FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Distance Edge Monitoring Set Problem with Respect to Structural Parameters
Student:	Václav Lepič
Supervisor:	RNDr. Ondřej Suchý, Ph.D.
Study program:	Informatics
Branch / specialization:	Computer Science
Department:	Department of Theoretical Computer Science
Validity:	until the end of summer semester 2022/2023

Instructions

Get familiar with the Distance Edge Monitoring Set problem and the most important known results about it.
Get familiar with the basic notions and ideas of Parameterized Complexity.

Develop a parameterized algorithm for the problem with respect to the size of the minimum vertex cover, the minimum size of a feedback edge set,
or the minimum size of a feedback vertex set of the input graph or find major obstacles in developing such algorithms.

After consulting with the supervisor select one of the algorithms for the problem and implement it in a suitable language.

Test the resulting program on suitable data, evaluate its performance.

References:
Florent Foucaud, Shih-Shun Kao, Ralf Klasing, Mirka Miller, Joe Ryan: Monitoring the edges of a graph using distances,
Discrete Applied Mathematics, Elsevier, 2021.

Electronically approved by doc. Ing. Jan Janoušek, Ph.D. on 13 September 2021 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Distance Edge Monitoring Set Problem with Respect to Structural Parameters

Václav Lepič

Department of Theoretical Computer Science
Supervisor: RNDr. Ondřej Suchý, Ph.D.

May 12, 2022

Acknowledgements

I would like to thank my supervisor RNDr. Ondřej Suchý, Ph.D. for his advice and patience. I would also like to thank Lucka, Soptík and Dex for their moral support and help.

Declaration

In Prague on May 12, 2022 …………………

Citation of this thesis

Lepič, Václav. *Distance Edge Monitoring Set Problem with Respect to Structural Parameters.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2022.

Abstract

This thesis is about the distance edge monitoring set problem. The problem is first described and then approached from a point of structural parameters.

Algorithm parameterized by vertex cover number was proposed and its correctness was proven. The algorithm was then implemented and tested.

Algorithm parameterized by the feedback edge set number was proposed, implemented, and tested.

Keywords Edge Monitoring, Distances, Vertex Cover, Feedback Edge Set Number, Parameterized Complexity

Abstrakt

Tato práce se zabývá problémem monitorování hran pomocí vzdáleností.Problém je nejdříve popsán a poté je k němu přistoupeno z pohledu strukturálních parametrů.

Algoritmus parametrizovaný velikostí vrcholového pokrytí byl navržen a jeho korektnost byla dokázána. Následně byl implementován a jeho rychlost byla otestována.

Algoritmus, jehož parametrem je velikost feedback edge set byl navržen, a následně implementovaný a otestovaný.

Klíčová slova Monitorování hran, vzdálenosti, vrcholové pokrytí, Parametrizovaná složitost

Contents

List of Figures

List of Tables

Introduction

The Distance Edge Monitoring Set problem is a new concept of network monitoring using distance probes. The motivation is to detect a network connection failure by measuring graph distance from a set of vertices to any other network vertex. The idea is that this distance should change if any connection fails.

The thesis summarizes the Distance Edge Monitoring Set problem and the most important known results. The thesis aims to develop a parameterized algorithm for the problem with respect to the size of the minimum vertex cover and the minimum size of a feedback edge, implement these, and test and evaluate their performance.

In the first chapter, some basic terminology is defined, and preliminaries are given. Also, we formally define the Distance Edge Monitoring Set problem.

The second chapter summarizes known results, gives basic lemmas about the problem and presents the basic algorithm for the problem and the way of approximating the problem.

The next chapter offers basic properties of the problem used in algorithms presented in the fourth and fifth chapters.

The fourth chapter introduces an algorithm for the distance edge monitoring set problem parameterized by the size of the minimum vertex cover of the input graph. The chapter also presents the implementation of the algorithm and evaluates the its performance.

Chapter five describes the algorithm parameterized by feedback edge set number, presents the implementation of the algorithm, and shows the results of the performance testing of the said algorithm.

Problem Description

In this thesis, we will take a look at the DISTANCE EDGE MONITORING SET problem, which was introduced by Foucaud et al. [1]. This problem consists in finding a set of vertices which can be used to detect edge failures.

Detecting edge failures is done by measuring the distance between vertices, at least one of which is in our set. For the distance between vertices u and v to change after removing edge e, e has to belong to every shortest path between u and v. Finding any distance edge monitoring set is relatively simple; a trivial example would be the set of all vertices. However, finding the smallest possible distance edge monitoring set is more challenging. In fact, Foucaud et al. [1] have already shown it to be NP-complete.

In this thesis, we will approach this problem with regards to some structural parameters. The goal is to design an algorithm, where the non-polynomial part is related to the parameter and not the size of the entire graph.

The first parameter we will take a look at is the vertex cover number of the input graph. a vertex cover of a graph is by itself a distance edge monitoring set, though it does not have to be the smallest one. Another parameter we will consider is the feedback edge set number, where there is already known an upper bound related to that [1].

Chapter 1

Preliminaries

In this chapter we first define terms from graph theory that will be used later in this thesis and then we will provide formal definition of the Distance Edge Monitoring Set problem.

1.1 Graph Theory Definitions

First, we start by defining what the graph is.

Definition 1 (inspired by Gross, Yellen [2])**.** All graphs considered for the purposes of this thesis are simple, i.e., multi-edges and self-loops are not permitted. A *graph* $G = (V, E)$ consists of sets V and E.

- V is a set of *vertices*, also referred to as $V(G)$, when there are multiple graphs.
- E is a set of *edges*, also $E(G)$, when there are multiple graphs.
- Each edge consists of exactly two vertices, which are called *endpoints*. An edge *joins* its endpoints.
- Vertices are *adjacent* if and only if there is an edge that joins them.
- Adjacent vertices may be called *neighbours*, set of all neighbours of a vertex v is called *neighbourhood* and denoted $N(v)$.
- The *degree* of a vertex v is the size of its neighbourhood.
- An edge is *incident* to a vertex v, if and only if v is one of its endpoints.

Since we will be utilizing the fact that monitored edges have to lie on every shortest path, we need a definition of the path and its length.

Definition 2 (inspired by Gross, Yellen [2])**.** A *path* in a graph G is an alternating sequence of vertices and edges $P = v_0, e_1, v_1, e_2, \ldots, e_n, v_n$, where for each $j \in \{1, \ldots, n-1, n\}$, and v_{j-1}, and v_j are endpoints of e_j, and no vertex is repeated in the sequence.

- Vertex v_0 is the *initial vertex.*
- Vertex v_n is the *terminal vertex.*
- If a vertex is neither initial nor terminal it is called an *internal vertex.*
- a u-v-path is a path with initial vertex u and terminal vertex v
- The *length* of a path u-v-path is the number of edges of such path.
- The length of a shortest path with u as initial and v as terminal vertex is called distance of u and v and denoted $d(u, v)$.

Since we will be talking about graphs with a vertex or a set of vertices removed, it will be helpful to have a notation for it.

Definition 3. Given graph G and $v \in V(G)$, let S be the set of edges incident to v, we define $G - v := (V(G) \setminus v, E(G) \setminus S)$.

Definition 4. Given graph G, and $S \subseteq V(G)$, let E_S be the set of all edges incident to any vertex from S, let $G - S := (V(G) \setminus S, E(G) \setminus E_S)$.

Even though we will be talking primarily about connected graphs, it is essential to define what it means for a graph to be connected or disconnected.

Definition 5 (West [3])**.** A graph G is *connected* if it has a u-v-path for every $u, v \in V(G)$ (otherwise, G is *disconnected*). If G has a u-v-path, then u is connected to v in G. The connection relation on $V(G)$ consists of the ordered pairs (u, v) such that u is connected to v.

The *components* of a graph G are its maximal *connected* subgraphs. a *component* (or a graph) is trivial if it has no edges; otherwise it is nontrivial. An isolated vertex is a vertex of degree 0.

Vertex cuts are crucial for our problem because they divide the graph into multiple components, where every path going from one component to another component has to pass through the cut.

Definition 6 (West [3])**.** A *vertex cut* of a graph G is a set $S \subseteq V(G)$ such that $G-S$ has more than one connected component. The *connectivity* of G, denoted $k(G)$, is the minimum size of a vertex set S such that $G - S$ is disconnected or has only one vertex. a graph G is *k-connected* if its connectivity is at least k.

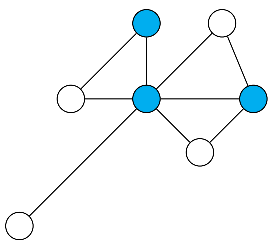

Figure 1.1: An example of a vertex cover. Vertices of the cover are highlighted in blue.

Like vertex cuts, edge cuts are essential for this problem because they split the graph into components, where every path from one component to another has to go through the cut.

Definition 7 (West [3])**.** A *disconnecting set* of edges is a set $F \subseteq E(G)$ such that $G - F$ has more than one connected component. a graph is k-*edge-connected* if every disconnecting set has at least k edges. The edge-connectivity of G, denoted $k'(G)$, is the minimum size of a disconnecting set (equivalently, the maximum k such that G is *k-edge-connected*).

Given $S, T \subseteq V(G)$, we write $[S, T]$ for the set of edges having one endpoint in S and the other in T. An *edge cut* is an edge set of the form $[S, \overline{S}]$, where S is a nonempty proper subset of $V(G)$ and $\overline{S}$ denotes $V(G) \setminus S$.

If this set contains only one edge, we call this edge a *bridge*.

Definition 8 (West [3])**.** A *clique* in a graph G is a set of pairwise adjacent vertices. The maximum size of a clique in G is a *clique number*, denoted $\omega(G)$. An *independent set* in a graph is a set of pairwise nonadjacent vertices. The maximum size of an independent set is an *independence number*.

1.2 Other Related Problems

One of the problems we will utilize is the VERTEX COVER problem. It consists in finding a set of vertices such that every edge is incident to at least one of these vertices. It is known, that vertex cover monitors all of the edges, though it does not have to be the smallest set to do so [1].

Definition 9 (Inspired by Cygan et al. [4])**.** A *vertex cover* of a graph G is a set $S \subsetneq V(G)$, such that $G - S$ is an independent set. Size of the smallest possible vertex cover is called *vertex cover number*.

In other words, this means that every edge of G has at least one of its endpoints in S.

Another thing we will use is the SET COVER problem. This problem consists in finding the sets, the union of which covers the universe of these sets.

In case of our problem, we will deal with the case where the universe is the edges of a graph, and we need to cover them with sets of edges monitored by individual vertices.

Definition 10 (Cygan et al. [4])**.** Let F be a family of sets over a universe U. For a subfamily $F' \subseteq F$ and a subset $U' \subseteq U$, we say that F' covers U' if every element of U' belongs to some set of F', that is, $U' \subseteq \bigcup F'$. In the Set Cover problem, we are given a family of sets F over a universe U and a positive integer k, and the task is to check whether there exists a subfamily $F' \subseteq F$ of size at most k such that F' covers U.

1.3 Parameterized Problems

Since this thesis looks at the problem with regard to structural parameters, we need to actually define, what it means for the problem to be parameterized.

Definition 11 (Cygan et al. [4])**.** A parameterized problem is a language $L \subseteq \Sigma^* \times N$, where Σ is a fixed, finite alphabet. For an instance $(x, k) \in \Sigma^* \times N$, k is called the parameter.

A fixed-parameter algorithm is an algorithm for a problem such that the complexity is polynomial with regards to the size of the problem, though it can be much worse, e.g., exponential, with regards to the parameter. a typical example of such a parameter is the size of the solution, though Foucaud et al. [1] have noted that the *distance edge monitoring set* problem is unlikely to be fixed parameter tractable with regards to the solution size.

Definition 12 (Cygan et al. [4])**.** A parameterized problem $L \subseteq \Sigma^* \times N$ is called *fixed-parameter tractable* (*FPT*) if there exists an algorithm A, called a *fixed-parameter algorithm*, a computable function $f : N \to N$, and a constant c such that, given $(x, k) \in \Sigma^* \times N$, the algorithm A correctly decides whether $(x, k) \in L$ in time bounded by $f(k) \cdot |(x, k)|^c$. The complexity class containing all fixed-parameter tractable problems is called *FPT*.

We will use *data reduction rules.* These rules allow us to reduce the problem into a smaller instance of the same problem and do that in polynomial time. However, it does not have to be able to reduce it into a problem of any arbitrary size.

Definition 13 (Cygan et al. [4])**.** A *data reduction rule*, or simply, *reduction rule*, for a parameterized problem Q is a function ϕ: $\Sigma^\star \cdot N \to \Sigma^\star \cdot N$ that maps an instance (I, k) of Q to an equivalent instance (I_0, k_0) of Q such that ϕ is computable in time polynomial in $|I|$ and k. We say that two instances of Q are equivalent if $(I, k) \in Q$ if and only if $(I_0, k_0) \in Q$.

1.4 Formal Definition of The Problem

Our goal in this problem is to find the smallest set of vertices that would monitor all of the edges of a graph.

First, we need to define what it means for an edge to be monitored by a vertex u. We want a distance from u to some other vertex v to change after removing the edge; thus, the edge must be part of every shortest path between u and v.

Definition 14 (Inspired by Foucaud et al. [1])**.** An edge e is *monitored* by a vertex u if and only if there is another vertex v such that e is a part of every shortest path between u and v.

Set of all edges, that are monitored by a vertex v is called $EM(v)$. For a set $S \subseteq V(G)$ we define $EM(S)$ as set of all edges that are monitored by any vertex from S, i.e., $EM(S) = \bigcup_{x_i \in S} EM(x_i)$.

Note that every vertex monitors its incident edges.

We consider G to be monitored by some set M when every edge of G is monitored by M. Since we are often interested in the size of the distance edge monitoring set, we define it as $dem(G)$.

Definition 15 (Foucaud et al. [1])**.** A set of vertices M is a *distance edge monitoring set* of a graph G if and only if $E(G) = EM(M)$, i.e., every edge is monitored by at least one vertex from M.

Degree of edge monitoring, denoted $dem(G)$ is the smallest possible size of a *distance edge monitoring set* of G.

Chapter 2

Known Results

In this chapter we will look at what is known about the problem. Since the problem was introduced in the year 2021 by Foucaud et al. [1] and there do not appear to be any new articles dealing with the problem since then, we summarize the parts of that article that are important for this thesis.

The following lemma shows that we can find out if the edge has failed and which edge has failed. This is due to the fact that no two edges have the same sets of pairs of vertices, where at least one of them is a part of the edge monitoring set such that the edge would belong to every shortest path between them.

Lemma 16 (Foucaud et al. [1])**.** Let M be a distance edge monitoring set of a graph G, and let $e_1, e_2 \in E(G)$ be two distinct edges. Let S_1 be the set of pairs of vertices (x, y), $x \in M, y \in V(G)$ such that e_1 belongs to every path between x and y and let S_2 be the set of pairs of vertices (x, y), $x \in M, y \in V(G)$ such that e_2 belongs to every shortest path between x and y, then $S_1 \neq S_2$.

Trees are easy to monitor since they can be monitored by one vertex. However, it means that they are not interesting for the purposes of this thesis.

Lemma 17 (Foucaud et al. [1])**.** For a graph G, $dem(G) = 1$ if and only if G is a tree.

It is not even necessary for the entire graph to be a tree since the edge being a bridge is enough for it to be monitored by any vertex. See Figure 2.1 for illustration.

Lemma 18 (Foucaud et al. [1])**.** Let G be a connected graph and let e be a bridge of G. For any vertex x of G, we have $e \in EM(x)$.

It has been proven that for an edge to be monitored by a vertex, the incident vertex that is further away can not be adjacent to more than one vertex at the same distance as the closer one.

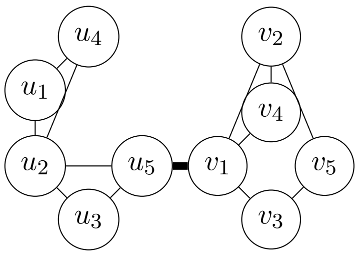

Figure 2.1: As described in Lemma 18, a bridge is monitored by any vertex in a connected graph. Every path from vertices $u_1, \ldots, u_5$ to v_1 passes trough (u_5, v_1), and every path from vertices $v_1, \ldots, v_5$ to u_5 passes trough (u_5, v_1).

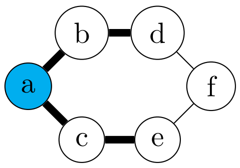

Figure 2.2: Highlighted edges are monitored by vertex a (highlighted in blue). Edges (d, f) and (e, f) are not monitored by a, because vertex f is adjacent to two vertices at a distance of 2 from vertex a.

Lemma 19 (Foucaud et al. [1])**.** Let x be a vertex of a connected graph G. Then, an edge (u, v) belongs to $EM(x)$ if and only if $d(x, u)$ and v is the only neighbour of u at a distance $i - 1$, from x.

Another trivial example would be monitoring an incident edge since the path of length 1 is always unique.

Lemma 20 (Foucaud et al. [1])**.** Let G be a graph and x a vertex of G. Then, for any edge e incident with x, we have $e \in EM(x)$.

One of the consequences of this lemma is that a *vertex cover* is an example of a *distance edge monitoring set.*

Every vertex monitors its incident edges, but it also usually monitors some other edges unless the following condition is met.

Lemma 21 (Foucaud et al. [1])**.** Let G be a connected graph and let x be its vertex. The following two conditions are equivalent.

- $EM(x)$ is the set of edges incident with x.
- For every vertex y of G with $y \in V(G) \setminus (\{x\} \cup N(x))$, there exist two shortest paths from x to y sharing at most one edge, the one incident with x.

2.1 General Algorithm by Foucaud et al.

Foucaud et al. [1] have provided relatively simple algorithm for finding the *distance edge monitoring set.* It works as follows:

See Algorithm 1 for illustration.

Algorithm 1

```
for $v \in V(G)$ do
    for $e \in E(V)$ do
        if Removing $e$ changes the distance from $v$ to endpoint of $e$ then
            Add $e$ to $EM(v)$
        end if
    end for
end for
Find set cover of these edge monitoring sets
```

The authors have also provided $\ln(|E(G)|+1)$ approximation, which works by approximating the set cover.

In this thesis, we will attempt to improve this algorithm using reduction rules.

Chapter 3

Basic Properties

In this chapter, we will look at a relations of distance edge monitoring set and graph structure.

There might be a vertex on every shortest path between 2 other vertices in some graphs. The following lemma describes what this means for the set of edges monitored by that vertex.

Lemma 22. For $u, v \in V(G)$ and $e \in E(G)$. If there is a $w \in V(G)$ such that every shortest path with u as its initial vertex and v as its terminal vertex contains both e and v, then $e \in EM(w)$.

Proof. Since w is in every shortest path between u, v, we can infer that $d(u, v) = d(u, w) + d(w, v)$. Removing e from a graph changes $d(u, v)$, thus changing either $d(u, w)$ or $d(w, v)$. Since removing the edge changes the distance, it must be a part of every shortest path. Therefore e is monitored by v. $\square$

We also know that for an edge to be monitored by some vertex v, it has to be part of every shortest path from that vertex to the endpoint further away from v. This is mostly just a weaker version of the Lemma 19, where it is unnecessary to know the distance to endpoints of the incident vertices.

Lemma 23. If a vertex $u \in V(G)$ monitors an edge $e = (v, w)$, e is part of every shortest path from u to v, or or it is part of every shortest path form u to w.

Proof. By Lemma 19, for an edge e to be monitored by a vertex w, the distance $d(u, v)$ has to differ from $d(u, w)$ by exactly one. Without loss of generality suppose that distance u, v is lower. No shortest path between u and v contains the edge (v, w), since the length of that path would be at least $d(u, w) + 1 = d(u, v) + 2$, thus for the Lemma 23 to hold, e has to be a part of every shortest path between u and w. Since e is monitored by u, according to Lemma 19

v is the only vertex adjacent to w that is closer to u, e has to belong to every shortest path from u to w. □

Since not every graph has to be connected we should consider what that means for the distance edge monitoring set. It turns out that we can consider each of the components separately.

Lemma 24. If G is not connected, then let C_G be the set of its connected components. We have $dem(G) = \sum_{G_i \in C_G} dem(G_i)$.

Proof. For any vertex $u \in G_1$ and let $e = (v_1, v_2)$ be any edge in G_2. According to Lemma 19 for v to monitor e the distance $d(u, v_1)$ would have to differ from $d(u, v_2)$ by 1, but since u is in a different connected component than v_1 and v_2 these distances are undefined, therefore v does not monitor e. □

Vertex cuts are also interesting for distance edge monitoring. Any vertex on either side of a given vertex cut monitors at most as much on the other side of a given cut as the vertex cut itself.

Lemma 25. Let C be a vertex cut of a graph G. Let A be one of connected components of $G \setminus C$ and let $B = (G \setminus C) \setminus V(A)$. Then $EM(A) \cap E(B) \subseteq EM(C) \cap E(B)$, in other words, all edges from B monitored by A are also monitored by C.

Proof. Assume that there is an edge $e \in E(G)$ monitored by vertex $v \in V(A)$ not monitored by any vertex from C. Let us say that e is part of every shortest path between v and $u \in V(B)$. Since C is a vertex cut that would mean that every such path has to include at least 1 vertex $v_c \in C$ (there can be multiple such vertices). Since every shortest path from v to u trough v_c contains e which means every shortest path from v_c to u also contains e. □

As expected, sets of edges monitored by vertices with the same neighbourhoods are similar, though not always identical. In fact, they can differ by their incident edges only.

Lemma 26. Let u and v be two vertices with identical neighbourhoods, then $EM(u)$ and $EM(V)$ can differ only by edges incident to u and v.

Proof. For any two vertices $u, v \in V(G)$, such that $N(u) = N(v)$ and $w \in V(G)$ is not adjacent to either of u, v. Let $e \in EM(U)$ be an edge incident to w, such that w is the vertex with a larger distance from u. Since u and v have the same neighbourhood, distance from u and v to any other vertex is the same, thus there is also only one vertex at a distance $d(u, w) - 1 = d(v, w) - 1$ (the one incident to e), therefore according to Lemma 19 v monitors e. □

In case there are multiple vertices with the same neighbourhood, unless these vertices are of degree 1, we know that these vertices do not monitor edges incident to other vertices with the same neighbourhood. If these were of degree 1, we can deal with them later with Reduction Rule 31.

Lemma 27. If there is an independent set $S \subseteq V(G)$, where every vertex from S has the same neighbourhood N, and $|N| \geq 2$, then there is no vertex $u \in S$ that would monitor any edge incident to any $v \in S \setminus u$.

Proof. Let $u_1, u_2 \in S$ and $v_1, v_2 \in N$, see Figure 3.1 for illustration. According to Lemma 19 for any edge e incident to u_2 be monitored by u_1 there would have to be exactly one vertex incident to u_2 at a distance $d(u_1, u_2) - 1 = 1$. However, there are at least 2 such vertices, namely v_1, v_2, therefore u_1 does not monitor any edge incident to u_2. In other words, no vertex monitors edges incident to other vertices with the same neighbourhood, unless there is only 1 vertex in that neighbourhood. □

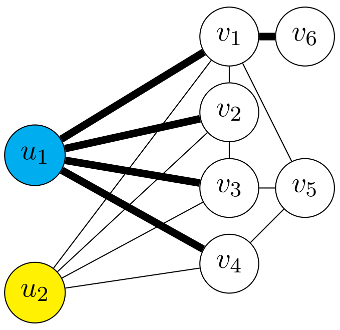

Figure 3.1: Illustration for Lemma 27. No edges monitored by vertex u_1 (highlighted by thick line) can be incident to u_2.

Sometimes there is an independent set of vertices with the same neighbourhood. The following lemma states that there can not be an arbitrary number of vertices from this set in a minimal distance edge monitoring set. Possibilities are limited to all, one or none of the vertices from that set belonging to any minimal distance edge monitoring set.

Note that the case, where $|N| = 1$ would mean that any edge between S and N is a bridge and therefore monitored by any vertex in its connected component.

Lemma 28. If there is an independent set $S \subseteq V(G)$, where every vertex from S has the same neighbourhood N, and $|N| \geq 2$, then for every smallest distance edge monitoring set M, we have either $S \cap M = \emptyset$, $S \cap M = S$, or $|S \cap M| = 1$.

Proof. Due to Lemma 26 we know that any vertex $v_1 \in S$ monitors the same set of edges as any other vertex $v_2 \in S$ except those incident to v_1, thus adding v_2 into monitoring set M adds only edges incident to v_2. This means that for more than 2 but less than all vertices from S to be a part of a minimal distance edge monitoring set it would mean that there is a vertex $u_1 \in S$ that has all of its incident edges monitored, while $u_2 \in S$ does not, i.e., there is an edge $(u_2, v_n), v_n \in N$ which is not monitored. Since u_1 and u_2 have the same neighbourhoods there has to be an edge (u_1, v_n) monitored by some vertex $x \in V(G)$. According to Lemma 19 that means that v_n is the only vertex at a distance $d(u_1, x) - 1$ from x adjacent to u_1. Since u_2 has the same neighbourhood as u_2, v_n is the only vertex at a distance $d(u_1, x) - 1 = d(u_2, x)$ from x adjacent to u_2, thus x monitors (u_2, v_n). Therefore either all edges incident to vertices from S are monitored or every vertex from S that is not a part of a distance edge monitoring set has at least 1 incident edge that is not monitored, hence the minimal distance edge monitoring set contains either 0, 1 or all vertices from S.

□

This is quite useful, though we can improve on this even further. The following lemma rules out the possibility of the entire set S being part of a minimal distance edge monitoring set in case it is too big.

Lemma 29. If there is an independent set $S \subseteq V(G)$, with the same neighbourhood N and $|S| > |N|$, then every smallest edge monitoring set contains at most one vertex from S.

Proof. According to Lemma 28 there are either 0, 1 or all vertices from S in every minimal distance edge monitoring set. Due to Lemma 25 and Lemma 27 we know, that $EM(S) \subseteq EM(N)$. Since $|S| > |N|$, no distance edge monitoring set containing all vertices from S can be minimal, thus any minimal distance edge monitoring set has to contain at most 1 vertex from S. □

Not only are bridges easy to monitor, but we can also show that we can search for the smallest distance edge monitoring set rather independently on either side of it.

Lemma 30. Let G be a connected graph and $e \in E(G)$ is a bridge. Let u and v be endpoints of e. Let G_u and G_v, be connected components remaining after removing e from G, and $u \in V(G_u)$, $v \in V(G_v)$. If $dem(G_u) \geq 2$ and $dem(G_v) \geq 2$, then we can get smallest edge monitoring set of G by taking the union of smallest edge monitoring set of G_u containing u and smallest edge monitoring set of G_v containing v and removing u and v from resulting set.

Note that according to Lemma 17 $dem(G_u) = 1$ means that G_u is a tree, same for G_v. We will deal with that case with Reduction Rule 31.

Proof. We first prove that edge from G_u monitored by a vertex from G_v is also monitored by u: Since e is a bridge, every path with one terminal vertex in G_u and other vertex in G_v contains e, thus it also contains its endpoints u and v, which when combined with lemma Lemma 22 means that all edges from G_u that are monitored by a vertex from G_v are also monitored by u and all edges from G_v that are monitored by a vertex from G_u are also monitored by v.

Next we prove that every edge from G_u monitored by u is also monitored by any vertex from G_v: Assume there is an edge $e_1 \in E(G_u)$ that is monitored by u, i.e, $\exists x \in V(G_u)$, e_1 is a part of every shortest path with u and x as terminal vertices, but is not monitored by a vertex $y \in G_v$. That would mean that there is a shortest path from y to x which does not contain e_1. We know there is such path since G is connected. Since x is in G_u and G_v, that path has to contain u and since e_1 is in G_u, that means that the part of the path that is different also has to be in G_u, which would mean that e_1 is not part of every path between u and x either. The proof of $EM(G_u) \cap E(G_v)$ being the same as $EM(v) \cap E(G_v)$ is analogous.

The edge e is monitored according Lemma 18. Due to degree of edge monitoring of both G_u and G_v being at least 2, there will be at least one more vertex from both G_u and G_v in the distance edge monitoring set and as such edges that were monitored by u and v are still monitored, therefore this approach yields a valid distance edge monitoring set. Since any edge monitored by other component is also monitored by u or v in its respective component, thus in order to find any smaller distance edge monitoring set would we would need to find smaller distance edge monitoring set in at least one of the components, but due to the fact we have selected minimal distance edge monitoring set for such component, thus the resulting distance edge monitoring set is minimal. $\square$

Chapter 4

The Algorithm Parametrized by Vertex Cover Number

One of the ways we will approach the problem is utilizing vertex cover, or rather properties of the graph, which are dependent on the vertex cover number. While the complexity of the resulting algorithm will depend on the vertex cover number, no part of it requires us to find the vertex cover. One of the reasons why the vertex cover number was chosen is that it is already known that there is a relation between these problems—*vertex cover* is a *distance edge monitoring set*, though it does not have to be minimal.

4.1 Reduction Rules

In this section, we will introduce reduction rules that will be utilized.

As was noted before, trees are easy to monitor. This also holds when the tree is attached to another graph. The following data reduction rule allows us to get rid of trees attached to the graph, i.e., we get a graph where every vertex has a degree 2 or higher. In the original article by Foucaud et al. [1] they use a notion of a base graph introduced by Leah Epstein, Asaf Levin and Gerhard J. Woeginger [5] for this purpose.

Reduction Rule 31. If G contains a vertex v of degree 1 connected to a vertex u of degree > 1, remove it. Then new $G - v$ has the same minimal distance edge monitoring set.

Lemma 32. Reduction Rule 31 is correct.

Proof. Since v is adjacent only to u and u is adjacent to at least 1 other vertex, $\{u\}$ is a vertex cut. According to Lemma 25, u monitors all edges monitored by v, since there is no edge in connected component containing v after removing u from the graph. Since (u, v) is a bridge, according to Lemma 18 it is monitored by any vertex in the same connected component

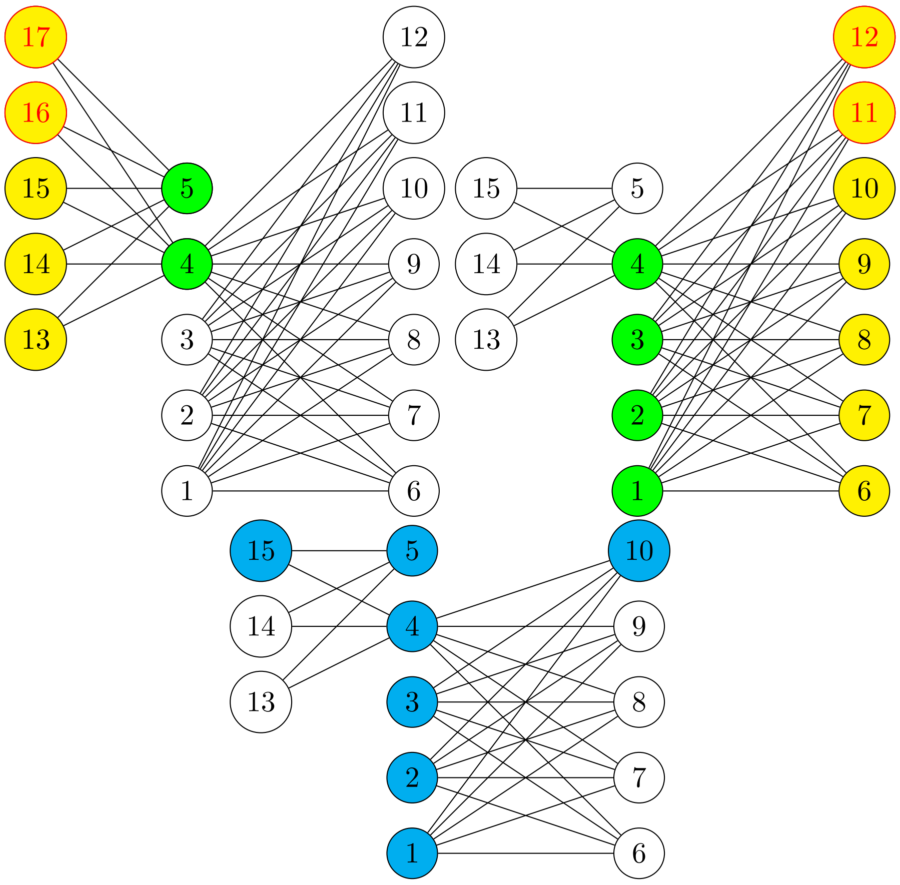

Figure 4.1: Graph before and after each application of Reduction Rule 33. Vertices highlighted in yellow form an independent set with the same neighbourhood (highlighted in green). Due to lemma Lemma 29 we can consider only vertices highlighted in blue when looking for minimal *distance edge monitoring set.*

as u. There has to be such vertex in a minimal distance edge monitoring set, since there is at least one more edge incident to u that needs to be monitored. Hence we can see that minimal distance edge monitoring set of $G - v$ is also minimal distance edge monitoring set of G. $\square$

Reduction Rule 33. If G contains an independent set I, where all vertices in I have the same neighbourhood N, where $|N| - |I| \geq 2$ then for $v \in I$, $G - v$ has the same minimal distance edge monitoring set.

Lemma 34. Reduction Rule 33 is correct.

Proof. Since N is the neighbourhood of vertices from I, it is a vertex cut. Therefore due to Lemma 25, $EM(I)$ is a subset of $EM(N)$. Due to Lemma 28 there will be either 0, 1, or all vertices from I in any minimal distance edge monitoring set. Since $|I| - |N| \geq 2$, There can not be all vertices from I in any minimal distance edge monitoring set and it will stay that way even after removing v from G. Due to Lemma 26 it does not matter which vertex from I we choose to remove from G, the size of the resulting minimal distance edge monitoring set will stay the same. $\square$

If we are only interested in the size of the distance edge monitoring set and not the individual vertices, we might remove a vertex from I even if $|N| - |I| = 1$.

Even though Reduction Rule 33 does not require us to find the vertex cover, the number of vertices left after applying this reduction rule as much as possible is related to the *vertex cover number* because having a small vertex cover limits the number of possible neighbourhoods.

Lemma 35. Reduction Rule 33 reduces the number of vertices to $\mathcal{O}\left(k + 2^k \cdot (k+1)\right)$, where k is the vertex cover number of G.

Proof. Every vertex outside of the vertex cover can only have edges to vertices that are part of the vertex cover, hence there are 2^k possible neighbourhoods for any given vertex outside of vertex cover. These vertices form an independent set. After applying Reduction Rule 33 as many times as possible, there will be at most one more vertex in each independent set than the size of its neighbourhood. Since no vertex can be connected to more vertices than k (all vertices in the vertex cover), there are at most $k+1$ vertices per unique neighbourhood. Since there are at most 2^k unique neighbourhoods, there are at most $k + 2^k \cdot (k+1)$ vertices. $\square$

We would need all vertices from I to monitor all edges between I and S. This can be done using all vertices from S. This has been shown in Lemma 25. Thus as long as there are more vertices in I than in S, at most one will be part of the smallest distance edge monitoring set.

However, it might be better to remember that all but one vertex from I can be ignored when making the set cover. Edges incident to marked vertices will have to be kept part of the universe.

4.2 Final Algorithm

We first apply the reduction rules, which might, depending on the input graph, result in a significantly smaller graph. In such graph we can proceed as normal by constructing edge monitoring sets for each vertex of the reduced graph and then finding the set cover of these sets. For more details see Algorithm 2.

Algorithm 2 An algorithm parameterized by the vertex cover number.

```
while $G$ contains vertex of degree 1 do
    Apply Reduction Rule 31 on $G$
end while
while Reduction Rule 33 can be applied do
    Apply Reduction Rule 33 on $G$
end while
for $v \in V(G)$ do
    for $e \in E(G)$ do
        if Distance from $v$ to at least one of the endpoints of $e$ changes when $e$ is removed from $G$ then
            Add $e$ to the set of edges monitored by $v$
        end if
    end for
end for
Find set cover, with universe being $E(G)$ and sets being edges monitored by individual vertices.
```

4.3 Implementation and Testing

The C++ language was chosen for the implementation because of its performance benefits. The generator of the test data and given algorithms were implemented. The algorithm parameterized by vertex cover number can be found as binary `vc_dem_solver.out`. It expects the input graph on the standard input in the following form:

The first line contains the time limit in milliseconds, after which the program is forced to stop. The second line contains a single integer n - the number of vertices. The following n lines contain the id of the vertex and a list of its neighbours.

The application returns the degree of edge monitoring for a given graph on the standard output. It returns -1 in cases where the time limit was exceeded.

The testing aimed to evaluate the performance of the implementation and efficiency of the reduction rules. The 2020 model year Macbook Air with an M1 CPU and 8GB RAM was used for the testing.

4.3.1 Test Data

Since the complexity of this algorithm depends on the size of the vertex cover, we have to use a graph G with a relatively small vertex cover. Otherwise, the modifications might not even help. We start with a vertices in the vertex cover and then continue adding vertices to the graph that are adjacent only to vertices in the vertex cover. (See Algorithm 3.)

The utility for generating graphs can be found as `vc_graph_generator.out`

Algorithm 3 Generating random graph with n vertices and a vertex cover number k and an edge probability P.

```
Generate a random graph $G$ of a size $k$, let vertices of $G$ be $u_1, u_2, \dots, u_k$
while $|V(G)| < n$ do
    Add a new vertex $v_i$ to $G$
    for $u_i \in \{u_1, u_2, \dots, u_k\}$ do
        With probability $P$ add an edge $(u_i, v_i)$ to $G$
    end for
end while
```

takes three parameters: the size of the vertex cover, number of vertices, and edge probability. The last number is the integer for time limit in milliseconds, so that the output of this utility can be redirected to the solver without modification.

these should be in order, on the standard input, separated by whitespace characters.

We have experimented with those three parameters. We have generated graphs with vertex cover of sizes ranging from 2 to 9, edge probability from 0.2 to 0.9 and a total number of vertices from 10 to 100.

4.3.2 Measured Results

The results were not surprising; the algorithm worked reasonably well on graphs with lower vertex cover numbers. We can see the dependency of application's run-time on the number of vertices in the graph with a fixed vertex cover number of four in Fig. 4.2. As expected, the run-time increases rapidly. However, it does not keep increasing like this forever; there is a limit on the number of vertices remaining after applying Reduction Rule 33, though these cases would have exceeded the time limit (in this case, 20 seconds).

Fig. 4.3 shows the dependency of the run-time on the edge probability. The vertex cover size was fixed at 4, and the number of vertices was fixed at 50. The time limit was set to 6 minutes; examples with such run-time have reached the limit and did not finish the calculation. As you can see, the run-time increases with increasing edge probability, but only up to some point.

Figure 4.4 shows dependency of the implementation's run-time on the vertex cover number with a number of vertices fixed at 100, and an edge probability fixed at 0.4. As expected, the run-time increases rapidly. Since the run-time exceeded 10 minutes with a vertex cover number of 5, we also show dependency of the number of vertices remaining after application of Reduction Rule 33 on the vertex cover number, though in that case we have started with 1000 vertices. (See Fig. 4.5.) There is an exponential correlation between the number of vertices and the run-time, although it is not perfect.

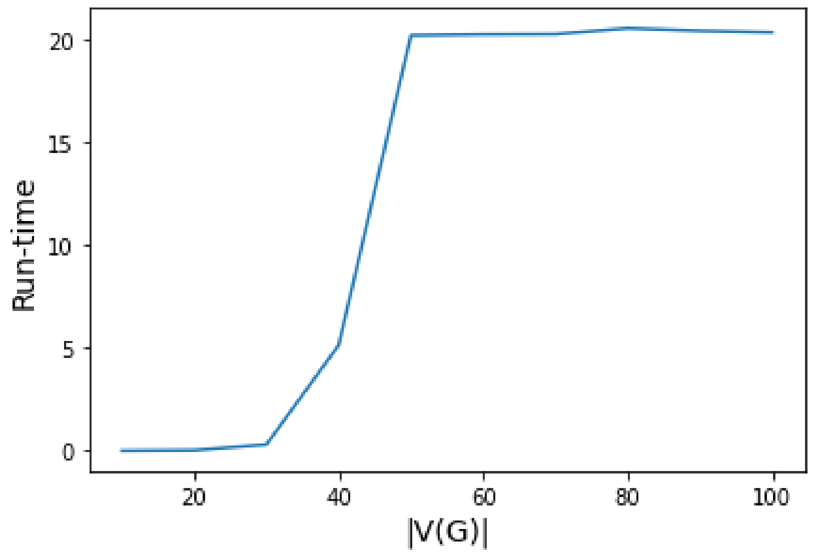

Figure 4.2: Relation between the number of vertices and run-time.

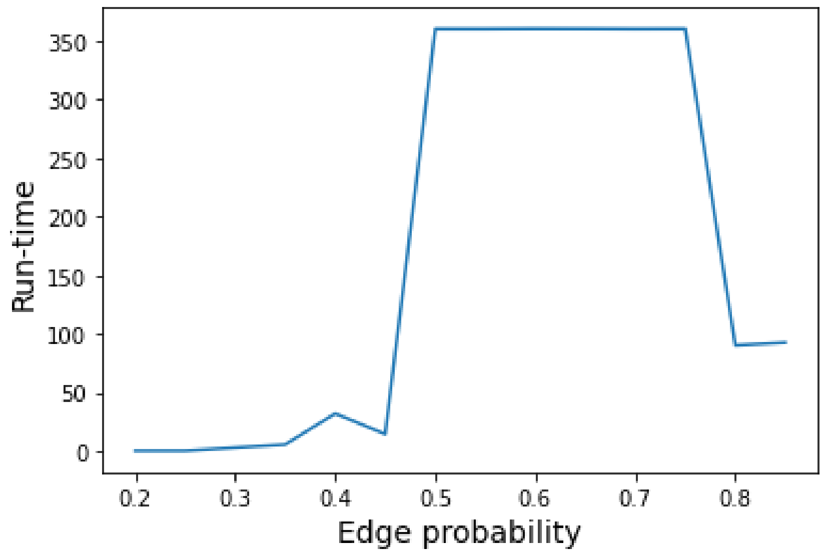

Figure 4.3: Relation between edge probability and run-time.

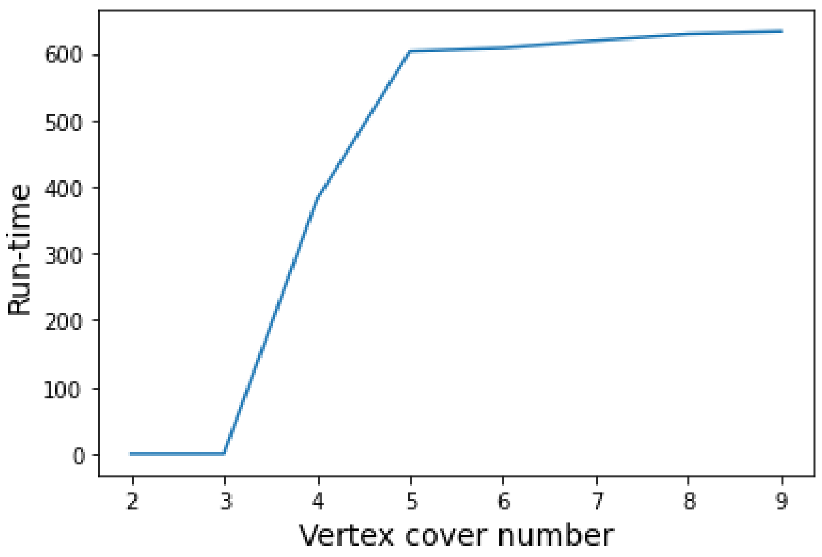

Figure 4.4: Relation between vertex cover number and runtime.

You can see more details about the performance in Table 4.1. Timeout for examples shown in there was set to 10 seconds.

The data shown on Fig. 4.3, Fig. 4.4, and Fig. 4.2 were visualized using Matplotlib [6] library in the Python programming language.

$\|V(G)\|$	edge probability	vertex cover #	$\|V(G')\|$	$dem(G)$	time
10	0.4	3	6	2	0.001
30	0.4	3	10	3	0.017
50	0.4	3	11	3	0.032
70	0.4	3	12	3	0.055
90	0.4	3	12	3	0.05
110	0.4	3	12	3	0.051
130	0.4	3	12	3	0.052
150	0.4	3	12	3	0.056
170	0.4	3	12	3	0.044
190	0.4	3	12	3	0.049
10	0.4	4	8	3	0.005
30	0.4	4	16	4	0.362
50	0.4	4	20	4	5.467
70	0.4	4	24	-	-
90	0.4	4	24	-	-
110	0.4	4	31	-	-
130	0.4	4	29	-	-
150	0.4	4	31	-	-
170	0.4	4	31	-	-
190	0.4	4	31	-	-
10	0.4	5	10	3	0.015
30	0.4	5	24	-	-
50	0.4	5	29	-	-
70	0.4	5	45	-	-
90	0.4	5	43	-	-
110	0.4	5	56	-	-
130	0.4	5	56	-	-
150	0.4	5	65	-	-
170	0.4	5	66	-	-
190	0.4	5	68	-	-
10	0.4	6	9	3	0.011
30	0.4	6	25	-	-
50	0.4	6	40	-	-
70	0.4	6	51	-	-
90	0.4	6	70	-	-
110	0.4	6	79	-	-
130	0.4	6	96	-	-
150	0.4	6	87	-	-
170	0.4	6	97	-	-
190	0.4	6	118	-	-
10	0.4	7	9	4	0.013
30	0.4	7	26	-	-

Table 4.1: Performance of the Algorithm 2.

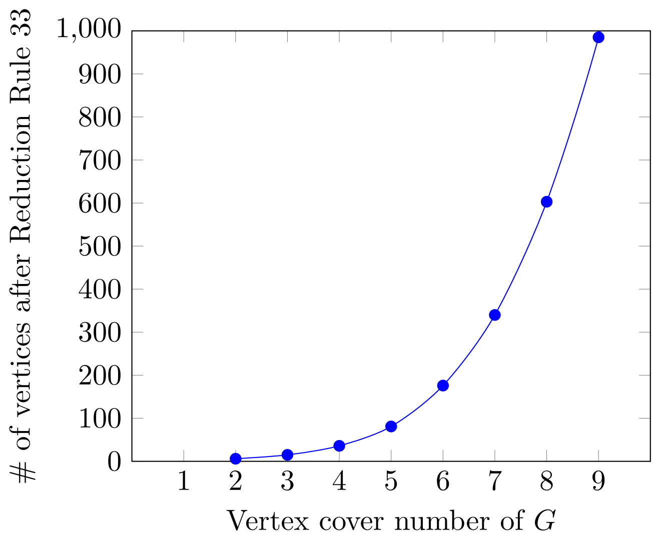

Figure 4.5: Efficiency of Reduction Rule 33 with relation to vertex cover number.

Chapter **5**

Parametrization By the Feedback Edge Set Number

Let G be a connected graph with no vertices of degree 1. We can get such a graph with the same *distance edge monitoring set* from any connected graph by iterative application of Reduction Rule 31 unless such graph is a tree. Let $v \in G$ be a vertex of degree 2 adjacent to at least 1 other vertex of degree 2 and its neighbours are not adjacent. Then we can remove v from the graph and join its neighbours by an edge. Let k be the feedback edge set number. Repeating this step until no longer possible gives a graph G' with at most $5k-5$ vertices [1]. This is because in such graph are at most $2k-2$ vertices of degree ≥ 3 and at most $3k-3$ vertices of degree $2k-2$.

The set $V(G')$ is an edge monitoring set of G, however it does not have to be minimal. Note that this is not a reduction rule; Fig. 5.1 shows examples where applying this step can both increase and decrease the degree of edge monitoring.

Lemma 36. Set $V(G')$ monitors all edges of G.

Proof. Every edge (u, v) from G' was either part of G, in which case it is monitored, since every vertex monitors incident edges, or it was originally an induced path. If the path had an odd number of vertices, or the endpoints are not adjacent, all of the edges on it are monitored by the endpoints, since every edge lies on the only shortest path between the closer endpoint and the middle vertex. If it had an even number of vertices, all of the edges except the middle one are monitored by the closer endpoint, since every path to the middle vertex that would not use only vertices from this path would have to go trough the other endpoint which would be longer by at least 1. We prove that middle edge (v_1, v_2) of such path is monitored by any vertex w on the path, assume that v_2 is further away from w, than v_1 when endpoints of the path are not used, i.e., on the path containing (v_1, v_2). Since both v_1 and v_2 were at the same distance from both endpoints when going trough the other

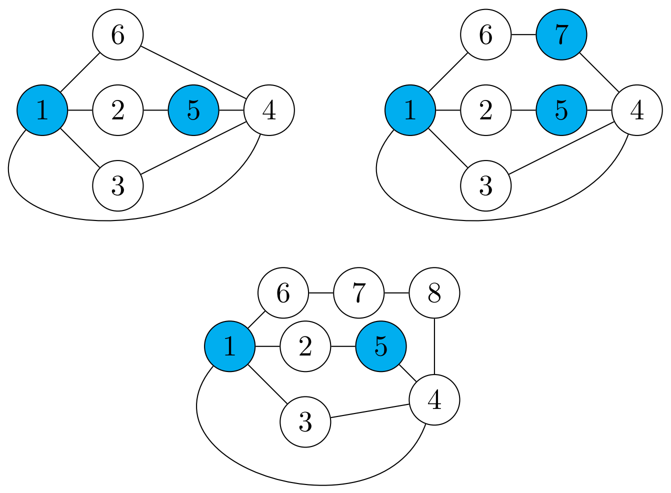

Figure 5.1: Dividing an edge can both increase and decrease size of a distance edge monitoring set. First graph is monitored by {1,5}, second by {1,5,7} and third one is monitored by {1,5}. In all cases these are optimal.

endpoint as when going trough (v_1, v_2) and distance between w and v_2 along the path has to be at least 1 shorter, while distance using endpoints is at least 1 longer, w monitors (v_1, v_2). □

We can further improve this by finding smallest subset of $V(G')$ that is still a distance edge monitoring set.

5.1 Performance Testing

Since this approach is proven only to provide a distance edge monitoring set, we need to check if it usually results in a minimal one. We might use the approach parameterized by vertex cover number to get the correct answer slightly faster, though that might not help too much for many graphs. For this we generate random graphs with low feedback edge set numbers and then compare the results to the approach that is proven to give correct results. Together with comparing results, we will also compare the performance of this algorithm compared to that of the original algorithm by Foucaud et al. [1].

As for the vertex cover number solver, the C++ language was used for the implementation. The binary executable of this implementation can be found as binary `fes_dem_solver.out`. The application expects the graph on the standard input in the same format as the `vc_dem_solver.out`.

For the data generation there is a binary named `fes_generator.out`. It uses Algorithm 5 and expects input in the following format:

Algorithm 4 Approach parameterized by FES number

```
while $G$ contains vertex of degree 1 do
    Apply Reduction Rule 31 on $G$
end while
$G' \leftarrow G$
while There are 2 adjacent vertices $u, v$ of degree 2 in $G'$, and $N(u) \cap N(v) = \emptyset$ do
    Add edge between $u$ and the other neighbour of $v$ to $G'$.
    Remove $v$ from $G'$
end while
                    ▷ Construct monitoring sets of $G$ for vertices present in $G'$
for $v \in V(G')$ do
    for $e \in E(G)$ do
        if Distance from $v$ to at least one of the endpoints of $e$ changes when $e$ is removed from $G$ then
            Add $e$ to the set of edges monitored by $v$
        end if
    end for
end for
Find set cover, with universe being $E(G)$ and sets being edges monitored by individual vertices.
```

An integer for the number of vertices in a graph before dividing edges. A decimal number for edge probability of such graph. An integer for the number of edge divisions. The last number is the integer for time limit in milliseconds, so that the output of this utility can be redirected to `fes_dem_solver.out`.

5.1.1 Test Graph Generation

For this approach to help, we need to have a graph with a relatively low feedback edge set number. We start with a random graph and then divide some of its edges to get such a graph. Such division would be done by removing the edge and adding a new vertex adjacent to its endpoints.

See Algorithm 5 for illustration.

Algorithm 5 Generating graph with low FES number

```
Generate a random graph $G$ with $k$ vertices
for $i \in \{1, 2, ..., n - k\}$ do
    Select random edge $e \in E(G)$ with endpoints $u, v$
    Add a new vertex $w$ to $G$
    Add edges $(u, w)$ and $(v, w)$ to $G$
    Remove $e$ from $G$
end for
```

$\lvert V(G)\rvert$	$dem(G)$	time default	$\lvert V(G')\rvert$	$dem(G)$ per Algorithm 4	time Algorithm 4
20	1	0.011	3	1	0.002
24	2	0.011	9	2	0.006
28	1	0.035	7	1	0.012
32	2	0.415	13	2	0.091
36	2	0.147	12	2	0.051
40	-	-	18	3	0.668
44	-	-	25	1	10.448
48	-	-	27	-	-
52	-	-	28	3	6.352
56	-	-	30	-	-
60	-	-	33	-	-
64	-	-	43	-	-

Table 5.1: Table showing efficiency of FES based approach and a case where it has failed.

5.1.2 Results

As you can see on Table 5.1, there can be a significant reduction in run time, if Algorithm 4 is used, assuming suitable data for its use. Unfortunately this also sometimes leads to a distance edge monitoring set, that is not minimal, as can also be seen on Table 5.1. Data for this table were generated with 0.2 as edge probability, for results with much more reasonable parameters see Table 5.2. For this table we have started with a graph with an edge probability of 0.4 and then we have added three times the number of the original vertices by dividing random edges. Cells without values mean, that the time limit was exceeded.

Since there already is an approximation that runs in polynomial time [1], we do not see much use for this algorithm in the future.

$\lvert V(G)\rvert$	$dem(G)$	time	$\lvert V(G')\rvert$	$dem(G)$ per Algorithm 4	time using Algorithm 4
20	1	0.012	3	1	0.002
24	2	0.011	9	2	0.006
28	1	0.035	7	1	0.012
32	2	0.424	13	2	0.094
36	2	0.155	12	2	0.053
40	-	-	18	3	0.68
44	-	-	25	1	10.458
48	-	-	27	-	-
52	-	-	28	3	6.414

Table 5.2: Table showing efficiency of FES based approach on graphs with more reasonable parameters.

Conclusion

The goal of the thesis was to summarize the Distance Edge Monitoring Set problem and the most important known results about it. The thesis aimed to develop a parameterized algorithm for the problem with respect to the size of the minimum vertex cover and the minimum size of a feedback edge and to implement these, test and evaluate their performance.

We got familiar with the distance edge monitoring set problem as introduced by Foucaud et al. [1] and what makes the problem difficult. We also looked at the concept of parameterized complexity with relation to this problem.

Various reduction rules for the problem were proposed, and their correctness was proven. An algorithm parameterized by the vertex cover number was proposed, and its correctness was proven. The algorithm was implemented in the C++ language, tested and evaluated. Unsurprisingly it performs best on graphs with a small vertex cover number and a comparatively vast number of vertices, where the reduction of the number of vertices that need to be considered was significant.

An algorithm parameterized by the feedback edge set number was also developed and implemented in the C++ language. The implementation of the algorithm was tested, and the dependency of the implementation's runtime on various parameters of the input graph was examined.

Possible Improvements

Other structural parameters could be considered; for example, the feedback vertex set and the approach parameterized by the feedback edge set could probably be improved. The implementation here mainly was to evaluate the real-world performance of these algorithms. Thus, could be packaged as a library with a friendly interface to make it usable by other people.

Bibliography

1. FOUCAUD, Florent; KAO, Shih-Shun; KLASING, Ralf; MILLER, Mirka; RYAN, Joe. Monitoring the edges of a graph using distances. *Discrete Applied Mathematics.* 2021. ISSN 0166-218X. Available from DOI: 10.1016/j.dam.2021.07.002.
2. GROSS, Jonathan L.; YELLEN, Jay (eds.). *Handbook of Graph Theory.* Chapman & Hall / Taylor & Francis, 2003. Discrete Mathematics and Its Applications. ISBN 978-1-58488-090-5. Available from DOI: 10.1201/9780203490204.
3. WEST, Douglas B. *Introduction to Graph Theory.* 2nd ed. Prentice Hall, 2000. ISBN 0130144002.
4. CYGAN, Marek; FOMIN, Fedor V.; KOWALIK, Lukasz; LOKSHTANOV, Daniel; MARX, Dániel; PILIPCZUK, Marcin; PILIPCZUK, Michal; SAURABH, Saket. *Parameterized Algorithms.* Springer, 2015. ISBN 978-3-319-21274-6. Available from DOI: 10.1007/978-3-319-21275-3.
5. EPSTEIN, Leah; LEVIN, Asaf; WOEGINGER, Gerhard J. The (Weighted) Metric Dimension of Graphs: Hard and Easy Cases. *Algorithmica.* 2015, vol. 72, no. 4, pp. 1130–1171. Available from DOI: 10.1007/s00453-014-9896-2.
6. HUNTER, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering.* 2007, vol. 9, no. 3, pp. 90–95. Available from DOI: 10.1109/MCSE.2007.55.

Appendix **A**

Contents of enclosed SD card

```
Readme.md ............ the file with SD card contents description
├── exe .................... the directory with executables
├── src ...................... the directory of source codes
│   ├── implementation ............ implementation sources
│   └── thesis ....... the directory of LaTeX source codes of the thesis
└── text ........................ the thesis text directory
    └── thesis.pdf .............. the thesis text in PDF format
```