe and can be applied exhaustively in $\mathcal{O}(|A| \cdot n^2)$ time.*

**Proof** *Safeness:* Let $(G, A, k, x)$ be an instance of CNP-NDV for which $C$ is a critical node cut. Let $G'$ be the graph after Reduction Rule 5.2 is applied exhaustively on $G$. We show that a vulnerable connection exists in $G - C$, if and only if it exists in $G' - C$. Then, we conclude that the connectivity of $G - C$ is at most $x$, if and only if the connectivity of $G' - C$ is at most $x$. The safeness of Reduction Rule 5.2 follows.

Let $d$ be a vulnerable vertex in $G$ and $G'$ and let $\{d, v\}$ be a vulnerable connection in $G' - C$. Thus, in $G' - C$, we can find a path $d, u_1, \ldots, u_t, v$. In the following, we refer to $d$ as $u_0$ and to $v$ as $u_{t+1}$. Let $i$ be the integer such that $\{u_i, u_{i+1}\} \in E(G')$, but $\{u_i, u_{i+1}\} \notin E(G)$. Without loss of generality, the vertex $u_i \in K$ and $u_{i+1} \in N_G(K)$. It follows that there is a path $u_i, w_1, \ldots, w_\ell, u_{i+1}$ through $K$ and $\{w_\ell, u_{i+1}\} \in E(G)$. Then, there is a path in $G - C$ that connects $d$ and $v$, and $\{d, v\}$ is a vulnerable connection in $G - C$.

Let $\{d, v\}$ be a vulnerable connection in $G - C$. Because $G$ is a subgraph of $G'$, $\{d, v\}$ is also a vulnerable connection in $G' - C$.

*Running time:* Because we have at most $|A|$ connected components in $G[A]$, Reduction Rule 5.2 can be applied at most $|A|$ times. We then have to compute the neighbors of any subset of $V$ in $\mathcal{O}(m)$ time to add at most $|A| \cdot |V \setminus A|$ edges. Notice that, in Reduction Rule 5.1, some edges could have been added to $G$. Thus, the number of edges in $G$ could be a lot higher than in the original instance, but the number of edges is less than $n^2$. Thus, Reduction Rule 5.2 can be applied exhaustively in $\mathcal{O}(|A| \cdot n^2)$ time. □

In the following, we provide a reduction rule that bounds the number of vulnerable vertices that a non-vulnerable vertex can have as neighbors. This is the last reduction rule we provide to show the kernelization.

**Reduction Rule 5.4** *For every non-vulnerable vertex $v$ that has at least $\lceil \sqrt{2x}\, \rceil$ vulnerable neighbors, remove $v$ from $G$ and decrease $k$ by one.*

**Lemma 5.5** *For every instance of* CNP-NDV*, Reduction Rule 5.4 is safe and can be applied exhaustively in $\mathcal{O}(|V \setminus A| \cdot n^2)$ time.*

**Proof** *Safeness:* The safeness of Reduction Rule 5.4 follows from the observation that, if a vertex $v$ has $\lceil \sqrt{2x}\, \rceil$ vulnerable vertices as neighbors, then there are $\lceil \sqrt{2x} \rceil$ vulnerable connections $\{v, w\}$ with $w \in N(v)$ and $\binom{\lceil \sqrt{2x}\, \rceil}{2}$ vulnerable connections $\{w_1, w_2\}$ with $w_1, w_2 \in N(v)$. Altogether there are

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

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

*Running time:* Reduction Rule 5.4 can be applied at most $|V \setminus A|$ times, once for every non-vulnerable vertex. Every time this rule is applied, we need to find the neighborhood and count the number of vulnerable vertices in the neighborhood. This can be computed in $\mathcal{O}(m)$ time. Notice that, in Reduction Rule 5.1, some edges could have been added to $G$. Thus, the number of edges in $G$ could be a lot higher than in the original instance, but the number of edges is at most $n^2$. Thus, Reduction Rule 5.4 can be applied exhaustively in $\mathcal{O}(|V \setminus A| \cdot n^2)$ time. □

Now that we have all reduction rules, we can prove that CNP-NDV admits a kernelization with regard to the parameter $|V \setminus A| + x$. For that, we bound the number of vulnerable vertices by $|V \setminus A| \cdot \lceil \sqrt{2x} \rceil$.

**Theorem 5.6** CNP-NDV *admits a kernelization with $\mathcal{O}(|V \setminus A| \cdot \sqrt{x})$ vertices which can be computed in $\mathcal{O}(n^3)$ time.*

**Proof** Let $(G, A, k, x)$ be an exhaustively reduced instance regarding Reduction Rules 5.1, 5.2, and 5.4. We want to bound the number of vulnerable vertices. Because the instance is exhaustively reduced, it follows that in $G$ every vulnerable vertex has a neighbor in $|V \setminus A|$, because of Reduction Rules 5.1 and 5.2. Also, every non-vulnerable vertex has at most $\lceil \sqrt{2x} \rceil$ vulnerable neighbors after Reduction Rule 5.4 has been applied exhaustively. Thus, the number of vulnerable vertices is at most $|V \setminus A| \cdot (\lceil \sqrt{2x} \rceil)$. Therefore, we have the desired kernel that consists of at most $|V \setminus A| \cdot (\lceil \sqrt{2x} \rceil + 1) \in \mathcal{O}(|V \setminus A| \cdot \sqrt{x})$ vertices.

We have proven that the Reduction Rules 5.1, 5.2, and 5.4 can be applied exhaustively in $\mathcal{O}(n^3)$ time. Thus, the kernelization can be computed in $\mathcal{O}(n^3)$ time. □

## 5.2 Kernelization based on the Neighborhood-Diversity

In this section we consider the neighborhood-diversity. Recall that the neighborhood-diversity nd is the number of neighborhood-classes and that a neighborhood-class contains all vertices that have the same open or closed neighborhood. Further, recall that, due to Corollary 3.13, both problems CNP-V and CNP-NDV are fixed-parameter tractable with respect to the parameter neighborhood-diver-sity nd. Also, due to Corollaries 3.6 and 3.4 both problems CNP-V and CNP-NDV are fixed-parameter tractable with respect to the parameter $k+x$. In the following, we show that CNP-V and CNP-NDV admit a kernelization of polynomial size in $\text{nd} + k + x$.

In this entire section, we separate the neighborhood-classes into vulnerable vertices and vertices that are non-vulnerable. Afterwards, we have at most $2 \cdot \text{nd}$ neighborhood-classes. We start with a lemma to prove that this division can be computed in linear time. Then, in three reduction rules, we decrease the size of every neighborhood-class that contains more than $k+x+1$ vertices and thus we can bound the number of vertices in a reduced instance by $\mathcal{O}(\text{nd} \cdot (k+x+1))$.

**Lemma 5.7** *In $\mathcal{O}(n+m)$ time, we can compute all neighborhood-classes, divide them into vulnerable and non-vulnerable classes, and compute the size of the separated classes.*

**Proof** Hsu and Ma [HM91] showed that all neighborhood-classes can be computed in $\mathcal{O}(n+m)$ time. Recall that all vertices have a Boolean flag to check whether a vertex is vulnerable. Then, by iterating over the vertices of all neighborhood-classes, we can separate vulnerable from non-vulnerable vertices and we can compute the number of vulnerable or non-vulnerable vertices. □

In this first reduction rule, we bound the number of vertices of a neighborhood-class that consists only of non-vulnerable vertices.

**Reduction Rule 5.8** *If there is a neighborhood-class $K$ that is a subset of $V \setminus A$ and $K$ has more than $k+x+1$ vertices, then delete an arbitrary vertex from $K$.*

**Lemma 5.9** *For an instance of* CNP-V *or* CNP-NDV*, Reduction Rule 5.8 is safe and can be applied exhaustively in $\mathcal{O}(n+m)$ time.*

**Proof** *Safeness:* Let $(G, A, k, x)$ be an instance of CNP-V or CNP-NDV and suppose that there is a neighborhood-class $K$, which is a subset of $V \setminus A$ and $|K| > k+x+1$. Let $v$ be an arbitrary vertex of $K$. 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.

Let $(G - \{v\}, A, k, x)$ be a yes-instance of CNP-V or CNP-NDV. We define $K' = K \setminus \{v\}$. Then, there is a critical node cut $C$ for $(G - \{v\}, A, k, x)$. Assume towards a contradiction that $G' := (G - \{v\}) - C$ contains a vulnerable connection $\{d, u\}$ with $d \in A$ and $u \in K'$. It follows that there is a path $d, w_1, \ldots, w_t, u$ in $G'$. Because $\{w_t, u\} \in E(G')$, by the definition of neighborhood-classes, it is $\{w_t, u'\} \in E(G')$ for all $u' \in K' \setminus C$. Thus, for all $u' \in K' \setminus C$, there exists the vulnerable connection $\{d, u'\}$ in $G$. These are $|K' \setminus C| \geq x + k + 1 - k > x$. This is a contradiction to $C$ being a critical node cut. Thus, in $G - \{v\}$ the set $C$ separates $A$ and $K'$. It follows that $G - C$ has the same as the connectivity as $(G - \{v\}) - C$, and $(G, A, k, x)$ is a yes-instance of CNP-V or CNP-NDV.

Let $(G, A, k, x)$ be a yes-instance of CNP-V or CNP-NDV. Since the graph $G - \{v\}$ is an induced subgraph of $G$, a critical node cut for $(G, A, k, x)$ is also a critical node cut for $(G - \{v\}, A, k, x)$. Thus, $(G - \{v\}, A, k, x)$ is a yes-instance of CNP-V or CNP-NDV as well.

*Running time:* By Lemma 5.7, we compute the size of a neighborhood-class that is a subset of non-vulnerable vertices in $\mathcal{O}(n+m)$ time. Doing this, once we counted $k+x+1$ non-vulnerable vertices in a neighborhood-class $K$, we can remove all non-vulnerable vertices that we find in $K$ from the graph. By this, we count every vertex at most once. Thus, Reduction Rule 5.8 can be applied exhaustively in linear time. □

So far, we bounded the size of neighborhood-classes that consist of non-vulnerable vertices. In the remaining reduction rules, we bound the size of neighborhood-classes that consist of vulnerable vertices. For that, we need to distinguish between instances of CNP-V and instances of CNP-NDV and furthermore between neighborhood-classes that are critical cliques and neighborhood-classes that are critical independent sets. Observe that a neighborhood-class is either a critical clique or a critical independent set. Recall that a critical clique is a neighborhood-class in which all vertices are pairwise adjacent and a critical independent set is a neighborhood-class in which all vertices are pairwise not adjacent.

**Reduction Rule 5.10**

1. *Let $(G, A, k, x)$ be an instance of* CNP-NDV*. If $G$ contains a critical clique $K \subseteq A$ with at least $\lceil \sqrt{2x}\, \rceil + 2$ vertices, then return no.*

2. *Let $(G, A, k, x)$ be an instance of* CNP-V*. If $G$ contains a critical clique $K \subseteq A$ with at least $\lceil \sqrt{2x}\, \rceil + k + 2$ vertices, then return no.*

**Lemma 5.11** *For an instance of* CNP-V *or* CNP-NDV*, Reduction Rule 5.10 is safe and can be applied exhaustively in $\mathcal{O}(n+m)$ time.*

**Proof** *Safeness:* In CNP-NDV, a clique $K$ (in fact even a connected component) of vulnerable vertices causes $\binom{|K|}{2}$ vulnerable connections that cannot be removed. Thus, if $|K| \geq \lceil \sqrt{2x} \rceil + 2$, we can return no, since $\binom{|K|}{2} > x$. In CNP-V, we can add vulnerable vertices to a critical node cut $C$. Suppose, $G$ contains a critical clique $K \subseteq A$ of size at least $\lceil \sqrt{2x} \rceil + k + 1$. Because the size of $C$ is at most $k$, even $G[K \setminus C]$ contains at least $\binom{|K| - |C|}{2} > x$ vulnerable connections. Thus, we can return no.

*Running time:* By Lemma 5.7 we can compute the size of all neighborhood-classes in $\mathcal{O}(n + m)$ time. We can apply Reduction Rule 5.10 at most once per instance. Thus, Reduction Rule 5.10 can be applied exhaustively in linear time. □

The last reduction rule that we need bounds the size of critical independent sets that consist of vulnerable vertices. Analogous to Reduction Rule 5.10, we need to distinguish between instances of CNP-V and instances of CNP-NDV. We proved that the size of a critical clique $C$ can be bounded, because otherwise the number of vulnerable connections within $C$ would be more than $x$. In contrast, an independent set $I$ does not contain edges. However, it is useful that we know that all neighbors of a neighborhood-class are common. Thus, if only a single neighbor $v$ of $I$ is not in a critical node cut, all vertices of $I$ are pairwise connected over $v$. We use this observation for the next reduction rule. Recall that with $N[S]$ we denote the closed neighborhood of a vertex set $S$, which is $N(S) \cup S$.

**Reduction Rule 5.12** *1. Let $(G, A, k, x)$ be an instance of* CNP-NDV. *If a critical independent set $I \subseteq A$ with $|I| \geq \lceil \sqrt{2x} \rceil + 2$ exists such that $N(I) \cap A \neq \emptyset$, then return no. If a critical independent set $I \subseteq A$ with $|I| \geq \lceil \sqrt{2x} \rceil + 2$ exists such that $N(I) \cap A = \emptyset$, then decrease $k$ by $|N(I)|$ and remove $N[I]$ from the graph $G$.*

*2. Let $(G, A, k, x)$ be an instance of* CNP-V. *If a critical independent set $I \subseteq A$ with $|I| \geq \lceil \sqrt{2x} \rceil + k + 2$ exists, then decrease $k$ by $|N(I)|$ and remove $N[I]$ from the graph $G$.*

**Lemma 5.13** *For an instance of* CNP-V *or* CNP-NDV*, Reduction Rule 5.12 is safe and can be applied exhaustively in $\mathcal{O}(n + m)$ time.*

**Proof** *Safeness:* Let $(G, A, k, x)$ be an instance of CNP-NDV or CNP-V. Suppose $I \subseteq A$ is a critical independent set. If a vertex $v \in N(I)$ is not in a critical node cut $C$, then there are at least $\binom{|I \setminus C|}{2}$ vulnerable connections in the graph $G - C$: for every $u, w \in I \setminus C$ with $u \neq w$ form a vulnerable connection $\{u, w\}$ over the path $u, v, w$. For an instance of CNP-NDV we know $C \cap A = \emptyset$ and thus $C \setminus I = \emptyset$. It follows that, if $|I| \geq \lceil \sqrt{2x} \rceil + 2$ and $v \notin C$, we can return no. Thus, if $v \in A$, we can return no. If $v \notin A$, we can assume $v \in C$. If all neighbors of $I$ are in $C$, then $I$ consists of isolated vertices in $G - C$. Thus, we can also remove $I$. The reduction rule is correct.

For an instance of CNP-V we know $\binom{|I \setminus C|}{2} > x$ if $|I| \geq \lceil \sqrt{2x}\, \rceil + k + 2$. The rest of the argument can be proven analogous.

*Running time:* With help of Lemma 5.7 we can compute the size of all neighborhood-classes in $\mathcal{O}(n + m)$ time. We can apply Reduction Rule 5.10 at most once per neighborhood-class and at most $k$ times in total. After an application, we do not have to compute the size of the neighborhood-classes again, because they are either removed or still the same. Thus, Reduction Rule 5.10 can be applied exhaustively in linear time. $\square$

Now that we are familiar with these three reduction rules, we can prove that both of the problems CNP-V and CNP-NDV admit a kernelization with $\mathcal{O}(\mathrm{nd} \cdot (k + x))$ vertices. We would like to emphasize that we can apply every reduction rule exhaustively in linear time. Thus, we can compute the kernelization in linear time.

**Theorem 5.14** CNP-V *and* CNP-NDV *both admit a kernelization with* $\mathcal{O}(\mathrm{nd} \cdot (k + x))$ *vertices that can be computed within* $\mathcal{O}(n + m)$ *time.*

**Proof** With Reduction Rule 5.8, we can decrease the size of a neighborhood-class in $V \setminus A$ to contain at most $k + x + 1$ vertices. With Reduction Rule 5.10 and 5.12, we can bound the size of a neighborhood-class in $A$ to be at most $\lceil \sqrt{2x}\, \rceil + k + 2 < k + x + 1$. As $\mathrm{nd} \cdot (k + x + 1) \in \mathcal{O}(\mathrm{nd} \cdot (k + x))$ we follow that CNP-V and CNP-NDV admit a kernelization with at most $\mathcal{O}(\mathrm{nd} \cdot (k + x))$ vertices.

Observe that all three reduction rules can be performed in linear time. After we use the algorithm provided by Hsu and Ma [HM91], we can iterate over all vertices of all neighborhood-classes to execute all three reduction rules. Thus, we compute the kernelization in linear time. $\square$

## 5.3 Kernelization based on the vertex cover number

In this section, we show that CNP-V has a kernelization of polynomial size for the parameter $|A| + \mathrm{vc} + x$ and that CNP-NDV has a kernelization of polynomial size for the parameter $\mathrm{vc} + k + x$. By Corollary 3.9, CNP-V is fixed-parameter tractable with respect to $|A| + x$. Due to Corollary 3.14, CNP-V and CNP-NDV are fixed-parameter tractable with respect to the parameter vc. It follows that CNP-V is fixed-parameter tractable for the bigger parameter $|A| + \mathrm{vc} + x$ and CNP-NDV is fixed-parameter tractable with respect to the bigger parameter $\mathrm{vc} + k + x$. We strengthen this positive result in this section and show that CNP-V admits a polynomial kernel with respect to the parameter $|A| + \mathrm{vc} + x$ and CNP-NDV admits a polynomial kernel with regard to the parameter $\mathrm{vc} + k + x$.

Recall that a vertex cover $C$ of $G$ is a set of vertices $C \subseteq V(G)$, such that the induced subgraph $G - C$ does not contain an edge. Finding a vertex cover $C$ is NP-hard [Kar72]. However, we can compute a maximal matching and add all the incident vertices to a set $Z$ in polynomial time. Then, the size of $Z$ is at most $2 \cdot |C|$ vertices and $Z$ contains a vertex cover. Even though it is possible to find a better approximation than by factor 2 [Kar09, DLP13], it is unlikely that a much better approximation can be found [KR08].

For the rest of the section, we fix an instance $I := (G = (V, E), A, k, x)$ of CNP-V or CNP-NDV. Let $Z$ be a 2-approximation of vertex cover of $G$. Next, we define a subset $B$ of the vertices depending on the problem, which is either CNP-V or CNP-NDV. If we are dealing with CNP-V, we define a set $B := A \cup Z$. In case of CNP-NDV, we define $B := Z$. We name $B$ the *base*. By definition, $|B| \leq 2 \cdot \text{vc}$ when we deal with an instance of CNP-NDV and $|B| \leq |A| + 2 \cdot \text{vc}$ when we deal with an instance of CNP-V. All remaining vertices $Y := V \setminus B$ are then an independent set, because $B$ contains a vertex cover.

For the kernelization, we bound the size of $Y$. For this, we first remove isolated vertices from $Y$. Then, we can use the Expansion Lemma, which was introduced by Elena Prieto Rodríguez [Rod05], to bound the size of $Y$.

To show the claimed kernelizations, we start with a simple rule.

**Reduction Rule 5.15** *Remove isolated vertices from $G$.*

The correctness of the rule is obvious and it can be applied at most $|V|$ times. We can compute in linear time whether a vertex has a neighbor. Afterwards, we can iterate over all vertices and remove all vertices which do not have a neighbor. Thus, Reduction Rule 5.15 can be applied exhaustively in $\mathcal{O}(n + m)$ time.

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.

**Reduction Rule 5.16** *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 5.17** *For an instance of* CNP-NDV, *Reduction Rule 5.16 is safe and can be applied exhaustively in $\mathcal{O}(n^2)$ time.*

**Proof** *Safeness: Case 1:* We show that, if there is a critical node cut $C$ for $(G, A, k, x)$, 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 $|(N(v) \setminus \{d\}) \setminus C| \geq x$. Together with $\{d, v\}$ in $G - C$, there are more than $x$ vulnerable connections. We follow that $v \in C$.

*Case 2:* 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 5.16 removes a vertex, the reduction rule 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 in 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 Elena Prieto Rodríguez [Rod05]. We use the formulation by Cygan et al. [CFK$^+$15]:

---

Let $H$ be a bipartite graph with vertex bipartition $(C, D)$. For a positive integer $q$, a set of edges $M \subseteq E(H)$ is called a *$q$-expansion of $C$ into $D$*, if every vertex of $C$ is incident with exactly $q$ edges of $M$ and the edges in $M$ are incident with exactly $q \cdot |C|$ vertices in $D$.

Let $q \geq 1$ be a positive integer and $H$ be a bipartite graph with vertex bipartition $(C, D)$ such that $|D| \geq q \cdot |C|$ and there are no isolated vertices in $D$. Then, there exist nonempty vertex sets $X \subseteq C$ and $Y \subseteq D$ such that there is a $q$-expansion of $X$ into $Y$ and $N_H(Y) \subseteq X$. Furthermore, the sets $X$ and $Y$ can be found in time polynomial in the size of $H$.

---

Because 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 to 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 $(B, Y)$.

In the following, we assume that Reduction Rules 5.15 and 5.16 are exhaustively applied.

**Reduction Rule 5.18** *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 nonempty 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 5.19** *For an instance of* CNP-V *or* CNP-NDV*, Reduction Rule 5.18 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 Reduction Rule 5.15 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 nonempty 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 especially $A \cap Q = \emptyset$. In an instance of CNP-NDV, after Reduction Rule 5.16 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 $G$, every critical node cut $C$ for $(G, A, k, x)$ is also a critical node cut $C \setminus \{v\}$ for $(G - \{v\}, A, k, x)$.

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 connectivity of $(G - \{v\}) - C$ is the connectivity of $G - C$ and $C$ is also a critical node cut for $(G, A, k, x)$.

*Running time:* Every time Reduction Rule 5.16 is applied, a vertex is deleted. Thus, the reduction rule can be applied at most $|Y| \leq n$ times. From the Expansion Lemma, we know that we can find nonempty 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. As we can remove an arbitrary vertex of $Q$, this step can be done in linear time. Thus, we can apply Reduction Rule 5.18 exhaustively in polynomial time. In this computation, the bottle neck is the time to compute the vertex sets $P$ and $Q$ from the Expansion Lemma. □

Now that we have presented our reduction rules, we can bound the size of the kernelization.

**Theorem 5.20** *An instance $(G, A, k, x)$ of* CNP-V *or* CNP-NDV *contains less than $|B| \cdot (k + x + 3)$ vertices after Reduction Rules 5.15, 5.16, and 5.18 have been applied exhaustively.*

**Proof** Let $(G, A, k, x)$ be an instance that is reduced exhaustively by the Reduction Rules 5.15, 5.16, and 5.18.

Because Reduction Rule 5.18 has been applied exhaustively, 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. □

Since $|B| \leq |A| + 2 \cdot \mathrm{vc}$ for CNP-V and $|B| \leq 2 \cdot \mathrm{vc}$ for CNP-NDV, we conclude that:

**Corollary 5.21** *For an instance $(G, A, k, x)$ of* CNP-V*, we can compute a kernelization with less than $(|A| + 2 \cdot \mathrm{vc}) \cdot (k+x+3)$ vertices in polynomial time.*

**Corollary 5.22** *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.*

# Chapter 6

# Discussion

In this work, with Critical Node Problem with Vulnerable Nodes (CNP-V) we presented a problem for which the well-studied Critical Node Problem (CNP) is a special case. With Critical Node Problem with Non-Deletable Vulnerable Nodes (CNP-NDV) we also presented a related problem. We analyzed both problems within the framework of parameterized complexity.

## 6.1 Summary

We showed that CNP-V is fixed-parameter tractable with respect to the parameters $k+x$, the neighborhood-diversity nd, and $y$ which is the number of vulnerable connections to be removed. We showed that CNP-NDV is fixed-parameter tractable with respect to the parameters $k+x$, the neighborhood-diversity nd, and the number of non-vulnerable vertices. To the best of our knowledge, the fixed-parameter algorithms for the problem CNP-V with parametrization $k+x$ or $y$ are as fast as the previously best known fixed-parameter algorithms for the problem CNP with the same parametrization. CNP-NDV parameterized with $k+x$ can be solved even faster, than CNP with the same parametrization. If anyone wants to implement one of our algorithms, we advise to implement the fixed-parameter algorithms with parametrization $k+x$.

Furthermore, we showed that it is unrealistic that CNP-NDV is fixed-parameter tractable with respect to the parametrization $k+y$, even if the input graph is bipartite or a split graph and contains only one vulnerable vertex.

## 6.2 Outlook

As we have not covered all possible parametizations, the task remains to identify further parameters for which CNP-V or CNP-NDV are fixed-parameter tractable with respect to not yet considered parametrizations. Especially, the question whether CNP-V is fixed-parameter tractable with respect to the parameter $|A|$ remains open; at the moment we only know that CNP-V with respect to the parameter $|A|$ is in the class slice-wise polynomial.

Also, natural questions that remain open are whether CNP-V or CNP-NDV admit a polynomial size kernelization for specific parameters. As we have not shown if CNP-V or CNP-NDV admit a kernel of polynomial size in $k + x$, this is an important question to consider. Remark, that all kernelizations that we provided were for parameters that are greater than $k + x$. We know that, unless coNP $\subseteq$ NP/poly, CNP-NDV does not admit a kernelization of polynomial size with respect to the parameter $k + y + \omega$, where $\omega$ is the treewidth of the input graph. However, it remains an open question whether CNP-NDV is W[1]-hard with respect to $k + y + \omega$, as we expect.

In the versions we presented, a connected pair caused one vulnerable connection, if and only if one of the vertices was vulnerable. It is thinkable to generalize the cost for connected pairs. Further problems that can be examined are more general weighted problems of CNP where the deletion of vertices is also weighted. It follows directly from results of Guo et al. [GHK$^{+}$08] that the single weighted version is para-NP-hard for the parameters $p + x$ and $\omega + x$. Di Summa et al. [DSGL11] proved that the single weighted version is NP-hard, even on trees.

Furthermore, these two new problems can also be analyzed within the frameworks of Integer Linear Programming. Also, heuristics and approximations for CNP-V or CNP-NDV can be researched.

# Bibliography

[ACEP09] Ashwin Arulselvan, Clayton W Commander, Lily Elefteriadou, and Panos M Pardalos. Detecting critical nodes in sparse graphs. *Computers & Operations Research*, 36(7):2193–2200, 2009.

[ADSG13] Bernardetta Addis, Marco Di Summa, and Andrea 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.

[AGH16] Roberto Aringhieri, Andrea Grosso, and Pierre Hosteins. A genetic algorithm for a class of critical node problems. *Electronic Notes in Discrete Mathematics*, 52:359–366, 2016.

[BC09] Vladimir Boginski and Clayton W Commander. Identifying critical nodes in protein-protein interaction networks. In *Clustering challenges in biological networks*, pages 153–167. World Scientific, 2009.

[CFK$^+$15] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer, 2015.

[CHBA03] Reuven Cohen, Shlomo Havlin, and Daniel Ben-Avraham. Efficient immunization strategies for computer networks and populations. *Physical review letters*, 91(24):247901–1–247901–4, 2003.

[Die17] Reinhard Diestel. *Graph Theory.* Springer, 2017.

[DLP13] François Delbot, Christian Laforest, and Raksmey Phan. New approximation algorithms for the vertex cover problem. In *Proceedings of the 24th International Workshop on Combinatorial Algorithms (IWOCA '13)*, volume 8288 of *Lecture Notes in Computer Science*, pages 438–442. Springer, 2013.

[DSGL11] Marco Di Summa, Andrea Grosso, and Marco Locatelli. Complexity of the critical node problem over trees. *Computers & Operations Research*, 38(12):1766–1774, 2011.

[DSGL12] Marco Di Summa, Andrea Grosso, and Marco Locatelli. Branch and cut algorithms for detecting critical nodes in undirected graphs. *Computational Optimization and Applications*, 53(3):649–680, 2012.

[FGK13] Fedor V Fomin, Petr A Golovach, and Janne 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)*, pages 421–432. Springer, 2013.

[GHK$^+$08] Jiong Guo, Falk Hüffner, Erhan Kenar, Rolf Niedermeier, and Johannes Uhlmann. Complexity and exact algorithms for vertex multicut in interval and bounded treewidth graphs. *European Journal of Operational Research*, 186(2):542–553, 2008.

[GJS76] Michael R Garey, David S Johnson, and Larry Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[GLZ21] Xiucui Guan, Chao Liu, and Qiao Zhang. The critical node problem based on connectivity index and properties of components on trees. *Asia-Pacific Journal of Operational Research*, 38(1):2050041:1–2050041:27, 2021.

[HKKN16] Danny Hermelin, Moshe Kaspi, Christian Komusiewicz, and Barak Navon. Parameterized complexity of critical node cuts. *Theoretical Computer Science*, 651:62–75, 2016.

[HM91] Wen-Lian Hsu and Tze-Heng Ma. Substitution decomposition on chordal graphs and applications. In *Proceedings of the 2nd International Symposium on Algorithms (ISA '91)*, volume 557 of *Lecture Notes in Computer Science*, pages 52–60. Springer, 1991.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[Kar09] George Karakostas. A better approximation ratio for the vertex cover problem. *ACM Trans. Algorithms*, 5(4):41:1–41:8, 2009.

[KR08] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-epsilon. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

[Lam12] Michael Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012.

[LTK16] Mohammed Laslou, Mohammed Amin Tahraoui, and Hamamache Kheddouci. Component-cardinality-constrained critical node problem in graphs. *Discrete Applied Mathematics*, 210:150–163, 2016.

[NCH20] Adel Nabli, Margarida Carvalho, and Pierre Hosteins. Complexity of the multilevel critical node problem. *CoRR*, abs/2007.02370, 2020.

[Nie06] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press, 2006.

[NSB16] Joe Naoum-Sawaya and Christoph Buchheim. Robust critical node selection by benders decomposition. *INFORMS Journal on Computing*, 28(1):162–174, 2016.

[Rod05] Elena Prieto Rodríguez. *Systematic kernelization in FPT algorithm design.* PhD thesis, The University of Newcastle, 2005.

[SW13] Manuel Sorge and Mathias Weller. The graph parameter hierarchy. 2013.

[VA14] Mario Ventresca and Dionne Aleman. A derandomized approximation algorithm for the critical node detection problem. *Computers & Operations Research*, 43:261–270, 2014.

[VHOB18] Mario Ventresca, Kyle Robert Harrison, and Beatrice M Ombuki-Berman. The bi-objective critical node detection problem. *European Journal of Operational Research*, 265(3):895–908, 2018.

[ZH17] Yangming Zhou and Jin-Kao Hao. A fast heuristic algorithm for the critical node problem. In *Proceedings of the 19th Genetic and Evolutionary Computation Conference Companion (GECCO '17)*, pages 121–122. ACM, 2017.