FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF BACHELOR’S THESIS

Title:	Minimum Eccentricity Shortest Path Problem with Respect to Structural Parameters
Student:	Martin Kučera
Supervisor:	RNDr. Ondřej Suchý, Ph.D.
Study Programme:	Informatics
Study Branch:	Computer Science
Department:	Department of Theoretical Computer Science
Validity:	Until the end of summer semester 2020/21

Instructions

Get familiar with the Minimum Eccentricity Shortest Path problem and with the basic notions of Parameterized Complexity.
Get familiar with the XP algorithm for the problem parameterized by the desired eccentricity and eventually with the polynomial time approximation algorithms.

Develop a parameterized algorithm for the problem with respect to the size of the minimum vertex cover 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 above mentioned algorithms and implement it in a suitable language.

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

References

Feodor F. Dragan, Arne Leitert: On the minimum eccentricity shortest path problem. Theor. Comput. Sci. 694: 66-78 (2017)

doc. Ing. Jan Janoušek, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague February 4, 2020

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Minimum Eccentricity Shortest Path Problem with Respect to Structural Parameters

Martin Kučera

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

June 1, 2020

Acknowledgements

First and foremost, I wish to express my deepest gratitude to my supervisor, Ondřej Suchý. Without his dedicated involvement and valuable insights in every step of the process, this thesis would have never been accomplished. I am also grateful to the whole Department of Theoretical Computer Science at FIT CTU, for being great and inspiring teachers who have always known how to make me excited about their subjects. Special thanks go to Ó Velký Tung and all other members of the FIKS seminar, for their suggestions and encouragement. Finally, I wish to thank my parents for their love and support throughout my studies.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46 (6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity.

In Prague on June 1, 2020 …………………

Citation of this thesis

Kučera, Martin. *Minimum Eccentricity Shortest Path Problem with Respect to Structural Parameters*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

Abstract

The Minimum Eccentricity Shortest Path Problem consists in finding a shortest path with minimum eccentricity in a given graph. The problem is known to be NP-complete and W[2]-hard with respect to the minimum eccentricity. In this thesis, we present FPT algorithms for the problem parameterized by the maximum leaf number, neighborhood diversity, twin cover, distance to cluster, and the combination of distance to disjoint paths with the minimum eccentricity. In addition, we present an experimental evaluation of the last algorithm, which we have implemented.

Keywords graph theory, minimum eccentricity shortest path, parameterized complexity, fixed-parameter tractable

Abstrakt

Problém Nejkratší Cesty s Minimální Excentricitou spočívá v nalezení takové nejkratší cesty v daném grafu, jejíž excentricita je minimální. O problému je známo, že je NP-úplný a W[2]-těžký vzhledem k minimální excentricitě. V práci představujeme FPT algoritmy pro tento problém parametrizované velikostí „maximum leaf number“, „neighborhood diversity“, „twin cover“, „distance to cluster“ a kombinací „distance to disjoint paths“ s minimální excentricitou. Dále představujeme experimentální vyhodnocení posledního z algoritmů, který jsme také naimplementovali.

Klíčová slova teorie grafů, nejkratší cesta s minimální excentricitou, parametrizovaná složitost, parametrizovaný algoritmus

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Minimum Eccentricity Shortest Path (MESP) problem asks, given a graph, to find a shortest path with minimum eccentricity—a shortest path whose distance to all other vertices in the graph is minimal. This shortest path may be viewed as the "most accessible", and as such, may find applications in communication networks, transportation planning, water resource management, and fluid transportation [1]. Furthermore, the MESP can be used to obtain the best to date approximation for a minimum distortion embedding of a graph into the line [1].

The problem was first introduced by Dragan and Leitert [1]. They showed that it is NP-hard on general graphs and constructed a slice-wise polynomial (XP) algorithm, which finds a shortest path with eccentricity at most k in a graph with n vertices and m edges in $\mathcal{O}(n^{2k+2}m)$ time. They also presented a linear-time algorithm that solves the MESP problem for trees. Additionally, they developed a 2-approximation, a 3-approximation, and an 8-approximation algorithm that runs in $\mathcal{O}(n^3)$ time, $\mathcal{O}(nm)$ time, and $\mathcal{O}(m)$ time, respectively. Birmelé, Montgolfier, and Planche [2] further improved the 8-approximation to a 3-approximation, which still runs in linear time. Dragan

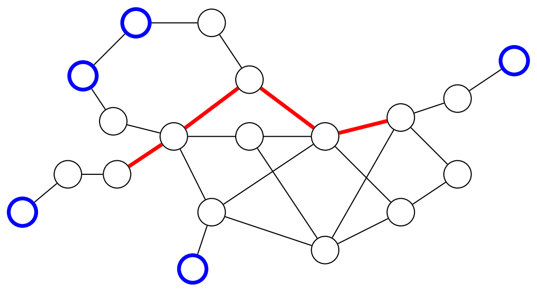

Figure 1.1: Example of a path with eccentricity 2. The path is highlighted in red, the vertices at distance 2 from the path are highlighted in blue.

and Leitert [3] showed that the MESP problem can be solved in linear time for distance-hereditary graphs (generalizing the previous result for trees) and in polynomial time for chordal graphs and dually chordal graphs. Later, they proved [4] that the problem is NP-hard even for bipartite planar graphs with maximum degree 3, and W[2]-hard with respect to the minimum eccentricity for general graphs. Furthermore, they showed that in a graph with a shortest path of eccentricity k, a minimum k-dominating set can be found in $n^{\mathcal{O}(k)}$ time. Birmelé et al. [5] presented a generalization of the MESP problem by decomposing a graph into subgraphs with bounded shortest-path eccentricity, the hub-laminar decomposition, which is motivated by DNA assembly in biology.

Our contribution The goal of this thesis is to develop parameterized algorithms for the MESP problem with respect to various structural parameters of the graph. We introduce parameterized algorithms with respect to the maximum leaf number, neighborhood diversity, twin cover, distance to cluster, and distance to disjoint paths combined with the minimum eccentricity. In addition, we present an implementation of one of the algorithms and provide an experimental evaluation.

Outline In Chapter 2, we provide necessary notations and formal definitions that will be used throughout the thesis. Next, in Chapter 3, we describe parameterized algorithms with respect to the maximum leaf number, neighborhood diversity, twin cover, distance to cluster, and distance to disjoint paths combined with the minimum eccentricity. In Chapter 4, we describe the implementation of the distance to disjoint paths parameterized algorithm and show various benchmarks. Finally, in Chapter 5, we summarize our contributions and discuss possible future work.

CHAPTER 2

Preliminaries

We consider finite, connected, unweighted, undirected, loopless graphs without multiple edges. Graph $G = (V, E)$ is a pair of a set of vertices V and a set of edges E. An edge is a set of vertices of size 2. We denote $n = |V|$ and $m = |E|$.

We denote an ordered sequence of elements $\boldsymbol{s} = (s_1, \ldots, s_{|\boldsymbol{s}|})$. For two sequences $\boldsymbol{s} = (s_1, \ldots, s_{|\boldsymbol{s}|}), \boldsymbol{t} = (t_1, \ldots, t_{|\boldsymbol{t}|})$ we denote their concatenation

$$\boldsymbol{s} \frown \boldsymbol{t} = (s_1, \ldots, s_{|\boldsymbol{s}|}, t_1, \ldots, t_{|\boldsymbol{t}|}).$$

A path is a sequence of vertices such that every two consecutive vertices in the sequence are connected by an edge. The length of a path P from vertex $u \in V$ to $v \in V$ is the number of edges in it, i.e., $|P| - 1$. The distance $d_G(u, v)$ between two vertices $u, v \in V$ is the length of the shortest path between u and v.

The distance between a vertex $u \in V$ and a set of vertices $S \subseteq V$ is $d_G(u, S) = \min_{s \in S} d_G(u, s)$. The eccentricity of a vertex $v \in V$ is defined as $\mathrm{ecc}_G(v) = \max_{u \in V} d_G(u, v)$. The eccentricity of a set $S \subseteq V$ is $\mathrm{ecc}_G(S) = \max_{u \in V} d_G(u, S)$.

For a path P, we denote $V(P) \subseteq V$ the set of vertices that are in P. However, we simply use P instead of $V(P)$ if it is clear from the context that we are dealing with a set of vertices. For example, $d_G(u, P) = d_G(u, V(P))$ and $\mathrm{ecc}_G(P) = \mathrm{ecc}_G(V(P))$.

For vertex $u \in V$ we denote $\deg_G(u)$ the degree of u, $N_G(u)$ the open neighborhood of u and $N_G[u]$ the closed neighborhood of u. We denote $N_G^i[u] = \{v \in V \mid d_G(u, v) \leq i\}$ the set of vertices at distance at most i from u. We denote $G[S]$ the induced subgraph of G on vertices $S \subseteq V$. We denote $G \setminus S = G[V \setminus S]$.

In this thesis, we focus on the following problem.

Definition 1 (Minimum Eccentricity Shortest Path Problem)**.** For a given graph G, and an eccentricity $k \in \mathbb{N}$, decide whether there exists a shortest path P in G such that $\mathrm{ecc}_G(P) \leq k$.

Observation 1. Given a graph G and a path P in G, the value of $\mathrm{ecc}_G(P)$ may be calculated in $\mathcal{O}(n + m)$ time.

Proof. Perform a breadth-first search (BFS) starting in all vertices from P with their distance set to 0. The distance of the last-discovered vertex is the eccentricity of the path. □

The following is known about the complexity of the problem.

Theorem 1 (Dragan, Leitert [1])**.** The MESP problem is NP-complete.

Theorem 2 (Dragan, Leitert [4])**.** The MESP problem is W[2]-hard with respect to the eccentricity k.

In this thesis, we focus on parameterized algorithms for the MESP problem, i.e., algorithms that run in polynomial time for any graph with a fixed value of a certain parameter.

Definition 2. An algorithm with input x is called *fixed-parameter tractable* (FPT) with respect to a parameter p if it runs in $f(p) \cdot |x|^{\mathcal{O}(1)}$ time for some computable function f.

Definition 3. An algorithm with input x is called *slice-wise polynomial* (XP) with respect to a parameter p if it runs in $|x|^{f(p)}$ time for some computable function f.

In this thesis, we present FPT algorithms for the MESP problem with respect to several different structural parameters. First, we define a property that will be useful for defining neighborhood diversity and twin cover.

Definition 4. Let u, v be two vertices of a graph G. We say u and v are *twins* if $N_G(u) \setminus \{v\} = N_G(v) \setminus \{u\}$.

Now we define all the parameters for which we show FPT algorithms in Chapter 3.

Definition 5 (Maximum leaf number)**.** The *maximum leaf number* of a graph G is the maximum number of leaves in a spanning tree of G.

Definition 6 (Neighborhood diversity)**.** For a graph G, the *neighborhood diversity* is the minimum number $d \in \mathbb{N}$ such that the vertices of G can be partitioned into d disjoint sets $C_1, \ldots, C_d$, such that for each $i \in \{1, \ldots, d\}$ every two vertices $u, v \in C_i$ are twins.

Definition 7 (Vertex cover)**.** For a graph $G = (V, E)$, a *vertex cover* is a vertex subset $X \subseteq V$ such that for every edge $\{u, v\} \in E : u \in X \vee v \in X$. The *vertex cover number* is the size of the smallest vertex cover.

Definition 8 (Twin cover)**.** For a graph $G = (V, E)$, a *twin cover* is a vertex subset $X \subseteq V$ such that for every edge $\{u, v\} \in E$ either

(a) $u \in X \vee v \in X$, or

(b) u and v are twins.

The *twin cover number* is the size of the smallest twin cover.

Definition 9 (Distance to cluster)**.** For a graph $G = (V, E)$, a *modulator to cluster* is a vertex subset $X \subseteq V$ such that $G \setminus X$ is a vertex-disjoint union of cliques. The *distance to cluster* is the size of the smallest modulator to cluster.

Definition 10 (Distance to disjoint paths)**.** For a graph $G = (V, E)$ a *modulator to disjoint paths* is a vertex subset $X \subseteq V$, such that $G \setminus X$ is a vertex-disjoint union of paths. The *distance to disjoint paths* is the size of the smallest modulator to disjoint paths.

Chapter 3

Parameterized Algorithms

In this chapter, we present several FPT algorithms for the MESP problem. In Section 3.1, we present an algorithm parameterized by the maximum leaf number. In Section 3.2, we present an algorithm parameterized by the neighborhood diversity. In Section 3.3 we define the Constrained Set Cover (CSC) problem and present an algorithm to solve it. Then we show an FPT algorithm for the MESP problem on bipartite graphs parameterized by the size of one partition by reducing the MESP problem to the CSC problem. In Section 3.4, we generalize the algorithm to allow edges in the parameterized partition and thus show an FPT algorithm parameterized by the vertex cover. In Section 3.5, we show that, in fact, the same algorithm also works correctly with the twin cover parameter. In Section 3.6, we further generalize the algorithm and make it parameterized by the distance to cluster. In Section 3.7, we present an FPT algorithm parameterized by the distance to disjoint paths and the minimum eccentricity, combined. Again, this algorithm depends on the solution of the CSC problem.

3.1 Maximum Leaf Number

In this section, we present an FPT algorithm for the MESP problem parameterized by the maximum leaf number. We use the following lemma to find a smaller graph in which the problem can be solved by brute-force.

Lemma 1 (Bouland [6])**.** If the maximum leaf number of a graph G is equal to ℓ, then G is a subdivision of a graph $\widetilde{G} = (C, \widetilde{E})$, such that $C \subseteq V$ and $|C| = \mathcal{O}(\ell)$.

Observation 2. Let $G = (V, E)$ be a graph with maximum leaf number ℓ. Then, $\widetilde{G}$ is easy to obtain by contracting all degree 2 vertices as long as they are not part of a triangle.

Our algorithm will, in fact, rely not on the maximum leaf number, but on the set C of vertices from $\widetilde{G}$ instead. Thanks to Lemma 1, we know that the size of C is linear with respect to the maximum leaf number. The definition of FPT does not put any limitations on the time complexity with respect to the size of the parameter. In particular, this means we may "guess" the subset of vertices from C, which appear on the desired path by trying all possible combinations. Also, we may "guess" the order in which these vertices should appear on the path by trying all possible permutations.

Definition 11. Suppose that a shortest path P with $\mathrm{ecc}_G(P) \leq k$ exists. Let $s, t \in V$ be the endpoints of P. We denote $\widehat{C} = C \cup \{s, t\}$, $L = P \cap \widehat{C}$ and $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_{|L|})$ the permutation/order of vertices from L in which they appear on P.

Now, we observe that given the correct subset and order of vertices from C, the shortest path with eccentricity at most k is easy to find, if it exists.

Lemma 2. Between each π_i, π_{i+1} there is exactly one path such that it does not contain any vertices from $\widehat{C}$ other than π_i and π_{i+1}.

Proof. The part of P between π_i and π_{i+1} does not contain any other vertices from $\widehat{C}$, otherwise there would have to be some $\pi_j \in \boldsymbol{\pi}$ between π_i and π_{i+1}. As shown in Figure 3.1, no other such path between π_i and π_{i+1} exists because $\widetilde{G}$ is not a multigraph. □

Corollary 1. If $\boldsymbol{\pi}$ is known in advance, the path P can be found in linear time.

Proof. Find P by executing a BFS $|L|-1$ times, starting in each of $\pi_1, \ldots, \pi_{|L|-1}$ and from each π_i finding the shortest path to π_{i+1}, ignoring all vertices from $\widehat{C}$ other than π_{i+1}. While searching for the path between π_i and π_{i+1}, no vertex from the path between π_{i+1} and π_{i+2} will be discovered because P is a shortest path. Thus, in each BFS, all the vertices visited in the previous BFSs may be ignored, resulting in the overall complexity $\mathcal{O}(n + m)$. □

Now we present Algorithm 1. It simply iterates through all possible combinations of s and t, subsets L of $\widehat{C}$ and their permutations $\boldsymbol{\pi}$. For each of them, it tries to construct the corresponding path.

Lemma 3. Algorithm 1 finds a path with eccentricity at most k, if one exists. Otherwise, it reports that no such path exists.

Proof. If a shortest path P with $\mathrm{ecc}_G(P) \leq k$ exists, there is an iteration of the foreach-loop on line 2 in which $\bar{\boldsymbol{\pi}} = \boldsymbol{\pi}$. Then, by Corollary 1, the algorithm will find the path $\bar{P} = P$ on line 3.

Because of the check on line 4, we can be sure that if the algorithm returns a path, it is a shortest path with eccentricity at most k. □

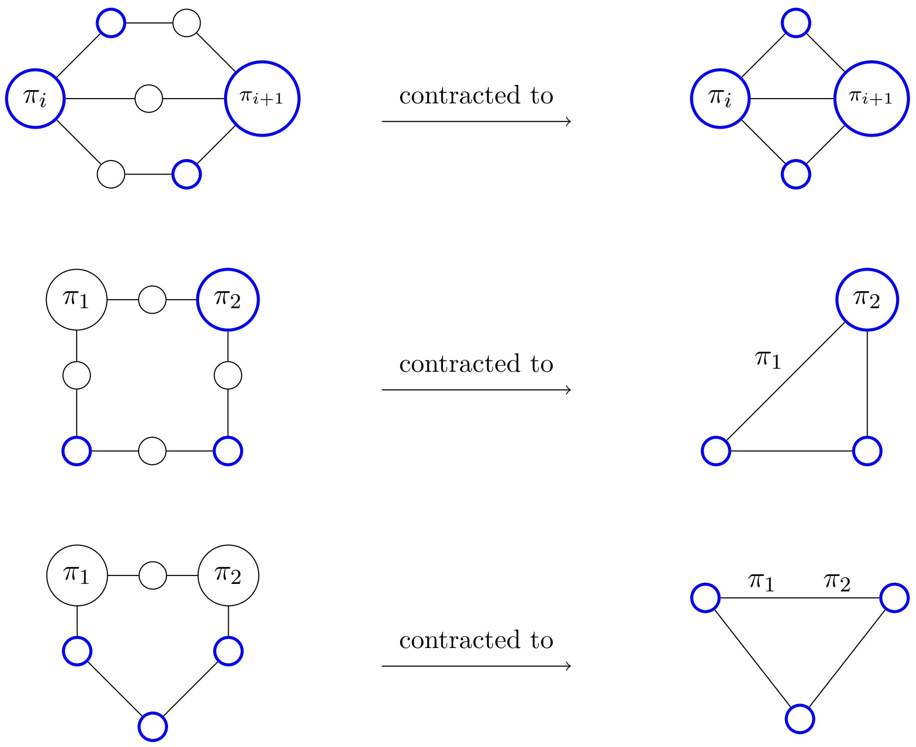

Figure 3.1: Examples of situations from Lemma 2. On the left, we see a part of the graph G with multiple paths between vertices π_i and π_{i+1}. On the right, we see the corresponding part in $\widetilde{G}$. Vertices from C are highlighted in blue.

Algorithm 1: FPT Maximum Leaf Number

```
  Input: Graph $G = (V, E)$, set $C \subseteq V$.
1 foreach $\{\bar{s}, \bar{t}\} \subseteq V, \bar{L} \subseteq C \cup \{\bar{s}, \bar{t}\}$ do
2     foreach $\bar{\pi}$ permutation of $L$ with $\bar{\pi}_1 = \bar{s}, \bar{\pi}_{|\bar{L}|} = \bar{t}$ do
3         Try to construct path $\bar{P}$ from $\bar{\pi}_1$ through $\bar{\pi}_2, \ldots, \bar{\pi}_{|\bar{L}|-1}$ to $\bar{\pi}_{|\bar{L}|}$
            that does not contain any other vertices from $C$
4         if $\bar{P}$ exists, $d_G(\bar{\pi}_1, \bar{\pi}_{|\bar{L}|}) = |\bar{P}| - 1$, and $\mathrm{ecc}_G(\bar{P}) \leq k$ then
5             return $\bar{P}$
6 No shortest path with eccentricity at most $k$ exists
```

Lemma 4. Algorithm 1 can be implemented to run in $\mathcal{O}(2^c c! \cdot n^4)$ time where $c = |C|$.

Proof. There are $\mathcal{O}(n^2)$ combinations of $\{\bar{s}, \bar{t}\} \subseteq V$. For each of them, there are $\mathcal{O}(2^c)$ combinations of $\bar{L} \subseteq C$. For each $\bar{L}$ there are $\mathcal{O}(c!)$ permutations of $\bar{\boldsymbol{\pi}}$. Given $\bar{\boldsymbol{\pi}}$, path $\bar{P}$ can be found in $\mathcal{O}(n+m)$ time, as shown in Corollary 1. On line 4, the distance can be checked in $\mathcal{O}(n+m)$ time, and by Observation 1, so can the eccentricity of $\bar{P}$. □

3.2 Neighborhood Diversity

In this section, we present an FPT algorithm for the MESP problem parameterized by the neighborhood diversity.

Let $G = (V, E)$ be a graph with neighborhood diversity d and $C_1, \dots, C_d$ be the corresponding sets of vertices, as described in Definition 6. We start by observing that there is a metagraph for G on d vertices, which keeps many properties of G.

Observation 3. Every set of vertices $C_i \in \{C_1, \dots, C_d\}$ is either an independent set or a clique.

Proof. Clearly, this holds if $|C_i| \leq 2$. Suppose that $|C_i| \geq 3$ and let $\{x, y, z\} \subseteq C_i$. We have $\{x, y\} \in E$ if and only if $\{x, z\} \in E$ because y and z are twins, and $\{x, y\} \in E$ if and only if $\{z, y\} \in E$ because x and z are twins. □

Observation 4. For each pair of sets $C_i, C_j \in \{C_1, \dots, C_d\}$, there is either a full bipartite subgraph between the vertices of C_i and C_j, or there are no edges between the two sets.

Proof. Let $u_1, u_2 \in C_i$ and $v \in C_j$. We have $\{u_1, v\} \in E$ if and only if $\{u_2, v\} \in E$ because u_1 and u_2 are twins. □

Definition 12. The *metagraph* of G is a graph $G_M = (V_M, E_M)$ with $V_M = \{1, \dots, d\}$. An edge $\{i, j\}$ is present in E_M if and only if there is a complete bipartite subgraph between C_i and C_j in G.

Lemma 5. If G is connected, then G_M is also connected.

Proof. Let $i, j \in V_M, u \in C_i, v \in C_j$ and P some path from u to v in G. We can construct a sequence P_M of vertices from V_M starting in i and ending in j: Iterate over the vertices in P starting in u, and for each vertex $p \in P$, add $q \in V_M$ into P_M such that $p \in C_q$. For every two consecutive vertices $a, b \in P_M$, either $a = b$ or $\{a, b\} \in E_M$ because by Observation 4 there is a complete bipartite subgraph between C_a and C_b in G. P_M can be reduced into a walk by leaving out repeated vertices. Thus, i and j are in the same component in G_M. □

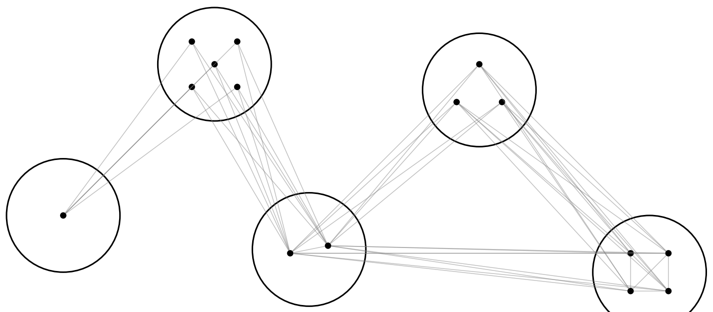

Figure 3.2: Example of a graph with neighborhood diversity 5.

Observation 5. The metagraph G_M can be obtained from G by iterating over all pairs of vertices, checking whether they are twins, and updating V_M accordingly. The presence of an edge $\{i, j\} \in E_M$ is then equivalent to the presence of an edge $\{u, v\} \in E$ for any $u \in C_i, v \in C_j$.

Now, we make some observations about the minimum eccentricity shortest path in a graph G whose neighborhood diversity is small.

Observation 6. Let G be a graph with neighborhood diversity 1. Because we only consider connected graphs, G must be a clique. Then, the minimum eccentricity shortest path in G is any single edge, and its eccentricity is 1.

Lemma 6. Let G be a graph with neighborhood diversity 2, i.e., $V_M = \{1, 2\}$. Because we only consider connected graphs, we have $\{1, 2\} \in E_M$. At least one of C_1, C_2 must be an independent set on at least two vertices in G.

Proof. Let C_1 and C_2 both be cliques (note that a single vertex is also a clique). Because G is connected, by Observation 4, there is a full bipartite subgraph between C_1 and C_2. Then, $C_1 \cup C_2$ is also a clique and every two vertices in G are twins, which is a contradiction with the graph having neighborhood diversity 2. □

Corollary 2. If G has neighborhood diversity 2, then the minimum eccentricity shortest path is $P = (x, y, z)$ and $\mathrm{ecc}_G(P) \leq 1$ for any $x, z \in C_i$ and $y \in C_j$ such that C_i is an independent set in G.

Proof. P is clearly a shortest path. If $G = P$, then P is clearly the minimum eccentricity shortest path with $\mathrm{ecc}_G(P) = 0$. Otherwise, we have $\mathrm{ecc}_G(P) = 1$ because every vertex in set C_i is a neighbor of all vertices from the other set C_j out of which at least one is in P, and vice versa. Clearly, a shortest path may contain at most two vertices from C_i and at most one vertex from C_j; therefore, P is the minimum eccentricity shortest path. □

Now, we should be able to find a minimum eccentricity shortest path in graphs with neighborhood diversity at most 2 pretty easily. We continue by showing that the structure of the metagraph restricts the structure of any shortest path in the original graph significantly.

Lemma 7. Let P be a shortest path in G of length at least 3. For each $i \in V_M$, there is at most one vertex from C_i in P.

Proof. Let $P = (p_1, \dots, p_\ell)$. Suppose that $p_s, p_t \in C_i$. Clearly, at least one of p_s, p_t is not an endpoint of P (otherwise the length of P would be at most 2). Without loss of generality, suppose that p_s is not an endpoint of P, thus it has a predecessor p_{s-1} on P. We have $P = (p_1, \dots, p_{s-1}, p_s, \dots, p_t, \dots, p_\ell)$. Because p_s and p_t are twins and $\{p_{s-1}, p_s\} \in E$, we have $\{p_{s-1}, p_t\} \in E$ and P may be shortened to $P' = (p_1, \dots, p_{s-1}, p_t, \dots, p_\ell)$. Therefore, P is not a shortest path. □

Corollary 3. If G has neighborhood diversity at least 3, then there is a minimum eccentricity shortest path in G which contains at most one vertex from C_i for each $i \in V_M$.

Proof. Let P be the minimum eccentricity shortest path in G. If the length of P is at least 3, then this holds by Lemma 7. If the length of P is at most 2 and P contains two vertices from some C_i, they must both be the endpoints of P. There is a vertex $j \in V_M$ such that $\{i, j\} \in E_M$ and P does not contain any vertex from C_j. Consider path P' created from P by appending a vertex $u \in C_j$ after an endpoint of P and removing the other endpoint of P. Clearly, P' is a shortest path and all neighbors of P are also neighbors of P'. Thus, $\mathrm{ecc}_G(P') \leq \mathrm{ecc}_G(P)$. □

We have shown that there is a minimum eccentricity shortest path which contains at most one vertex from each set C_i. Finally, we show that with respect to eccentricity, all vertices in the same set are equivalent, and it does not matter which of them we choose. In other words, the minimum eccentricity shortest path in G can be found by trying all shortest paths in G_M.

Lemma 8. Let P be a shortest path in G. Create path P' by substituting any vertex $p \in P$ by its twin p'. Then, $\mathrm{ecc}_G(P') = \mathrm{ecc}_G(P)$.

Proof. Let $u \in V$ such that $u \neq p$ and $u \neq p'$. Then, $d_G(u, p) = d_G(u, p')$ because p and p' are twins. Thus, $d_G(u, P) = d_G(u, P')$. Moreover, $d_G(p, P') = d_G(p', P) = 1$. □

Based on what we have shown, we propose Algorithm 2 to solve the MESP problem. It handles separately the graphs with neighborhood diversity 1 and 2. For neighborhood diversity at least 3, it iterates through all the subsets of V_M and all their permutations (containing all shortest paths in G_M), and for each of them checks whether they form a shortest path with eccentricity at most k in G.

Algorithm 2: FPT Neighborhood Diversity

```
Input: Graph $G = (V, E)$, sets $C_1, \dots, C_d \subseteq V$
if $d = 1$ then
    return any edge in $E$
else if $d = 2$ then
    let $C_i$ be an independent set in $G$
    let $j \in \{1, 2\}, j \neq i$
    let $x, z \in C_i, y \in C_j$
    return $(x, y, z)$
else
    foreach $L \subseteq \{1, \dots, d\}, \boldsymbol{\pi}$ permutation of $L$ do
        foreach $i \in \{1, \dots, |L|\}$ do
            let $p_i \in C_{\pi_i}$
        $P := (p_1, \dots, p_{|L|})$
        if $P$ is a shortest path and $\mathrm{ecc}_G(P) \leq k$ then
            return $P$
    No shortest path with eccentricity at most $k$ exists.
```

Lemma 9. Algorithm 2 finds a shortest path with eccentricity at most k if it exists and reports if it does not exist.

Proof. By Observation 6, the algorithm works correctly if the neighborhood diversity is 1 (lines 1–2).

By Corollary 2, the algorithm works correctly if the neighborhood diversity is 2 (lines 3–7).

If the neighborhood diversity is at least 3 and a shortest path with eccentricity at most k exists, then by Corollary 3 it contains at most one vertex from each C_i. Thus, there is an iteration on lines 10–14, which finds the corresponding sequence $\boldsymbol{\pi}$ of vertices in V_M. Then, by Lemma 8, it does not matter which vertex from each C_i is chosen, so the algorithm will find a path with the desired eccentricity.

Because of the check on line 13, the path that the algorithm finds is always a shortest path with eccentricity at most k. □

Lemma 10. Algorithm 2 runs in $\mathcal{O}(2^d d! \cdot n^2)$ time.

Proof. Lines 1–7 can be implemented in constant time. There are 2^d possible sets L, and for each of them there are at most $\mathcal{O}(d!)$ possible permutations $\boldsymbol{\pi}$. Lines 10–12 can be implemented in $\mathcal{O}(d)$ time. The checks on line 13 can be implemented in $\mathcal{O}(n + m)$ time. □

3.3 Bipartite Graphs

In this section, we present an FPT algorithm for the MESP problem on bipartite graphs parameterized by the size of one partition. All the following sections will be based on this algorithm. We start by defining the Constrained Set Cover (CSC) problem and proposing an algorithm to solve it. Then, we reduce the MESP problem on bipartite graphs to the CSC problem.

Definition 13 (Constrained Set Cover)**.** Let $\mathcal{R} = \{r_1, \dots, r_n\}$ be a set of requirements that need to be satisfied. Let $\mathcal{C} = C_1 \cup \dots \cup C_m$ be a union of disjoint sets of candidates. Let $\Psi : \mathcal{C} \to 2^{\mathcal{R}}$ be a function that determines for each candidate which requirements it satisfies.

The *constrained set cover* is a set of candidates, exactly one from each set: $s_1 \in C_1, \dots, s_m \in C_m$ such that together they satisfy all the requirements.

$$\Psi(s_1) \cup \dots \cup \Psi(s_m) = \mathcal{R}$$

To help us solve the CSC problem, we now define a function D_i for each $i \in \{1, \dots, m\}$.

Definition 14. Let $\mathcal{R} = \{r_1, \dots, r_n\}, \mathcal{C} = C_1 \cup \dots \cup C_m, \Psi : \mathcal{C} \to 2^{\mathcal{R}}$ be an instance of the CSC problem. We define function $D_i : 2^{\mathcal{R}} \to \mathcal{C} \cup \{\top, \bot\}$.

$$D_i(R) = \begin{cases} \top & \text{if } i = 0 \wedge R = \emptyset \\ s_i & \text{if } \exists\ (s_1 \in C_1, \dots, s_i \in C_i) : R \subseteq \Psi(s_1) \cup \dots \cup \Psi(s_i) \\ \bot & \text{otherwise} \end{cases}$$

Note that strictly speaking, D_i is not a function because, for some R, there might be multiple different values satisfying the definition (for example, if there are two different candidates that satisfy exactly the same requirements). Even in such a case, we suppose that D_i only has one value. As we will see, it does not matter which specific value it has, as long as it satisfies the definition.

Before using this function to solve the CSC problem, we need to know how to calculate its values effectively.

Lemma 11. *D_{i+1} can be computed recursively.*

$$D_{i+1}(R) = \begin{cases} s_{i+1} & \text{if } \exists\ K \subseteq R, s_{i+1} \in C_{i+1} : R \subseteq K \cup \Psi(s_{i+1}) \wedge D_i(K) \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

Proof. We suppose that the initial values of D_0 are taken directly from Definition 14 and prove the lemma for any $i \geq 1$. First, we show that if the recursion yields some value s_{i+1}, then it is one of the correct possible values of $D_{i+1}(R)$ according to Definition 14. Second, we show that if $D_i(R) \neq \bot$, then the recursion does not yield $\bot$ either.

If the recursion yields $s_{i+1} \in C_{i+1}$, then there is some $K \subseteq R$ such that $R \subseteq K \cup \Psi(s_{i+1}) \wedge D_i(K) \neq \bot$. From induction, the value of $D_i(K)$ is correct, thus there exist $(s_1 \in C_1, \ldots, s_i \in C_i) : K \subseteq \Psi(s_1) \cup \ldots \cup \Psi(s_i)$. Hence, $R \subseteq \Psi(s_1) \cup \ldots \cup \Psi(s_i) \cup \Psi(s_{i+1})$, which corresponds to Definition 14 and $D_{i+1}(R)$ yields a correct value.

If $D_{i+1}(R) = s_{i+1} \in C_{i+1}$, then there are some candidates $s_1 \in C_1, \ldots s_{i+1} \in C_{i+1}$ such that $R \subseteq \Psi(s_1) \cup \ldots \cup \Psi(s_{i+1})$. Let $K = \Psi(s_1) \cup \ldots \cup \Psi(s_i)$. Then, $R \subseteq K \cup \Psi(s_i + 1)$ and $D_i(K) = s_i$. Hence the recursion does not yield $\bot$. □

We continue by showing how a solution of the CSC problem may be extracted from the values of $D_1, \ldots, D_m$. We will use each function D_i to choose the candidate s_i from C_i.

Lemma 12. If $D_m(\mathcal{R}) = \bot$, then this instance of the CSC problem has no solution.

Proof. By the definition of D_m, we only have $D_m(\mathcal{R}) = \bot$ if no set of candidates $s_1 \in C_1, \ldots, s_m \in C_m$ exists, such that $\mathcal{R} \subseteq \Psi(s_1) \cup \ldots \cup \Psi(s_m)$. Thus, no solution of the CSC instance exists. □

Lemma 13. The CSC problem can be solved by dynamic programming. First, calculate all values of $D_1, \ldots, D_m$ using the recursion from Lemma 11. If $D_m(\mathcal{R}) = \bot$, then no solution exists. Otherwise, the solution can be found by iterating through the calculated values backwards and setting:

$$\begin{aligned}
s_m &= D_m(\mathcal{R}) \\
s_{m-1} &= D_{m-1}(\mathcal{R} \setminus \Psi(s_m)) \\
s_{m-2} &= D_{m-2}\Big(\mathcal{R} \setminus \big(\Psi(s_m) \cup \Psi(s_{m-1})\big)\Big) \\
&\ldots \\
s_i &= D_i(\mathcal{R} \setminus \bigcup_{j=i+1}^{m} \Psi(s_j)) \\
&\ldots \\
s_1 &= D_1(\mathcal{R} \setminus \bigcup_{j=2}^{m} \Psi(s_j))
\end{aligned}$$

Proof. By the definition of D_m, we can set $s_m = D_m(\mathcal{R})$. We know that given an $i \in \{1, \ldots, m-1\}$, all the requirements $\bigcup_{j=i+1}^{m} \Psi(s_j)$ are satisfied by $s_{i+1}, \ldots, s_m$, so the rest of them needs to be satisfied by $s_1, \ldots, s_i$. Also, by the definition of D_{i+1} we know that there exists $s_1, \ldots, s_i$ such that they satisfy the rest of the requirements. Then, by the definition of D_i, we have $D_i(\mathcal{R} \setminus \bigcup_{j=i+1}^{m} \Psi(s_j)) = s_i$. □

Finally, we propose Algorithm 3 to solve the CSC problem. It first computes all values for each of $D_1, \ldots, D_m$, and then constructs the solution $s_1, \ldots, s_m$ from them.

Algorithm 3: Constrained Set Cover

```
   Input: Set of requirements $\mathcal{R}$, sets of candidates $\mathcal{C} = C_1 \cup \ldots \cup C_m$,
          function $\Psi : \mathcal{C} \to 2^{\mathcal{R}}$
1  foreach $S \subseteq \mathcal{R}$ do
2      $D_0(S) \leftarrow \bot$
3  $D_0(\emptyset) \leftarrow \top$
4  for $i = 1, \ldots, m$ do
5      foreach $S \subseteq \mathcal{R}$ do
6          $D_i(S) \leftarrow \bot$
7      foreach $c \in C_i$ do
8          foreach $K \subseteq \mathcal{R}, F \subseteq \Psi(c)$ do
9              if $D_{i-1}(K) \neq \bot$ then
10                 $D_i(K \cup F) \leftarrow c$
11 if $D_m(\mathcal{R}) \neq \bot$ then
12     for $i = m, \ldots, 1$ do
13         $s_i \leftarrow D_i(\mathcal{R})$
14         $\mathcal{R} \leftarrow \mathcal{R} \setminus \Psi(s_i)$
15     return $(s_1, \ldots, s_m)$
16 else
17     No solution exists.
```

Lemma 14. Algorithm 3 works correctly and runs in $\mathcal{O}(2^{2|\mathcal{R}|}|\mathcal{R}| \cdot |\mathcal{C}|)$ time.

Proof. On lines 1–3 we set D_0 according to Definition 14. On lines 4–10 we compute $D_1, \ldots, D_m$ according to Lemma 11. On lines 11–17 we construct the solution from $D_1, \ldots, D_m$ according to Lemma 13.

The for-loop on line 4 iterates over all sets of candidates and the foreach-loop on line 7 iterates over all candidates in each set. Together, lines 8–10 will be executed $|\mathcal{C}|$ times. The foreach-loop on line 8 iterates over all subsets of $\mathcal{R}$ and all subsets of the output of Ψ, which sums to at most $\mathcal{O}(2^{|\mathcal{R}|} \cdot 2^{|\mathcal{R}|})$ iterations in total. Lines 9–10 can be implemented in $\mathcal{O}(|\mathcal{R}|)$ time. The for-loop on line 12 has m iterations, and lines 13–14 can be implemented in $\mathcal{O}(|\mathcal{R}|)$ time. $\square$

Now, we continue with the FPT algorithm for bipartite graphs. First, we discuss some properties of bipartite graphs in which a shortest path with eccentricity at most k exists.

Definition 15. Let $G = (U \cup V, E)$ be a bipartite graph with partitions U and V. Let P be a shortest path in G with $\mathrm{ecc}_G(P) \leq k$. We denote $L = P \cap U$ the vertices from partition U on the path P and $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_{|L|})$ the permutation/order of vertices from L in which they appear on the path P. We denote $\mathcal{R} = \{u \in U \mid d_G(u, P) = 1\}$ the set of vertices in partition U that are at distance 1 from P.

Observation 7. For every $i \in \{1, \ldots, |L| - 1\}$, there is exactly one vertex that lies on P between π_i and π_{i+1} on P. Moreover, this vertex is in the partition V.

Proof. There cannot be any vertex from U between π_i and π_{i+1} (otherwise $\boldsymbol{\pi}$ would not be the correct permutation of vertices from U on P). Because G is a bipartite graph, there must be exactly one vertex from V between π_i and π_{i+1}. $\square$

Definition 16. We use $C_i = \Big\{u \in V \mid \{\{\pi_i, u\}, \{u, \pi_{i+1}\}\} \subseteq E\Big\}$ for every $i \in \{1, \ldots, |L| - 1\}$ to denote the set of candidate vertices, one of which lies between π_i and π_{i+1}.

Intuitively, if we had the correct values of $\boldsymbol{\pi}$, we would only need to select the correct vertices from $C_1, \ldots, C_{|L|-1}$, one from each set, to solve the MESP problem. Later we will show that the problem of choosing these vertices is an instance of the CSC problem.

First, we define a function that will help us prove that the path constructed from the CSC solution will not have eccentricity higher than P.

Definition 17. We define function $e : U \cup V \to \mathbb{N}$ as follows.

$$e(u) = \min\left\{d_G(u, L),\ d_G(u, \mathcal{R}) + 1\right\}$$

Lemma 15. Function e is a good estimate of the distance from P:

1. $e(u) = d_G(u, P)$ for every $u \in U$, and
2. $e(u) = d_G(u, P \setminus \{u\})$ for every $u \in V$.

Proof. Clearly, if $e(u) = d_G(u, v)$ for some $v \in L \subseteq P$, then $d_G(u, P) \leq e(u)$. Clearly, if $e(u) = d_G(u, v) + 1$ for some $v \in \mathcal{R}$ neighbor of P, then $d_G(u, P) \leq e(u)$.

Let $y \in P$ be the nearest vertex to u on P. If $y \in U$, then $y \in L$ and $e(u) \leq d_G(u, y) = d_G(u, P)$. If $y \in V$ and $y = u$, i.e., $u \in P$, then u has a neighbor in L and $e(u) = 1 = d_G(u, P \setminus \{u\})$. If $y \in V$ and $y \neq u$, then let Q be the shortest path from u to y and let x be the last vertex on Q before y. Then, $x \in \mathcal{R}$ and $e(u) \leq d_G(u, x) + 1 = d_G(u, y) = d_G(u, P)$. $\square$

Now we are ready to show how the correct vertices from each C_i may be chosen by solving the CSC problem.

Lemma 16. Suppose that both endpoints of P are in U, and we have the corresponding values of $L, \boldsymbol{\pi}, \mathcal{R}$, as described in Definition 15. Let $(s_1, \ldots, s_{|L|-1})$ be a solution of the CSC problem with requirements $\mathcal{R}$, sets of candidates $\mathcal{C} = C_1 \cup \ldots \cup C_{|L|-1}$, and function $\Psi(v) = N_G(v)$. Consider the following path.

$$P' = (\pi_1, s_1, \ldots, \pi_i, s_i, \ldots, \pi_{|L|-1}, s_{|L|-1}, \pi_{|L|})$$

Then, P' is a shortest path and $\mathrm{ecc}_G(P') \leq k$.

Proof. By Observation 7, P' is a shortest path, since P is a shortest path.

All vertices in L are on both P and P'. All vertices in $\mathcal{R}$ are neighbors of P and, thanks to the way we chose $s_1, \ldots, s_{|L|-1}$, they are also neighbors of P'. Thus, $d_G(u, P') \leq e(u)$ for every $u \in U \cup V$. Lemma 15 says that $e(u) = d_G(u, P)$—and so $d_G(u, P') \leq d_G(u, P)$—for each $u \in U \cup (V \setminus P)$. For each $u \in V \cap P$ the distance from P' is at most 1. Moreover, if $P \neq P'$, then the distance of each vertex from $P \setminus P'$ to P' is also 1. ☐

Before constructing an algorithm, we need to discuss the cases where the path P has one or both endpoints in V. These endpoints are not contained in any set C_i and therefore, they are not present in the CSC either. We will handle these cases separately by trying all possible combinations for the first and last vertex on the path.

Observation 8. Suppose P has exactly one endpoint in V. Without loss of generality, let $u \in N_G(\pi_1)$ be an endpoint of P. Let $s_1, \ldots, s_{|L|-1}$ be a solution of the CSC problem with requirements $\mathcal{R}' = \mathcal{R} \setminus N_G(u)$, sets of candidates $\mathcal{C} = C_1 \cup \ldots \cup C_{|L|-1}$, and function $\Psi(v) = N_G(v)$. Consider the following path.

$$P' = (u, \pi_1, s_1, \ldots, \pi_{|L|-1}, s_{|L|-1}, \pi_{|L|})$$

Then, P' is a shortest path and $\mathrm{ecc}_G(P') \leq k$.

Proof. All vertices in L are on P', and, thanks to the way we chose $s_1, \ldots, s_{|L|-1}$, all vertices in $\mathcal{R}$ are neighbors of either u or one of $s_1, \ldots, s_{|L-1|}$. Thus, the same argument as in the proof of Lemma 16 applies. ☐

Observation 9. Analogically, if both endpoints of P are in V, P' may be constructed by removing both of their neighborhoods from $\mathcal{R}$ and adding them to the corresponding positions in P'.

Finally, we propose Algorithm 4 to solve the MESP problem. It finds the correct values for $L, \boldsymbol{\pi}, \mathcal{R}$, and the correct endpoints of P by trying all possible combinations. For each combination, it checks if the prospective solution of CSC with the current configuration would form a shortest path, and if yes, it tries to construct the CSC. If a solution of the CSC is found, it constructs a shortest path from it and checks its eccentricity, which might still be too high, if the current combination of values is incorrect.

Algorithm 4: FPT Bipartite

```
   Input: bipartite graph $G = (U \cup V, E)$
1  foreach $(\bar{L}, \bar{\boldsymbol{\pi}}, \bar{\mathcal{R}}) : \bar{L} \subseteq U, \bar{\boldsymbol{\pi}}$ is a permutation of $\bar{L}, \bar{\mathcal{R}} \subseteq (U \setminus \bar{L})$ do
2      for $i = 1, ..., |\bar{L}| - 1$ do
3          $C_i \leftarrow \Big\{u \in V \mid \{\{\bar{\pi}_i, u\}, \{u, \bar{\pi}_{i+1}\}\} \subseteq E\Big\}$
4      if $d_G(\bar{\pi}_1, \bar{\pi}_{|\bar{L}|}) \neq 2(|\bar{L}| - 1) \vee \exists i \in \{1, \ldots, |\bar{L}| - 1\} : C_i = \emptyset$ then
5          continue to the next $(\bar{L}, \bar{\boldsymbol{\pi}}, \bar{\mathcal{R}})$
6      let $s_1, \ldots, s_{|\bar{L}|-1}$ be the solution of CSC for requirements $\bar{\mathcal{R}}$,
         candidates $\mathcal{C} = C_1 \cup \ldots \cup C_{|\bar{L}|-1}$, and $\Psi(v) = N_G(v)$
7      if such a solution exists then
8          $P \leftarrow (\bar{\pi}_1, s_1, \ldots, \bar{\pi}_{|L|-1}, s_{|L|-1}, \bar{\pi}_{|L|})$
9          foreach $\bar{u} \in N_G[\bar{\pi}_1] \mid d_G(\bar{u}, \bar{\pi}_{|\bar{L}|}) = |P| \vee \bar{u} = \bar{\pi}_1$ do
10             $P' \leftarrow P$
11             if $\bar{u} \neq \bar{\pi}_1$ then
12                 $P' \leftarrow (\bar{u}) \frown P'$
13             foreach $\bar{v} \in N_G[\bar{\pi}_{|L|}] \mid d_G(\bar{u}, \bar{v}) = |P'| \vee \bar{v} = \bar{\pi}_{|\bar{L}|}$ do
14                 $P'' \leftarrow P'$
15                 if $\bar{v} \neq \bar{\pi}_{|\bar{L}|}$ then
16                     $P'' \leftarrow P'' \frown (\bar{v})$
17                 if $\mathrm{ecc}_G(P'') \leq k$ then
18                     return $P''$
19 A shortest path with eccentricity at most $k$ does not exist.
```

Lemma 17. Algorithm 4 finds a shortest path with eccentricity at most k if it exists. It reports correctly if such path does not exist.

Proof. If a shortest path P with $\mathrm{ecc}_G(P) \leq k$ exists, let u be the first vertex on P and v be the last vertex on P. All the following combinations will be tried in some iterations of the foreach-loop on line 1.

$$(\bar{L}, \bar{\boldsymbol{\pi}}, \bar{\mathcal{R}}) = \Big(L, \boldsymbol{\pi}, \mathcal{R}\Big) \tag{1}$$

$$(\bar{L}, \bar{\boldsymbol{\pi}}, \bar{\mathcal{R}}) = \Big(L, \boldsymbol{\pi}, \mathcal{R} \setminus \Psi(u)\Big) \tag{2}$$

$$(\bar{L}, \bar{\boldsymbol{\pi}}, \bar{\mathcal{R}}) = \Big(L, \boldsymbol{\pi}, \mathcal{R} \setminus \Psi(v)\Big) \tag{3}$$

$$(\bar{L}, \bar{\boldsymbol{\pi}}, \bar{\mathcal{R}}) = \Big(L, \boldsymbol{\pi}, \mathcal{R} \setminus (\Psi(u) \cup \Psi(v))\Big) \tag{4}$$

For each of these combinations, clearly $d_G(\bar{\pi}_1, \bar{\pi}_{|\bar{L}|}) = 2(|\bar{L}| - 1)$ and all of $C_1, \ldots, C_{|\bar{L}|-1}$ are non-empty. Also, for each of these combinations, the CSC clearly has a solution. There is an iteration of the foreach-loop on line 9 in which $\bar{u} = u$, and there is an iteration of the foreach-loop on line 13 in which $\bar{v} = v$. If both endpoints of P are in U, then by Lemma 16, the algorithm finds a solution for combination (1). If one of the endpoints of P is in U and the other is in V, then by Observation 8, the algorithm finds a solution for combination (2) or (3). If both endpoints of P are in V, then by Observation 9, the algorithm finds a solution for combination (4).

If the algorithm finds a solution, it is a shortest path thanks to the check on line 4, and its eccentricity is at most k thanks to the check on line 17. □

Lemma 18. Algorithm 4 runs in $\mathcal{O}(12^u u! \cdot n^4)$ time for $u = |U|$.

Proof. The for-loop on line 1 has at most $\mathcal{O}(3^u \cdot u!)$ iterations. The for-loop on lines 2–3 can be implemented in $\mathcal{O}(un)$ time. The CSC on line 6 can be solved by Algorithm 3 in $\mathcal{O}(2^{2u}un)$ time. Lines 7–19 can be implemented in $\mathcal{O}(n^2(n+m))$ time. □

3.4 Vertex Cover

In this section, we show that in order to construct an FPT algorithm parameterized by vertex cover, only a slight modification of Algorithm 4 is needed.

Observation 10. Let $G = (U \cup V, E)$ be a graph with vertex cover U. Then, there are no edges in $G[V]$, but, unlike in a bipartite graph, there may be edges in $G[U]$.

Proof. From Definition 7, every edge in G has one or both endpoints in U. □

Lemma 19 (Chen, Kanj, Xia [7])**.** The vertex cover of size u in a graph G can be found in $\mathcal{O}(1.2738^u + un)$ time.

Again, we start by discussing the properties of graphs in which a shortest path with eccentricity at most k exists. Note that the rest of this section is just a slight modification of Section 3.3.

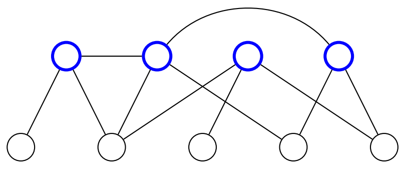

Figure 3.3: Example of a graph with vertex cover number 4. Vertices from the vertex cover are highlighted in blue. Note that all edges have at least one blue endpoint.

We use exactly the same definitions as in Section 3.3.

Definition 18. Let $G = (U \cup V, E)$ be a graph with vertex cover U and let P be a shortest path in G with $\mathrm{ecc}_G(P) \leq k$. We denote $L = P \cap U$ the vertices from the vertex cover on the path P and $\boldsymbol{\pi} = (\pi_1, \dots, \pi_{|L|})$ the permutation/order of vertices from L in which they appear on P. We denote $\mathcal{R} = \{u \in U \mid d_G(u, P) = 1\}$ the set of vertices in the vertex cover that are at distance 1 from P.

Observation 11. For every $i \in \{1, \dots, |L| - 1\}$: if π_i is connected to π_{i+1} by an edge, then there is no vertex between π_i and π_{i+1} on P. Otherwise, there is exactly one vertex that lies between π_i and π_{i+1} on P, and it is from V.

Proof. Clearly, there is no vertex from U between π_i and π_{i+1}. If $\{\pi_i, \pi_{i+1}\} \in E$ and some vertex u were between π_i and π_{i+1} on P, then P would not be a shortest path. If $\{\pi_i, \pi_{i+1}\} \notin E$, then there is exactly one vertex from V between them, as there are no edges in $G[V]$. □

Similarly as in Section 3.3, we need to select some vertices from V to put between the vertices in $\boldsymbol{\pi}$. However, this time we only need to select vertices between such π_i, π_{i+1} that $\{\pi_i, \pi_{i+1}\} \notin E$. To allow us to refer to these indices easily, we define the sequence $\boldsymbol{h}$.

Definition 19. We denote $\boldsymbol{h} = (h_1, \dots, h_\ell)$ the increasing sequence of all indices i such that $\{\pi_i, \pi_{i+1}\} \notin E$.

Now, we define the sets of candidate vertices, exactly the same way as in Section 3.3, but only for indices in $\boldsymbol{h}$.

Definition 20. We denote $C_{h_i} = \Big\{u \in V \mid \{\{\bar{\pi}_{h_i}, u\}, \{u, \bar{\pi}_{h_i+1}\}\} \subseteq E\Big\}$, the set of candidate vertices between π_{h_i} and π_{h_i+1}, for every $h_i \in \boldsymbol{h}$.

Intuitively, if we had the correct values of $\boldsymbol{\pi}$, we would only need to select the correct vertices from $C_{h_1}, \dots, C_{h_\ell}$, one from each set, to solve the MESP problem.

As in Section 3.3, we define function e and show that it is a good estimate of the distance from P.

Definition 21. We define function $e : U \cup V \to \mathbb{N}$ the same way as in Section 3.3.

$$e(u) = \min\left\{d_G(u, L),\ d_G(u, \mathcal{R}) + 1\right\}$$

Lemma 20. Function e is a good estimate of the distance from P:

1. $e(u) = d_G(u, P)$ for every $u \in U$, and
2. $e(u) = d_G(u, P \setminus \{u\})$ for every $u \in V$.

Proof. The proof is completely the same as the proof of Lemma 15. □

Now, we show how to construct a shortest path with eccentricity at most k by solving the CSC problem.

Lemma 21. Suppose that both endpoints of P are in U, and we have the corresponding values of $L, \boldsymbol{\pi}, \mathcal{R}$, as described in Definition 18. Let $s_{h_1}, \dots, s_{h_\ell}$ be a solution of the CSC problem with requirements $\mathcal{R}$, sets of candidates $\mathcal{C} = C_{h_1} \cup \ldots \cup C_{h_\ell}$, and function $\Psi(v) = N_G(v)$. Consider the following path.

$$P' = (\pi_1, \dots, \pi_{h_1}, s_{h_1}, \pi_{h_1+1}, \dots, \pi_{h_\ell}, s_{h_\ell}, \pi_{h_\ell+1}, \dots, \pi_{|L|})$$

Then, P' is a shortest path and $\mathrm{ecc}_G(P') \leq k$.

Proof. The proof is completely the same as in the proof of Lemma 16. □

As in Section 3.3, we need to discuss the cases in which the path P has one or both endpoints in V, before constructing an algorithm.

Observation 12. If P has one or both endpoints in V, we can find $s_{h_1}, \dots, s_{h_\ell}$ by subtracting the corresponding endpoint neighborhoods from $\mathcal{R}$ and solving the CSC. Then, by prepending/appending the endpoints to P' (constructed in the same way as in Lemma 21), we get a shortest path with $\mathrm{ecc}_G(P') \leq k$.

Proof. The proof is completely the same as the proof of Observation 8. □

Finally, we propose Algorithm 5 as a slight modification of Algorithm 4. The only difference is that candidate sets C_i with $\{\pi_i, \pi_{i+1}\} \in E$ are not considered as π_i must be directly connected to π_{i+1} in P (see lines 4–7, 10, 12), and the correct shortest path distance is modified (see line 8).

Lemma 22. Algorithm 5 finds a shortest path with eccentricity at most k if it exists. It reports correctly if such path does not exist.

Proof. The proof is completely the same as the proof of Lemma 17. □

Lemma 23. Algorithm 5 runs in $\mathcal{O}(12^u u! \cdot n^4)$ time for $|U| = u$.

Proof. The proof is completely the same as the proof of Lemma 18. □

Algorithm 5: FPT Vertex Cover

```
   Input: graph $G = (U \cup V, E)$ with vertex cover $U$
1  foreach $(\bar{L}, \bar{\boldsymbol{\pi}}, \bar{\mathcal{R}}) : \bar{L} \subseteq U, \bar{\boldsymbol{\pi}}$ is a permutation of $\bar{L}, \bar{\mathcal{R}} \subseteq (U \setminus \bar{L})$ do
2      $\ell \leftarrow 0$
3      for $i = 1, ..., |\bar{L}| - 1$ do
4          if $\{\bar{\pi}_i, \bar{\pi}_{i+1}\} \notin E$ then
5              $C_i \leftarrow \Big\{u \in V \mid \{\{\bar{\pi}_i, u\}, \{u, \bar{\pi}_{i+1}\}\} \subseteq E\Big\}$
6              $\ell \leftarrow \ell + 1$
7              $h_\ell \leftarrow i$
8      if $d_G(\bar{\pi}_1, \bar{\pi}_{|\bar{L}|}) \neq \sum_{i=1}^{|\bar{L}|-1} d_G(\bar{\pi}_i, \bar{\pi}_{i+1}) \vee \exists i \in \{h_1, \dots, h_\ell\} : C_i = \emptyset$
        then
9          continue to the next $(\bar{L}, \bar{\boldsymbol{\pi}}, \bar{\mathcal{R}})$
10     let $s_{h_1}, \dots, s_{h_\ell}$ be the solution of CSC for requirements $\bar{\mathcal{R}}$,
        candidates $\mathcal{C} = C_{h_1} \cup \ldots \cup C_{h_\ell}$, and $\Psi(v) = N_G(v)$
11     if such a solution exists then
12         $P := (\bar{\pi}_1, \dots, \bar{\pi}_{h_1}, s_{h_1}, \bar{\pi}_{h_1+1}, \dots, \bar{\pi}_{h_\ell}, s_{h_\ell}, \bar{\pi}_{h_\ell+1}, \dots, \bar{\pi}_{|L|})$
13         foreach $\bar{u} \in N_G[\bar{\pi}_1] \mid d_G(\bar{u}, \bar{\pi}_{|\bar{L}|}) = |P| \vee \bar{u} = \bar{\pi}_1$ do
14             $P' \leftarrow P$
15             if $\bar{u} \neq \bar{\pi}_1$ then
16                 $P' \leftarrow (\bar{u}) \frown P'$
17             foreach $\bar{v} \in N_G[\bar{\pi}_{|L|}] \mid d_G(\bar{u}, \bar{v}) = |P'| \vee \bar{v} = \bar{\pi}_{|\bar{L}|}$ do
18                 $P'' \leftarrow P'$
19                 if $\bar{v} \neq \bar{\pi}_{|\bar{L}|}$ then
20                     $P'' \leftarrow P'' \frown (\bar{v})$
21                 if $\mathrm{ecc}_G(P'') \leq k$ then
22                     return $P''$
23 A shortest path with eccentricity at most $k$ does not exist.
```

3.5 Twin Cover

In this section, we show that, in fact, Algorithm 5 from the previous section works correctly, even if the set U from its input graph $G = (U \cup V, E)$ is a twin cover and not a vertex cover. Thus, it is parameterized by twin cover. We do not consider the trivial case of a graph with twin cover number 0. A connected graph with twin cover number 0 is a clique, and any single edge in such a graph is the minimum eccentricity shortest path.

Observation 13. Let $G = (U \cup V, E)$ be a graph with twin cover U. Then, there may even be edges in $G[V]$. However, for every edge $\{u, v\}$ in $G[V]$, u and v are twins.

Lemma 24 (Ganian [8])**.** The twin cover of size u in a graph G can be found in $\mathcal{O}(1.2738^u + un + nm)$ time.

We will use the same definitions as in Section 3.4 and show that the claims made about a graph with vertex cover U also hold for a graph with twin cover U.

Definition 22. Let $G = (U \cup V, E)$ be a graph with twin cover U and let P be a shortest path in G with $\mathrm{ecc}_G(P) \leq k$. We denote $L = P \cap U$ the vertices from the twin cover on the path P and $\boldsymbol{\pi} = (\pi_1, \dots, \pi_{|L|})$ the permutation/order of vertices from L in which they appear on P. We denote $\mathcal{R} = \{u \in U \mid d_G(u, P) = 1\}$ the set of vertices in the twin cover that are at distance 1 from P.

We denote $\boldsymbol{h} = (h_1, \dots, h_\ell)$ the increasing sequence of all indices i such that $\{\pi_i, \pi_{i+1}\} \notin E$.

We denote $C_{h_i} = \Big\{u \in V \mid \{\{\bar{\pi}_{h_i}, u\}, \{u, \bar{\pi}_{h_i+1}\}\} \subseteq E\Big\}$, the set of candidate vertices between π_{h_i} and π_{h_i+1}, for every $h_i \in \boldsymbol{h}$.

Observation 14. For every $i \notin \boldsymbol{h}$, there is no vertex between π_i and π_{i+1} on P. For every $h_i \in \boldsymbol{h}$, there is exactly one vertex from C_{h_i} that lies between π_{h_i} and π_{h_i+1} on P.

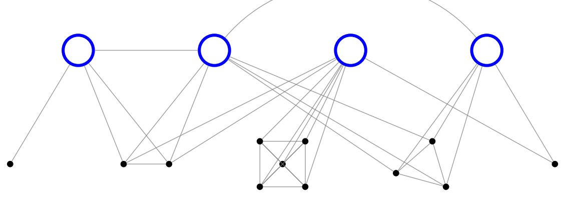

Figure 3.4: Example of a graph with twin cover number 4. Vertices from the twin cover are highlighted in blue. Note that all edges either have a blue endpoint, or their endpoints are twins.

Proof. Clearly, there is no vertex between π_i and π_{i+1} if $\{\pi_i, \pi_{i+1}\} \in E$ and there may only be vertices from V between π_{h_i} and π_{h_i+1} if $\{\pi_{h_i}, \pi_{h_i+1}\} \notin E$. The only edges in $G[V]$ are between twins. If there were two twins on the same path, it would not be a shortest path. □

Definition 23. We define function $e : U \cup V \to \mathbb{N}$ the same way as in Section 3.3.

$$e(u) = \min \{d_G(u, L),\ d_G(u, \mathcal{R}) + 1\}$$

Lemma 25. Function e is a good estimate of the distance from P:

1. $e(u) = d_G(u, P)$ for every $u \in U$, and
2. $e(u) = d_G(u, P \setminus \{u\})$ for every $u \in V$.

Proof. Clearly, $d_G(u, P) \leq e(u)$ for every $u \in U \cup V$.

Let $y \in P$ be the nearest vertex to u on P. If $y \in U$, then $y \in L$ and $e(u) \leq d_G(u, y) = d_G(u, P)$. If $y \in V$ and $y = u$ or y is a twin of u, then u has a neighbor in L and $e(u) = 1 = d_G(u, P \setminus \{u\})$. If $y \in V$ and $y \neq u$ and y is not a twin of u, then let Q be the shortest path from u to y and let x be the last vertex on Q before y. Then, $x \in \mathcal{R}$ and $e(u) \leq d_G(u, x) + 1 = d_G(u, y) = d_G(u, P)$. □

Now, we know that even if the set U is a twin cover and not a vertex cover, there is still exactly one vertex from each C_{h_i} between π_{h_i} and π_{h_i+1}. Furthermore, we know that the function e still estimates the distance from P correctly for all vertices other than those in $P \cap V$ (for which it estimates distance 1 instead of 0). Again, by constructing an instance of the CSC and creating path P' from it the same way as in Section 3.4, we can make sure that $d_G(u, P') \leq e(u)$ for every $u \in U \cup V$.

Lemma 26. Algorithm 5 works correctly and with the same time complexity even for a graph $G = (U \cup V, E)$ such that U is a twin cover of G, instead of vertex cover.

Proof. Thanks to Observation 14 and Lemma 25, the proof is exactly the same as the proofs of Lemma 22 and Lemma 23. □

3.6 Distance to Cluster

In this section, we present an FPT algorithm for the MESP problem parameterized by the distance to cluster. Again, this algorithm will only be a slight modification of Algorithm 5 from Section 3.4. We do not consider the trivial case of a graph with distance to cluster 0. A connected graph with distance to cluster 0 is a clique, and any single edge in such a graph is the minimum eccentricity shortest path.

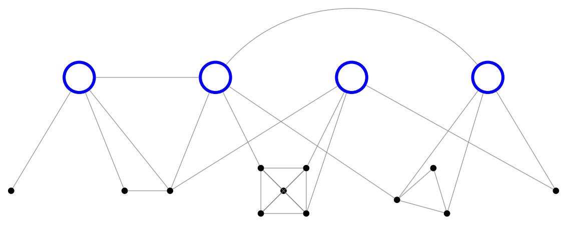

Figure 3.5: Example of a graph with distance to cluster 4. Vertices from the modulator to cluster are highlighted in blue. Note that the non-blue twin vertices form clusters, i.e., cliques.

Observation 15. Let $G = (U \cup V, E)$ be a graph with modulator to cluster U. Then, for any edge $\{u, v\}$ in $G[V]$, u and v are twins in $G[V]$.

Lemma 27 (Boral et al. [9])**.** A modulator to cluster of a graph with distance to cluster u can be found in $\mathcal{O}(1.9102^u \cdot (n + m))$ time.

Observation 16. Let $G = (U \cup V, E)$ be a graph with modulator to cluster U and let P be a shortest path with $\mathrm{ecc}_G(P) = k$. Then, there exists a shortest path P' such that it contains at least one vertex from U and $\mathrm{ecc}_G(P') \leq k$.

Proof. Suppose that P only contains vertices from V. All these vertices form a clique, so the length of P is at most 1. Let $P = (u, v)$. If there is some vertex $w \in U$ such that it is a neighbor of exactly one endpoint of P, then either $P' = (u, v, w)$ or $P' = (w, u, v)$. If all vertices in U are neighbors of both u and v, then $P' = (v, w)$ for any $w \in U$. □

We will use similar definitions as in the previous sections. The main difference between the following definition and Definition 22 from Section 3.5 is that instead of one requirements set $\mathcal{R}$, we define two sets: $\mathcal{R}_1$ and $\mathcal{R}_2$. Also, we require that the path P contains a vertex from U, and we skip the definition of sets C_i, which will be defined separately.

Definition 24. Let $G = (U \cup V, E)$ be a graph with modulator to cluster U and let P be a shortest path in G with $\mathrm{ecc}_G(P) \leq k$ that contains at least one vertex from the modulator to cluster U. We denote $L = P \cap U$ the vertices from the modulator to cluster on P and $\boldsymbol{\pi} = (\pi_1, \dots, \pi_{|L|})$ the permutation/order of vertices from L in which they appear on P. We denote $\mathcal{R}_i = \{u \in U \mid d_G(u, P) = i\}$ for $i \in \{1, 2\}$.

We denote $\boldsymbol{h} = (h_1, \dots, h_\ell)$ the increasing sequence of all indices i such that $\{\pi_i, \pi_{i+1}\} \notin E$.

We start by observing that even if U is a modulator to cluster of G, the alternatives for which vertices may appear between each π_i and π_{i+1} on P are still limited.

Observation 17. $d_G(\pi_i, \pi_{i+1}) \leq 3$ for each $i \in \{1, \ldots, |L| - 1\}$.

Proof. The shortest path between π_i and π_{i+1} consists of only vertices from V (except for the endpoints). All these vertices must be twins in $G[V]$ and thus form a clique in $G[V]$. There are at most two such vertices on a shortest path. If there were three vertices from the same clique on a path, it would not be a shortest path. □

Observation 18. For every $i \notin \boldsymbol{h}$, there is no vertex between π_i and π_{i+1} on P. For every $h_i \in \boldsymbol{h}$: If $d_G(\pi_{h_i}, \pi_{h_i+1}) = 2$, then there is one vertex on P between π_{h_i} and π_{h_i+1}, and it is from V. If $d_G(\pi_{h_i}, \pi_{h_i+1}) = 3$, then there are two vertices on P between π_{h_i} and π_{h_i+1}, and both are from V.

Proof. Clearly, there are no vertices from U between π_i and π_{i+1}. If the number of vertices between π_i and π_{i+1} did not correspond to the distance between π_i and π_{i+1}, then P would not be a shortest path. □

Now, we define the sets of candidate vertices C_{h_i}. The definition is similar to those in the previous sections. However, this time, we need to consider pairs of vertices, instead of single vertices, since there might be up to two vertices between each π_{h_i} and π_{h_i+1}.

Definition 25. We define the set C_{h_i} of candidate vertices between π_{h_i} and π_{h_i+1} for each $h_i \in \boldsymbol{h}$.

$$C_{h_i} = \begin{cases} \Big\{(u,u) \in V^2 \mid \{\{\pi_{h_i}, u\}, \{u, \pi_{h_i+1}\}\} \subseteq E\Big\} & \text{if } d_G(\pi_{h_i}, \pi_{h_i+1}) = 2 \\ \Big\{(u,v) \in V^2 \mid \{\,\{\pi_{h_i}, u\}\,, \{u,v\}, \{v, \pi_{h_i+1}\}\} \subseteq E\Big\} & \text{if } d_G(\pi_{h_i}, \pi_{h_i+1}) = 3 \\ \emptyset & \text{otherwise} \end{cases}$$

For $h_i \in \boldsymbol{h}$ with $d_G(\pi_{h_i}, \pi_{h_i+1}) = 2$, the set C_i contains pairs of the same vertices (u, u). To avoid adding some vertex into a path twice, we define the following function.

Definition 26. Function μ maps a pair of any two elements to a sequence of length 1 or 2.

$$\mu(u,v) = \begin{cases} (u) & \text{if } u = v \\ (u,v) & \text{if } u \neq v \end{cases}$$

Again, to solve the MESP problem, we need to choose one pair from each of $C_{h_1}, \ldots, C_{h_\ell}$ and satisfy all the requirements $\mathcal{R}_1 \cup \mathcal{R}_2$. To prove that a path constructed from vertices that satisfy all the requirements will have eccentricity at most k, we need to redefine function e.

Definition 27. We define function $e : U \cup V \to \mathbb{N}$ as follows.

$$e(u) = \min\{d_G(u, L),\ d_G(u, \mathcal{R}_1) + 1,\ d_G(u, \mathcal{R}_2) + 2\}$$

Lemma 28. Function e is a good estimate of the distance from P:

1. $e(u) = d_G(u, P)$ for every $u \in U$, and
2. $e(u) = d_G(u, P \setminus (N_G[u] \cap V))$ for every $u \in V$.

Proof. Clearly, $d_G(u, P) \leq d_G(u, P \setminus (N_G[u] \cap V)) \leq e(u)$.

Let z be the nearest vertex to u on P and Q be the shortest path from u to z. If there are any vertices from U on Q, let x be the last vertex from U on Q. We know that $d_G(x, z) \leq 2$ because Q is a shortest path and all vertices connected in $G[V]$ form a clique. If $x = z$, then $x \in L$. If $d_G(x, z) = 1$, then $x \in \mathcal{R}_1$. If $d_G(x, z) = 2$, then $x \in \mathcal{R}_2$. Hence, $e(u) \leq d_G(u, z) = d_G(u, P)$. If Q consists only of vertices from V, let s be the nearest vertex to u such that $s \in P \setminus (N_G[u] \cap V)$. Clearly, $s \in L$ and $e(u) \leq d_G(u, s) = d_G(u, P \setminus (N_G[u] \cap V))$. □

Now, we are ready to show how to find a shortest path with eccentricity at most k by solving the MESP problem.

Lemma 29. Suppose that both endpoints of P are in U, and we have the corresponding values of $L, \pi, \mathcal{R}_1, \mathcal{R}_2$ as described in Definition 24. Let $(s_{h_1}, \ldots, s_{h_\ell})$ be a solution of the CSC problem with requirements $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$, sets of candidates $\mathcal{C} = C_{h_1} \cup \ldots \cup C_{h_\ell}$, and function $\Psi(u, v) = N_G(u) \cup N_G(v) \cup \Big((N_G^2[u] \cup N_G^2[v]) \cap \mathcal{R}_2\Big)$. Consider the following path.

$$\begin{aligned} P' = {} & (\pi_1, \ldots, \pi_{h_1}) \frown \mu(s_{h_1}) \frown (\pi_{h_1+1}, \ldots, \pi_{h_2}) \\ & \ldots \\ & \frown \mu(s_{h_i}) \frown (\pi_{h_i+1}, \ldots, \pi_{h_{i+1}}) \frown \mu(s_{h_{i+1}}) \\ & \ldots \\ & \frown (\pi_{h_{\ell-1}+1}, \ldots, \pi_{h_\ell}) \frown \mu(s_{h_\ell}) \frown (\pi_{h_\ell+1}, \ldots, \pi_{|L|}) \end{aligned}$$

Then, P' is a shortest path and $\mathrm{ecc}_G(P') \leq \max\{2, k\}$.

Proof. Clearly, P' is a shortest path.

Thanks to the way we chose $s_{h_1}, \ldots s_{h_\ell}$ and from Lemma 28 we know that:

1. for every $u \in U : d_G(u, P') \leq e(u) = d_G(u, P)$,
2. for every $u \in V : d_G(u, P') \leq e(u) = d_G(u, P \setminus (N_G[u] \cap V))$.

If $u \in V$ and $P \cap (N_G[u] \cap V) \neq \emptyset$, then $d_G(u, P) \leq 1$. Because P contains at least one vertex from U and all vertices in $N_G[u] \cap V$ form a clique, $d_G(u, P \setminus (N_G[u] \cap V)) \leq 2$. □

Clearly, if $k \geq 2$, then we can use Lemma 29 to construct a shortest path with eccentricity at most k. Now, we discuss the case when $k = 1$.

Observation 19. If $\mathrm{ecc}_G(P) = 1$, then for every $u \in U \cup V : e(u) \leq 2$.

Proof. If $u \in U$, then either $u \in L$ or $u \in \mathcal{R}_1$, and $e(u) \leq 1$. For each $u \in V$, let $v \in L$ be the nearest vertex to u on P from U. Because all vertices that are connected in $G[V]$ form a clique, $d_G(u, v) \leq 2$, hence $e(u) \leq 2$. □

Corollary 4. If $\mathrm{ecc}_G(P) = 1$, then a path P' with $\mathrm{ecc}_G(P') \leq 1$ can be constructed similarly as in Lemma 29 but for each candidate set C_i which contains some pair $(x, y) \in V^2$ such that there is a neighbor $z \in V$ of x (and of y, they are twins in $G[V]$) with $e(z) = 2$, removing every $(u, v) \in V^2$ such that z is *not* a neighbor of u (and v) from C_i.

Proof. For any C_i, if any of the removed pairs were selected into P', then the distance of z to P' would be $d_G(z, P') = 2$ and therefore $\mathrm{ecc}_G(P') > 1$. □

We have shown how to construct a shortest path with eccentricity at most k by solving the CSC problem, even if $k = 1$. Finally, we observe that such a path can be constructed even if one or both of its endpoints are in V.

Observation 20. If P has an endpoint $s \in V$, its neighbor $t \in P$ might also be in V. Let $P = (s, t, \ldots)$. Similarly as in Observation 8, we may obtain P' with $\mathrm{ecc}_G(P') \leq k$ by removing $\Psi(s)$ (and $\Psi(t)$ if $t \in V$) from $\mathcal{R}$, finding $s_{h_1}, \ldots, s_{h_\ell}$ by solving the CSC, and preprending s (and t if $t \in V$) to P'.

We propose Algorithm 6 to solve the MESP problem. It is very similar to Algorithm 5 from Section 3.4. Before presenting the pseudocode, we list the only four differences.

1. The distance between each π_i and π_{i+1} may be not only 1 or 2, but also 3. Thus, the candidate sets $C_{h_1}, \ldots, C_{h_\ell}$ are not sets of individual vertices, but sets of pairs of vertices instead. If $d_G(\pi_i, \pi_{i+1}) = 2$, then each pair in C_i contains the same vertex twice.

2. If the desired eccentricity k is 1, there is a special condition that the candidate pairs must satisfy (see Corollary 4).

3. The CSC requirements are still vertices from U that need to be at some distance from P. However, they are further separated into two distinct subsets: $\mathcal{R}_1$ (vertices at distance 1 from P) and $\mathcal{R}_2$ (vertices at distance 2 from P). The Ψ function is updated accordingly.

4. All combinations must be tried not only for the first and last vertex on P, but also for the second and second-to-last, as all of these may be in V.

Algorithm 6: FPT Distance to Cluster

```
   Input: graph $G = (U \cup V, E)$ with modulator to cluster $U$
1  foreach $(\bar{L}, \bar{\boldsymbol{\pi}}, \bar{\mathcal{R}}) : \bar{L} \subseteq U, \bar{\boldsymbol{\pi}}$ is a permutation of $\bar{L}$, $\bar{\mathcal{R}} = \bar{\mathcal{R}}_1 \cup \bar{\mathcal{R}}_2 \subseteq (U \setminus \bar{L})$ do
2      $\ell \leftarrow 0$
3      for $i = 1, ..., |\bar{L}| - 1$ do
4          if $\{\bar{\pi}_i, \bar{\pi}_{i+1}\} \notin E$ then
5              if $d_G(\bar{\pi}_i, \bar{\pi}_{i+1}) = 2$ then
6                  $C_i \leftarrow \Big\{(u,u) \in V^2 \mid \{\{\bar{\pi}_i, u\}, \{u, \bar{\pi}_{i+1}\}\} \subseteq E\Big\}$
7              else
8                  $C_i \leftarrow \Big\{(u,v) \in V^2 \mid \{\ \{\bar{\pi}_i, u\}\ , \{u,v\}, \{v, \bar{\pi}_{i+1}\}\} \subseteq E\Big\}$
9              if $k = 1$ then
10                 $K \leftarrow \{z \in V \mid e(z) = 2 \wedge \exists (x,y) \in C_i : \{x,z\} \in E\}$
11                 $C_i \leftarrow C_i \setminus \{(u,v) \in C_i \mid \exists z \in K : \{u,z\} \notin E\}$
12             $\ell \leftarrow \ell + 1$
13             $h_\ell \leftarrow i$
14     if $d_G(\bar{\pi}_1, \bar{\pi}_{|\bar{L}|}) \neq \sum_{i=1}^{|\bar{L}|-1} d_G(\bar{\pi}_i, \bar{\pi}_{i+1}) \vee \exists i \in \{h_1, \ldots, h_\ell\} : C_i = \emptyset$ then
15         continue to the next $(\bar{L}, \bar{\boldsymbol{\pi}}, \bar{\mathcal{R}})$
16     let $s_{h_1}, \ldots, s_{h_\ell}$ be the solution of CSC for: requirements $\bar{\mathcal{R}}$,
         candidates $\mathcal{C} = C_{h_1} \cup \ldots \cup C_{h_\ell}$, and
         $\Psi(u,v) = N_G(u) \cup N_G(v) \cup \Big(\big(N_G^2[u] \cup N_G^2[v]\big) \cap \mathcal{R}_2\Big)$
17     if such a solution exists then
18         $P \leftarrow$ empty sequence
19         for $i = 1, \ldots, |\bar{L}| - 1$ do
20             $P \leftarrow P \frown \bar{\pi}_i$
21             if $\{\bar{\pi}_i, \bar{\pi}_{i+1}\} \notin E$ then
22                 $P \leftarrow P \frown \mu(s_i)$
23         foreach $\bar{t} \in N_G[\bar{\pi}_1], \bar{s} \in N_G[\bar{t}], \bar{u} \in N_G[\bar{\pi}_{|\bar{L}|}], \bar{v} \in N_G[\bar{u}]$ do
24             if $d_G(\bar{s}, \bar{v}) \neq d_G(\bar{s}, \bar{t}) + d_G(\bar{t}, \bar{\pi}_1) + d_G(\bar{\pi}_1, \bar{\pi}_{|\bar{L}|}) + d_G(\bar{\pi}_{|\bar{L}|}, \bar{u}) + d_G(\bar{u}, \bar{v})$
                 then continue
25             $P' \leftarrow P$
26             if $\bar{t} \neq \bar{\pi}_1$ then $P' \leftarrow (\bar{t}) \frown P'$
27             if $\bar{s} \neq \bar{t}$ then $P' \leftarrow (\bar{s}) \frown P'$
28             if $\bar{u} \neq \bar{\pi}_{|\bar{L}|}$ then $P' \leftarrow P' \frown (\bar{u})$
29             if $\bar{v} \neq \bar{u}$ then $P' \leftarrow P' \frown (\bar{v})$
30             if $\mathrm{ecc}_G(P') \leq k$ then
31                 return $P'$
32 A shortest path with eccentricity at most $k$ does not exist.
```

Lemma 30. Algorithm 6 finds a shortest path with eccentricity at most k if one exists, and reports if it does not exist.

Proof. The proof is very similar to the proof of Lemma 17. If a shortest path P with $\mathrm{ecc}_G(P) \leq k$ exists, let s be the first vertex on P, t the second vertex on P, u the next-to-last vertex on P, and v the last vertex on P. There will be an iteration of the foreach-loop on line 1 in which $(\bar{L}, \bar{\boldsymbol{\pi}}, \bar{\mathcal{R}}) = (L, \boldsymbol{\pi}, \mathcal{R})$, as well as iterations in which $\bar{\mathcal{R}}$ equals $\mathcal{R}$ without the requirements satisfied by any combination of s, t, u, and v. By Lemma 29 and Observation 20, there will be a solution of the CSC for some of the combinations (depending on which of s, t, u, v are in V). Then, there will be an iteration of the foreach-loop on line 23 in which $\bar{s} = s, \bar{t} = t, \bar{u} = u$, and $\bar{v} = v$. By Corollary 4, in this iteration the path P' will have $\mathrm{ecc}_G(P') \leq k$, even if $k = 1$.

If the algorithm finds a solution, it is a shortest path thanks to the check on line 14, and its eccentricity is at most k thanks to the check on line 30. □

Lemma 31. Algorithm 6 runs in $\mathcal{O}(2^{4u}u!u \cdot n^6)$ time for $u = |U|$.

Proof. The foreach-loop on line 1 has at most $\mathcal{O}(4^u u!)$ iterations. The for-loop on line 3 has at most $\mathcal{O}(u)$ iterations and lines 4–13 can be implemented in $\mathcal{O}(n + m)$ time. The CSC on line 16 can be solved by Algorithm 3 in $\mathcal{O}(2^{2u}un^2)$ time as $|\mathcal{R}| \leq u$ and $|\mathcal{C}| \leq n^2$. Lines 17–31 can be implemented in $\mathcal{O}(n^4(n + m))$ time. □

3.7 Distance to Disjoint Paths

In this section, we present an FPT algorithm for the MESP problem parameterized by the distance to disjoint paths and the minimum eccentricity, combined.

Lemma 32. The modulator to disjoint paths C of a graph G with distance to disjoint paths c can be found in $\mathcal{O}(3^c(n + m))$ time.

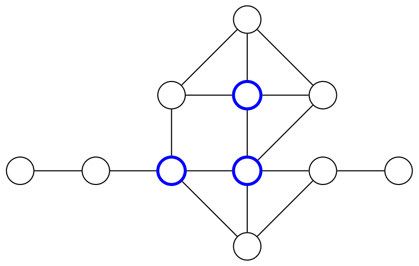

Figure 3.6: Example of a graph with distance to disjoint paths 3. Vertices from the modulator to disjoint paths are highlighted in blue.

Proof. If the highest degree in G is at most 2, then G consists only of disjoint paths and cycles, and the modulator to disjoint paths is a set of vertices containing one vertex from each cycle. Thus, the modulator to disjoint paths can be found in $\mathcal{O}(n+m)$ time by identifying all cycles with a depth-first search.

If the highest degree in G is at least 3, then the modulator to disjoint paths can be found by a simple branching rule.

1. Select any vertex u with $\deg_G(u) \geq 3$.
2. Either $u \in C$ or some subset $S \subseteq N_G(u)$ of size $|S| = \deg_G(u) - 2$ must be in C.

For a given maximum distance to disjoint paths c, this yields the following time complexity recursion.

$$\begin{aligned} T(0) &= \mathcal{O}(n+m) \\ T(c) &\leq 3T(c-1) \\ &\leq 3^c\mathcal{O}(n+m) \end{aligned}$$

□

We start by discussing some properties of graphs in which a shortest path P with $\mathrm{ecc}_G(P) \leq k$ does exist.

Definition 28. Let $P = (p_1, \ldots, p_{|P|})$ be a shortest path in G with $ecc_G(P) \leq k$. Let $\widehat{C} = C \cup \{p_1, p_{|P|}\}$. Let $L = P \cap \widehat{C}$. We denote $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_{|L|})$ the permutation/order of vertices from L on the path P. We define function $e(v) = d_G(v, P)$ for every $v \in V$.

Observation 21. For each consecutive pair of vertices $\pi_i, \pi_{i+1} \in L$, there may be multiple shortest paths connecting them, such that they do not contain any other vertices from $\widehat{C}$. Exactly one of these shortest paths is contained in P for each pair.

Proof. If there was some vertex from $\widehat{C}$ between π_i and π_{i+1} on P, then there would have to be some π_j between π_i and π_{i+1} in $\boldsymbol{\pi}$. □

Definition 29. We say $\bar{\sigma}$ is a candidate segment if it is a sequence of vertices on some shortest path from π_i to π_{i+1} excluding the endpoints π_i, π_{i+1} and $\bar{\sigma} \cap \widehat{C} = \emptyset$. We define $\Sigma(\pi_i, \pi_{i+1})$ as a set of all candidate segments $\bar{\sigma}$ between π_i and π_{i+1}. We denote $\widetilde{\Sigma} = \bigcup_{i=1}^{|L|-1} \Sigma(\pi_i, \pi_{i+1})$ the set of all candidate segments in G. We say that a candidate segment $\sigma \in \widetilde{\Sigma}$ is a true segment if $\sigma \subseteq P$.

Intuitively, if we had the correct values of $\boldsymbol{\pi}$, we would only need to select the true segment out of each $\Sigma(\pi_i, \pi_{i+1})$ for $i \in \{1, \ldots, |L|-1\}$, in order to construct the path P.

To select the true segments, we will need the following function, which estimates the distance from a vertex to the path P.

Definition 30 (estimate distance to P)**.** For a graph $G = (V, E)$, a set of vertices $\widehat{C} \subseteq V$ and a function $e : \widehat{C} \to \mathbb{N}$ we define $d^e_G : V \times 2^{\widehat{C}} \to \mathbb{N}$ as

$$d^e_G(v, S) = \min_{s \in S} d_G(v, s) + e(s).$$

Observation 22. If e corresponds to the shortest path P in G, then for every $v \in V : d_G(v, P) \leq d^e_G(v, \widehat{C})$. In particular, if $d^e_G(v, \widehat{C}) \leq k$, then $d_G(v, P) \leq k$.

Proof. By definition, for any $v \in V$, there is some $s \in \widehat{C}$ such that $d^e_G(v, \widehat{C}) = d_G(v, s) + e(s) = d_G(v, s) + d_G(s, P)$ and, from triangle inequality, $d_G(v, P) \leq d_G(v, s) + d_G(s, P)$. □

If we had the correct values for the permutation $\boldsymbol{\pi}$ of vertices from $\widehat{C}$ that are on P, we would still have to take care of those vertices $v \in V$ with $d^e_G(v, \widehat{C}) > k$, in order to solve the MESP problem. In particular, we would have to choose a segment from each $\Sigma(\pi_i, \pi_{i+1})$ in a way that for every vertex v with $d^e_G(v, \widehat{C}) > k$, there would be some chosen segment at distance at most k from v. We express this more formally in the following definition.

Definition 31. A candidate segment $\bar{\sigma} \in \widetilde{\Sigma}$ *satisfies* $v \in V \setminus \widehat{C}$ if the following inequality holds.

$$d_G(v, \bar{\sigma}) \leq k < d^e_G(v, \widehat{C})$$

We continue by showing that the number of vertices v with $d^e_G(v, \widehat{C}) > k$ which *do not* lie on P is bounded by the size of L.

Lemma 33. Let $\bar{\sigma} \in \widetilde{\Sigma}$ be a segment and $D = \{v \in V \setminus P \mid \bar{\sigma} \text{ satisfies } v\}$. Then $|D| \leq 2$.

Proof. Let $v \in D$ and $u \in \bar{\sigma}$ be the nearest vertex to v on segment $\bar{\sigma}$. There is a shortest path from u to v which does not contain any vertex from $\widehat{C}$ (if it did, then $d^e_G(v, \widehat{C}) = d_G(v, P) \leq k$). Because $G \setminus C$ is a union of disjoint paths and it contains the whole segment $\bar{\sigma}$ as well as the path from u to v, these two paths must be connected through their endpoints. Neither of the endpoints is in C, so no more than two such connections can be present in G. (see Figure 3.7) □

Corollary 5. Let $U = \{v \in V \setminus P \mid d^e_G(v, \widehat{C}) > k\}$. There are $|L| - 1$ true segments, therefore $|U| \leq 2(|L| - 1)$.

We have shown that there are not many vertices $v \notin P$ with $d^e_G(v, \widehat{C}) > k$. Now, we show that all such vertices actually have $d^e_G(v, \widehat{C}) = k + 1$.

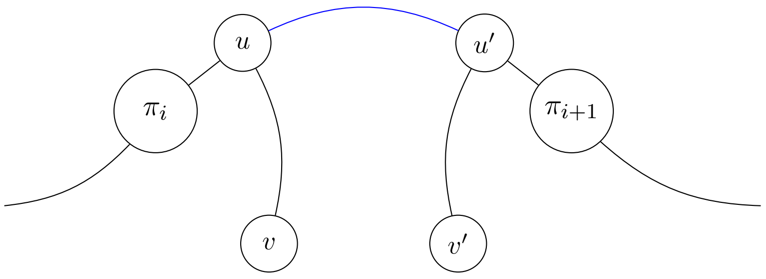

Figure 3.7: Example of a situation from Lemma 33. Segment $\bar{\sigma}$ is highlighted in blue and $D = \{v, v'\}$.

Lemma 34. Let $v \in V : d^e_G(v, \widehat{C}) \geq k + 1$. Let $u \in P$ be the nearest vertex to v on P. Then either $u = v$, or $d_G(u, v) = k$ and $d_G(u, \widehat{C}) = 1$.

Proof. If $v \in P$, then the nearest vertex on P is itself, so $u = v$. Suppose that $v \notin P$ (see Figure 3.8). There is no vertex from $\widehat{C}$ on any shortest path between v and u (otherwise $d^e_G(v, \widehat{C}) \leq d_G(v, u) \leq k$). In particular, $u \notin \widehat{C}$, therefore, u has exactly 2 neighbors on P. It also has at least one neighbor outside of P, through which it is connected to v. In $G \setminus \widehat{C}$, u must have at most 2 neighbors, thus at least one of its neighbors on P is in $\widehat{C}$. Because $L = P \cap \widehat{C}$, we get $d_G(u, L) = 1$. Then, from

$$k + 1 \leq d^e_G(v, \widehat{C}) \leq d_G(v, u) + d_G(u, L) = d_G(v, u) + 1 \leq k + 1,$$

we get $d_G(v, u) = k$. ☐

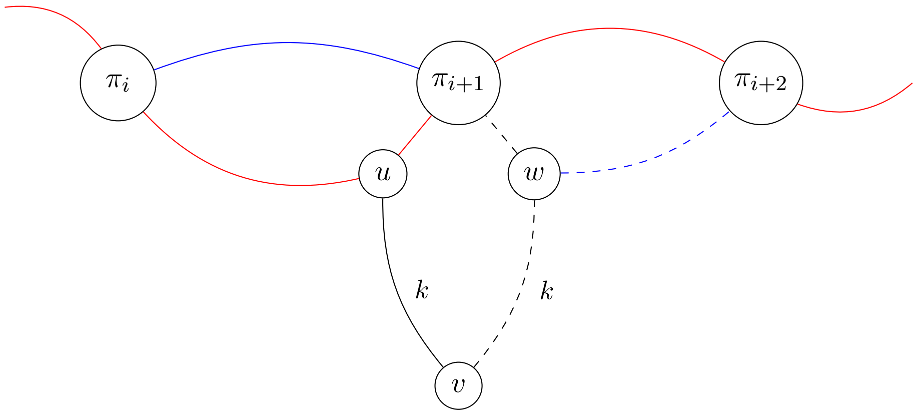

Figure 3.8: Example of a situation from Lemma 34. Path P is highlighted in red. Candidate segments are highlighted in blue. Clearly, the segment containing vertex u satisfies v. If all the dashed parts are present in G, then the segment containing vertex w also satisfies v.

Let $v \in V \setminus P : d^{e}_{G}(v, \widehat{C}) = k+1$ and $S \subseteq \widetilde{\Sigma}$ be a set of candidate segments that satisfy v. As shown in Figure 3.8, there may be multiple such candidate segments in S. However, we observe that if v itself lies on some candidate segment, then only those candidate segments in the same $\Sigma(\pi_i, \pi_{i+1})$ may satisfy v.

Observation 23. Let $\bar{\sigma} \in \Sigma(\pi_i, \pi_{i+1})$ be a candidate segment that contains some vertex u such that $d^{e}_{G}(u, \widehat{C}) = k+1$. Let $u' \in P$ be the nearest vertex to u on P. Then u' lies on a true segment $\sigma \in \Sigma(\pi_i, \pi_{i+1})$.

Proof. If $u' \in L$, then $d^{e}_{G}(u, \widehat{C}) \leq d_G(u, u') \leq k$. If $\sigma \notin \Sigma(\pi_i, \pi_{i+1})$, then P would not be a shortest path because $d_G(u, u') \leq k < k+1 \leq d_G(u, \{\pi_i, \pi_{i+1}\})$. □

We already know that if a candidate segment contains some vertex v with $d^{e}_{G}(v, \widehat{C}) > k+1$, then it must be a true segment. Now, we show another sufficient condition for a candidate segment to be a true segment.

Lemma 35. Let $\bar{\sigma} \in \Sigma(\pi_i, \pi_{i+1})$ be a candidate segment that contains some vertices $u, v \in \bar{\sigma}$ such that $u \neq v$ and $d^{e}_{G}(u, \widehat{C}) = d^{e}_{G}(v, \widehat{C}) = k+1$. Then, $\bar{\sigma}$ is a true segment.

Proof. Suppose that $\bar{\sigma}$ is not a true segment and let $\sigma \in \Sigma(\pi_i, \pi_{i+1})$ be a true segment. By Lemma 34 there must be some vertices $u', v' \in P$ with $d_G(u, u') = d_G(v, v') = k$. By Observation 23, both u' and v' are in σ. No shortest path between u and u', contains any vertex from $\widehat{C}$ (otherwise $d^{e}_{G}(u, \widehat{C}) \leq d_G(u, u') = k$). The same applies for any shortest path between v and v'. Thus, in $G \setminus \widehat{C}$, u is connected to v, v is connected to v', v' is connected to u', and u' is connected to u. Due to the length constraints, all these paths are disjoint (except for their endpoints). There is a cycle in $G \setminus \widehat{C}$, which is a contradiction with C being a modulator to disjoint paths. □

Let us summarize what we have shown so far.

If we had the correct values for the permutation $\boldsymbol{\pi}$ of vertices from $\widehat{C}$ that are on P, we would only need to select the true segment out of each $\Sigma(\pi_i, \pi_{i+1})$ to find a shortest path with eccentricity at most k.

There are some vertices $u \in V$ such that $d^{e}_{G}(u, \widehat{C}) \leq k$ and for these vertices, the distance to the resulting path will be at most k, no matter which segments we choose.

A segment which contains some vertex v with $d^{e}_{G}(v, \widehat{C}) > k+1$ must be a true segment. A segment which contains two vertices $u \neq v$ with $d^{e}_{G}(u, \widehat{C}) = d^{e}_{G}(v, \widehat{C}) = k+1$ must be a true segment as well.

For the remaining segments, we know that for every $u \in V$ with $d^{e}_{G}(u, \widehat{C}) > k$, the shortest path with eccentricity at most k needs to contain some $\sigma_u \in \widetilde{\Sigma}$ such that $d_G(u, \sigma_u) \leq k$. Furthermore, for every $v \in \widehat{C}$ with $d_G(v, L) > e(v)$, the path needs to contain some $\sigma_v \in \widetilde{\Sigma}$ such that $d_G(v, \sigma_v) \leq e(v)$.

Clearly, the problem of selecting one true segment out of each set of candidate segments is an instance of the CSC problem: the sets of candidates are $\mathcal{C} = \Sigma(\pi_1, \pi_2) \cup \ldots \cup \Sigma(\pi_{|L|-1}, \pi_{|L|})$, the requirements are $\mathcal{R} = \{v \in V \setminus \widehat{C} \mid d^e_G(v, \widehat{C}) > k\} \cup \{v \in \widehat{C} \setminus L \mid d_G(v, L) > e(v)\}$, and the function $\Psi(\bar{\sigma}) = \{v \in V \setminus \widehat{C} \mid \bar{\sigma} \text{ satisfies } v\} \cup \{v \in \widehat{C} \setminus L \mid d_G(v, \sigma) \leq e(v)\}$.

We know that the number of vertices outside of P that the true segments can satisfy is bounded by the size of L. Furthermore, we know that if a segment contains at least two vertices that need to be satisfied, then it must be a true segment. Lastly, we know that if a segment from some $\Sigma(\pi_i, \pi_{i+1})$ contains one vertex v with $d^e_G(v, \widehat{C}) = k+1$, then only segments from the same $\Sigma(\pi_i, \pi_{i+1})$ may satisfy v. Thus, all segments in $\Sigma(\pi_i, \pi_{i+1})$ that do not satisfy v may be disregarded. By this, we ensure that v will be satisfied no matter which segment is chosen, and v does not need to be added to the requirements $\mathcal{R}$. Hence, the requirements $\mathcal{R}$ do not need to contain any vertices from P, and the size of $\mathcal{R}$ is bounded by the size of C.

In the following lemma, we show that we do not need to explicitly check whether a segment contains some vertex v with $d^e_G(v, \widehat{C}) > k+1$ to decide that it is a true segment. This will simplify our pseudocode a bit.

Lemma 36. If a segment $\sigma \in \widetilde{\Sigma}$ contains some vertex u such that $d^e_G(u, \widehat{C}) > k+1$, then it must also contain two vertices v, v' such that $d^e_G(v, \widehat{C}) = d^e_G(v', \widehat{C}) = k+1$.

Proof. Let s, t be endpoints of σ, thus $d^e_G(s, \widehat{C}) = d^e_G(t, \widehat{C}) = 1$. If there was no $v \in \sigma$ with $d^e_G(v, \widehat{C}) = k+1$ between s and u, then there would have to be some neighbors $p, q \in \sigma$ such that $d^e_G(p, \widehat{C}) > d^e_G(q, \widehat{C}) + 1$. This is a contradiction because clearly $d^e_G(p, \widehat{C}) \leq d^e_G(q, \widehat{C}) + 1$ if p is a neighbor of q. The same holds for $v' \in \sigma$ with $d^e_G(v', \widehat{C}) = k+1$ between u and t. □

Finally, we propose an algorithm that solves the MESP problem. First, we describe the main ideas informally, and then we present the pseudocode as Algorithm 7.

The algorithm finds the correct values for $L, \boldsymbol{\pi}$, and $e(\widehat{C} \setminus L)$ by trying all possible combinations. For each combination, it performs the following steps.

1. For each π_i, π_{i+1}, check all candidate segments in $\Sigma(\pi_i, \pi_{i+1})$.
   a) If there are any segments containing a vertex u such that $d^e_G(u, \widehat{C}) = k+1$, then we may disregard all candidate segments which do not satisfy u.
   b) After disregarding these segments, if there is only one candidate segment left, it is a true segment. If there is no candidate segment left, then no solution exists.
2. If there is a vertex v such that $d^e_G(v, \widehat{C}) > k+1$, and it does not lie on a segment that we have marked as true segment, then no solution exists.

3. Construct the set U of vertices v that are not contained in any segment and have $d^e_G(v, \widehat{C}) = k + 1$. If $|U| > 2(|L| - 1)$, then no solution exists.

4. Choose the rest of the true segments from all candidate segments (except those disregarded in step 1) by solving the CSC problem, with requirements $u \in \widehat{C} \setminus L$ whose distance to the parts of P selected so far is greater than $e(u)$, and all of U.

5. If the CSC has a solution, construct a path from $\boldsymbol{\pi}$ and from the chosen candidate segments. If the resulting path is a shortest path with eccentricity at most k, return it.

Lemma 37. Algorithm 7 finds a shortest path with eccentricity at most k if it exists, and reports if it does not exist.

Proof. If a shortest path with eccentricity at most k exists, the values $(\bar{L}, \bar{\boldsymbol{\pi}}, \bar{e}) = (L, \boldsymbol{\pi}, e)$ will be found in some iteration of the foreach-loop on line 3.

By Observation 23, if there is a candidate segment containing some vertex v with $d^e_G(v, \widehat{C}) = k + 1$, then the true segment from $\Sigma(\pi_i, \pi_{i+1})$ must satisfy v. Thus, we may disregard all segments that do not satisfy v and line 9 is correct. If C_i does not contain any segment, then either there are no candidate segments in $\Sigma(\bar{\pi}_i, \bar{\pi}_{i+1})$ or there are some but none of them satisfies all vertices in K. In either case, this implies that there is no solution for the current configuration of $(\bar{L}, \bar{\pi}, \bar{e})$ and lines 10–11 are correct. If C_i contains only one segment, it must be the true segment, thus lines 12–14 are correct. If there are multiple segments to choose from, we will need to choose later by solving the CSC. However, no matter which candidate segment from C_i we choose, it will satisfy all vertices in K. Thus, lines 15–19 are correct.

From Lemma 34, if $d^e_G(u, \widehat{C}) > k$, then either $u \in P$ or $d^e_G(u, \widehat{C}) = k + 1$. However, if $u \in P$ and $d^e_G(u, \widehat{C}) > k + 1$, then u lies on a candidate segment which would have been chosen as some σ_i on line 13, because by Lemma 36 it contains two vertices v, v' with $d^e_G(v, \widehat{C}) = d^e_G(v', \widehat{C}) = k+1$ and by Lemma 35, u and u' are not satisfied by any other segment. Thus, lines 22–23 are correct.

From Corollary 5 we know that lines 27–28 are correct.

We have shown that the segments $\bar{\sigma}_i \neq \perp$ are the true segments. Now, for the resulting path to have eccentricity at most k, we need to satisfy $d_G(u, P) \leq e(u)$ for every $u \in \widehat{C} \setminus L$, and $d_G(u, P) \leq k$ for every $u \in U$. These are vertices that do not lie on P, but the vertices on P nearest to them lie on segments which have not been chosen yet. The constrained set cover solution, as described on lines 29–30, chooses the rest of the segments exactly to satisfy all these requirements.

Clearly, lines 31–32 are correct. We have shown that if a shortest path with eccentricity at most k exists, the algorithm finds and returns it.

If the algorithm returns a path, it is clearly a shortest path with eccentricity at most k, thanks to the check on line 31. □

Algorithm 7: FPT Distance to Disjoint Paths

```
Input: graph $G = (V, E)$, modulator to disjoint paths $C \subseteq V$
foreach $\bar{a}, \bar{b} \in (V \setminus C), \bar{a} \neq \bar{b}$ do
    $\widehat{C} = C \cup \{\bar{a}, \bar{b}\}$
    foreach $(\bar{L}, \bar{\boldsymbol{\pi}}, \bar{e}) : \bar{L} \subseteq \widehat{C}$, $\bar{\boldsymbol{\pi}} = (\bar{a} = \bar{\pi}_1, \ldots, \bar{\pi}_{|\bar{L}|} = \bar{b})$ is a
     permutation of $\bar{L}$, $\bar{e} : (\widehat{C} \setminus \bar{L}) \to \{1, ..., k\}$ do
        $S \leftarrow \emptyset$
        $I \leftarrow \bar{L}$
        $\ell = 0$
        for $i = 1, \ldots, |\bar{L}| - 1$ do
            $K \leftarrow \{u \in V \mid \exists \bar{\sigma} \in \Sigma(\bar{\pi}_i, \bar{\pi}_{i+1}) : u \in \bar{\sigma} \wedge d_G^{\bar{e}}(u, \widehat{C}) = k + 1\}$
            $C_i \leftarrow \{\bar{\sigma} \in \Sigma(\bar{\pi}_i, \bar{\pi}_{i+1}) \mid \forall u \in K : \bar{\sigma}$ satisfies $u\}$
            if $C_i = \emptyset$ then
                continue to the next $(\bar{L}, \bar{\pi}, \bar{e})$
            else if $|C_i| = 1$ then
                $\bar{\sigma}_i \leftarrow$ the only segment in $C_i$
                $I \leftarrow I \cup \bar{\sigma}_i$
            else
                $\bar{\sigma}_i \leftarrow \perp$
                $\ell \leftarrow \ell + 1$
                $h_\ell \leftarrow i$
                $S \leftarrow S \cup K$
        $\bar{U} \leftarrow \emptyset$
        foreach $v \in V \setminus (\widehat{C} \cup S \cup I)$ do
            if $d_G^{\bar{e}}(v, \widehat{C}) > k + 1$ then
                continue to the next $(\bar{L}, \bar{\boldsymbol{\pi}}, \bar{e})$
            else if $d_G^{\bar{e}}(v, \widehat{C}) = k + 1$ then
                $\bar{U} \leftarrow \bar{U} \cup \{v\}$
                $\bar{e}(v) \leftarrow k$
        if $|\bar{U}| > 2(|\bar{L}| - 1)$ then
            continue to the next $(\bar{L}, \bar{\boldsymbol{\pi}}, \bar{e})$
        let $\bar{\sigma}_{h_1}, \ldots, \bar{\sigma}_{h_\ell}$ be the solution of CSC for:
          requirements $\mathcal{R} = \{u \in (\widehat{C} \setminus \bar{L}) \cup \bar{U} \mid d_G(u, I) > e(u)\}$,
          candidates $\mathcal{C} = C_{h_1} \cup \ldots \cup C_{h_\ell}$,
          and $\Psi(\bar{\sigma}) = \{u \in \mathcal{R} \mid d_G(u, \bar{\sigma}) \leq \bar{e}(u)\}$
        $P := (\bar{\pi}_1) \frown \bar{\sigma}_1 \frown (\pi_2) \frown \ldots \frown (\bar{\pi}_{|\bar{L}|-1}) \frown \bar{\sigma}_{|\bar{L}|-1} \frown (\bar{\pi}_{|\bar{L}|})$
        if $d_G(\bar{\pi}_1, \bar{\pi}_{|\bar{L}|}) = \sum_{i=1}^{|\bar{L}|-1} d_G(\bar{\pi}_i, \bar{\pi}_{i+1}) \wedge ecc_G(P) \leq k$ then
            return $P$
A shortest path with eccentricity at most $k$ does not exist.
```

Lemma 38. Algorithm 7 runs in $\mathcal{O}(2^{5c}c!k^c c^2 \cdot n^4)$ time for $c = |C|$.

Proof. The distances between all pairs of vertices can be precomputed in $\mathcal{O}(n^3)$ time. The foreach-loop on line 1 will have at most $\mathcal{O}(n^2)$ iterations. The foreach-loop on line 3 will have at most $\mathcal{O}(2^c c!k^c)$ iterations. The for-loop on line 7 will have $\mathcal{O}(c)$ iterations. The set K on line 8 can be constructed in $\mathcal{O}(n + m)$ time. Because of Lemma 35, at most two vertices need to be stored in K for each candidate segment. Thus, by Lemma 33, each segment will satisfy at most four vertices from K (two lying on the segment, and two outside the segment). If $|K| > 4$, C_i may be set directly to $\emptyset$. Otherwise, C_i can be constructed in $\mathcal{O}(n + m)$ by iterating over all vertices in all the candidate segments and checking at most 4 conditions for each vertex. Line 14 can be implemented in $\mathcal{O}(n)$ time. Lines 10–13 and 15–19 can be implemented in constant time. The foreach-loop on line 21 will have at most $\mathcal{O}(n)$ iterations. Lines 22–26 can be implemented in $\mathcal{O}(c)$ time. Lines 27–28 can be implemented in constant time. The CSC on line 29 can be solved by Algorithm 3 in $\mathcal{O}(2^{4c}c^2)$ time as $|\mathcal{C}| = \mathcal{O}(c)$ and

$$\begin{aligned}
|\mathcal{R}| &\leq |(\widehat{C} \setminus \bar{L}) \cup \bar{U}| \\
&= |\widehat{C}| - |\bar{L}| + |\bar{U}| \\
&\leq |\widehat{C}| - |\bar{L}| + 2(|\bar{L}| - 1) \\
&= |\widehat{C}| + |\bar{L}| - 2 \\
&\leq (c + 2) + (c + 2) - 2 \\
&= 2c + 2.
\end{aligned}$$

The check on line 31 can be implemented in $\mathcal{O}(n + m)$ time. □

Chapter 4

Implementation and Measurements

We have implemented the FPT algorithm parameterized by the distance to disjoint paths and minimum eccentricity (Algorithm 7). We ran the program on some larger graphs with a small distance to disjoint paths and experimentally concluded that thanks to our algorithm, the MESP problem is now tractable even on graphs with hundreds of vertices.

4.1 Implementation

The implementation consists of two main programs: `mesp`—an implementation of Algorithm 7 which takes a graph and a modulator to disjoint paths as input and solves the MESP problem, and `paths`—an implementation of the branching algorithm from Lemma 32 which finds the modulator to disjoint paths. A third program `test` served mainly for testing during active development: it combines the first two programs, then runs on a number of small graphs, and checks the resulting paths against provided expected eccentricities.

All three programs are implemented in the `C++ 17` programming language using the `Boost 1.72` library, and are operated through a command-line interface. All three programs measure their computation time and report it back to the user along with other information. The current system time is read once before the computation starts (the `boost::chrono::system_clock::now()` function is used), and once after it ends. The computation time is obtained by subtracting the initial time from the end time, and it is printed to the standard output. Note that the initial system time is read *after* loading the input, so the reported computation time is a bit less than the running time.

The `mesp` program also supports parallel execution for each combination of start- and endpoints of the shortest path.

4.2 Environment

The experiments were run on a virtual machine instance of the Google Compute Engine. The VM configuration was: 8 virtual CPUs (Intel® Xeon® Skylake @ 2.00 GHz), 30 GB RAM, operating system Debian 10 “buster”. The program was compiled with the `gcc` compiler version 8.3.0 and with `-Ofast` optimizations enabled. We allocated eight threads for the program so that all the virtual CPUs were used up.

4.3 Datasets

Our goal was to evaluate the computation time of the program based on the values of distance to disjoint paths and the resulting eccentricity. To do this, we have created the Evaluation Dataset. Each graph in the dataset was generated using the `Python 3` programming language by performing the following steps.

1. Generate p paths of length ℓ.
2. Generate c vertices (these will form the modulator to disjoint paths).
3. Generate e extra edges at random, such that at least one endpoint of each edge is in the modulator to disjoint paths.

To generate the dataset, we used values $p = \ell = 23$ and $c \in \{1, \dots, 6\}$. The values p and ℓ are constant throughout the dataset to minimize the influence of parameters other than the distance to disjoint paths and resulting eccentricity on the running time. By changing the value of e, we aimed to control the resulting eccentricity.

To verify that our choice for the values p and ℓ in the Evaluation Dataset does not yield graphs with properties significantly different from what other values of these parameters would yield, we have created the Comparison Dataset. In this dataset, we have generated graphs the same way as in the Evaluation Dataset, but with different values of p and ℓ. There are seven graphs with $p = 23$ and $\ell \in \{20, \dots, 26\}$, seven graphs with $p \in \{20, \dots, 26\}$ and $\ell = 23$, and three graphs with $p = 23$ and a random path length (different for each generated path).

We have also created the Miscellaneous Dataset, which contains several larger graphs with random path lengths (generated similarly as the last three graphs in the Comparison Dataset). Additionally, we have added one graph on 101 vertices that was created manually to verify the correct handling of some edge cases that sprang to our minds.

Furthermore, we have decided to show that the algorithm can even be used for some real-world networks, and we have created a graph from the 2014 route map of the Taipei Mass Rapid Transit (MRT) [10]. Each vertex in the graph represents one MRT station, and each edge represents a direct connection between two stations through some rail line. The graph does not contain some small branch lines, such as the Xiaobitan branch line or the Xinbeitou branch line. The graph is included in the Miscellaneous Dataset.

4.4 Results

The measurements performed on the Miscellaneous Dataset show that Algorithm 7 is able to solve the MESP problem even for graphs with over 1,000 vertices. In comparison, a naive $\mathcal{O}(2^n)$ approach that tries all subsets of vertices would not be able to solve the problem even for graphs with 100 vertices. The results are displayed in Table 4.1.

The Evaluation Dataset shows a direct dependence of the computation time on the value of distance to disjoint paths c, as well as on the resulting eccentricity k. The number of vertices remains nearly constant throughout the dataset (between 553 and 558, depending on the value of c) while the number of edges varies significantly. However, we can see that the changes in the computation time are much smaller if the values of k and c remain fixed, as opposed to when either of the values changes. Thus, the influence of c and k on the computation time is more significant than the influence of the number of edges, as was expected from the time complexity of the algorithm. The results are displayed in Table 4.2.

The Comparison Dataset shows that slight changes in the values of p and ℓ do not affect the computation time significantly. In comparison, slight changes of the distance to disjoint paths or the resulting eccentricity affect the computation time much more. The results are displayed in Table 4.3.

The computation times in Table 4.1, Table 4.2, and Table 4.3 are as measured by the `mesp` program (see Section 4.1), and rounded to the nearest whole second. The running time of the `paths` program is not included as the problem of finding the distance to disjoint paths was not a goal of this thesis. Note, however, that even the simple algorithm we used is much faster than the MESP algorithm itself, and the `paths` program finished in less than one second for all graphs in our datasets.

Table 4.1: Miscellaneous Dataset. Each row represents one graph with the following properties, respectively: type of the graph, the total number of vertices, total number of edges, distance to disjoint paths, eccentricity of the resulting path, computation time.

Graph	n	m	c	k	Time
Taipei MRT	105	113	8	11	178 h 5 m 58 s
generated	1,895	3,330	5	6	50 h 33 m 32 s
generated	565	701	5	11	46 h 11 m 53 s
generated	4,272	7,574	3	6	33 h 19 m 28 s
generated	1,228	2,172	6	5	15 h 18 m 36 s
generated	1,210	2,653	7	4	15 h 3 m 5 s
generated	2,265	4,316	4	4	3 h 24 m 2 s
generated	1,063	3,006	7	3	1 h 13 m 28 s
hand-made	101	126	10	3	8 m 37 s

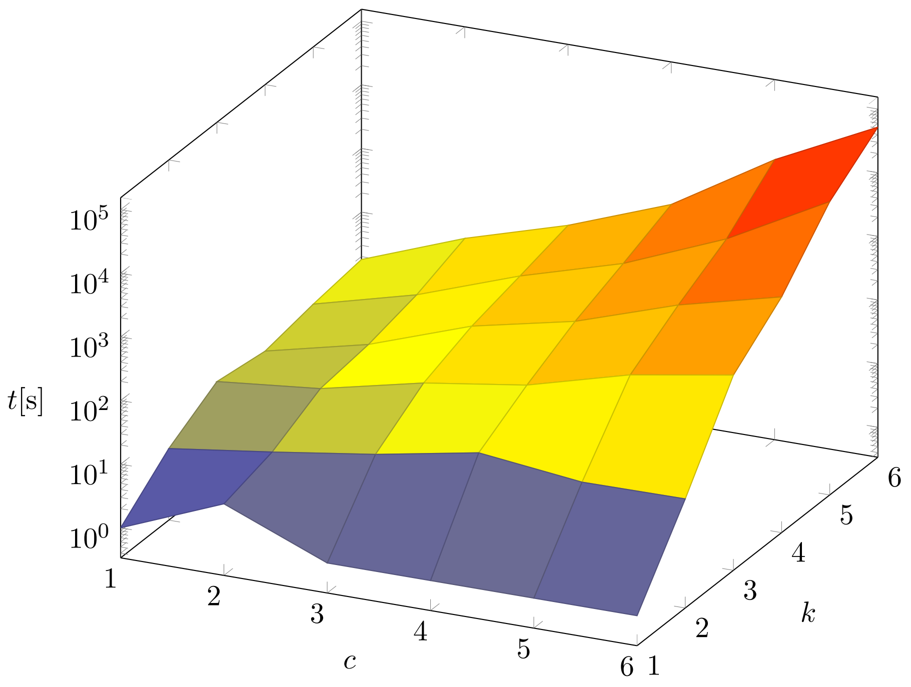

Figure 4.1: The Evaluation Dataset plot shows how the computation time t[s] of the program depends on the distance to disjoint paths c and the resulting eccentricity k. Note the logarithmic scale of the time axis. The plot was created from the data in Table 4.2. For some combinations of c and k, there are multiple values in the table. In such cases, the median computation time is used in the plot.

Table 4.2: Evaluation Dataset. The columns contain the following values, respectively: the total number of vertices, total number of edges, number of extra edges (as described in Section 4.3), distance to disjoint paths, eccentricity of the resulting path, computation time in seconds.

n	m	e	c	k	t[s]
558	929	400	6	6	52,003
558	954	425	6	5	13,751
558	979	450	6	4	2,632
558	1,029	500	6	4	1,522
558	1,079	550	6	4	1,776
558	1,129	600	6	4	1,640
558	1,229	700	6	3	515
558	1,329	800	6	3	188
558	1,429	900	6	3	294
558	1,529	1,000	6	3	395
558	1,629	1,100	6	3	399
558	1,729	1,200	6	2	12
558	1,829	1,300	6	2	11
558	2,429	1,900	6	2	23
558	3,529	3,000	6	2	44
558	3,629	3,100	6	1	1
557	1,029	500	5	6	8,740
557	1,044	515	5	5	1,836
557	1,054	525	5	4	548
557	1,079	550	5	5	1,932
557	1,129	600	5	4	469
557	1,229	700	5	4	801
557	1,329	800	5	4	924
557	1,429	900	5	3	196
557	1,529	1,000	5	3	179
557	1,629	1,100	5	3	225
557	1,729	1,200	5	2	10
557	1,829	1,300	5	3	249
557	1,929	1,400	5	2	12
557	2,529	2,000	5	2	22
557	3,029	2,500	5	2	32
557	3,129	2,600	5	1	1
556	779	250	4	6	889
556	829	300	4	5	413
556	929	400	4	5	407
556	1,029	500	4	4	209
556	1,129	600	4	4	186

n	m	e	c	k	t[s]
556	1,179	650	4	3	64
556	1,229	700	4	2	68
556	1,329	800	4	2	67
556	1,429	900	4	3	90
556	1,529	1,000	4	2	8
556	2,529	2,000	4	2	23
556	2,629	2,100	4	2	26
556	2,729	2,200	4	1	1
555	829	300	3	6	221
555	929	400	3	5	138
555	1,029	500	3	4	80
555	1,129	600	3	4	89
555	1,229	700	3	4	88
555	1,329	800	3	3	44
555	1,429	900	3	2	13
555	1,529	1,000	3	2	9
555	2,029	1,500	3	2	15
555	2,129	1,600	3	1	1
554	729	200	2	7	72
554	779	250	2	6	73
554	829	300	2	5	37
554	929	400	2	4	24
554	1,029	500	2	3	16
554	1,129	600	2	3	19
554	1,229	700	2	3	37
554	1,329	800	2	2	6
554	1,429	900	2	1	8
554	1,529	1,000	2	2	9
554	1,629	1,100	2	1	1
553	729	200	1	7	20
553	779	250	1	6	18
553	829	300	1	5	14
553	929	400	1	4	10
553	954	425	1	3	13
553	979	450	1	2	5
553	1,029	500	1	2	4
553	1,079	550	1	1	1

Table 4.3: Comparison Dataset. The columns contain the following values, respectively: the total number of vertices, total number of edges, number of paths, length of each path, number of extra edges (as described in Section 4.3), distance to disjoint paths, eccentricity of the resulting path, computation time in seconds. The dataset is divided into three sections: the first section contains graphs with a variable number of paths and fixed path length; the second section contains graphs with a fixed number of paths and variable path length; the third section contains graphs with a fixed number of paths and each path having a uniformly random length between 1 and 45.

n	m	p	ℓ	e	c	k	t[s]
486	960	20	23	500	6	4	986
510	983	21	23	500	6	4	1,751
534	1,006	22	23	500	6	4	2,278
558	1,029	23	23	500	6	4	2,030
582	1,052	24	23	500	6	4	4,235
606	1,075	25	23	500	6	4	2,517
630	1,098	26	23	500	6	5	10,305
489	960	23	20	500	6	4	1,967
512	983	23	21	500	6	4	1,307
535	1,006	23	22	500	6	4	1,707
558	1,029	23	23	500	6	4	1,258
581	1,052	23	24	500	6	5	8,191
604	1,075	23	25	500	6	5	9,530
627	1,098	23	26	500	6	4	5,233
552	1,023	23	1–45	500	6	4	1,221
493	964	23	1–45	500	6	4	1,364
591	1,062	23	1–45	500	6	5	7,639

Chapter 5

Conclusion

We have shown that the MESP problem is fixed-parameter tractable with respect to the maximum leaf number, neighborhood diversity, distance to cluster (and thus vertex cover and twin cover), and the combination of distance to disjoint paths with the minimum eccentricity. We have implemented one of the algorithms and shown that it can be used to find the minimum eccentricity shortest path in reasonably large graphs.

An open question remains, whether the presented algorithms are asymptotically optimal (assuming the Exponential Time Hypothesis [11]). The natural next steps in the research of parameterized complexity of the MESP problem would be to investigate the existence of FPT algorithms with respect to the distance to disjoint paths alone, and with respect to other structural parameters, such as tree depth.

Bibliography

1. DRAGAN, Feodor F.; LEITERT, Arne. On the Minimum Eccentricity Shortest Path Problem. In: DEHNE, Frank; SACK, Jörg-Rüdiger; STEGE, Ulrike (eds.). *Algorithms and Data Structures*. Cham: Springer International Publishing, 2015, pp. 276–288. ISBN 978-3-319-21840-3. Available from DOI: `10.1007/978-3-319-21840-3_23`.
2. BIRMELÉ, Étienne; MONTGOLFIER, Fabien de; PLANCHE, Léo. Minimum Eccentricity Shortest Path Problem: An Approximation Algorithm and Relation with the k-Laminarity Problem. In: CHAN, T-H. Hubert; LI, Minming; WANG, Lusheng (eds.). *Combinatorial Optimization and Applications*. Cham: Springer International Publishing, 2016, pp. 216–229. ISBN 978-3-319-48749-6. Available from DOI: `10.1007/978-3-319-48749-6_16`.
3. DRAGAN, Feodor F.; LEITERT, Arne. Minimum Eccentricity Shortest Paths in some Structured Graph Classes. *Journal of Graph Algorithms and Applications*. 2016, vol. 20, no. 2, pp. 299–322. Available from DOI: `10.7155/jgaa.00394`.
4. DRAGAN, Feodor F.; LEITERT, Arne. On the minimum eccentricity shortest path problem. *Theoretical Computer Science*. 2017, vol. 694, pp. 66–78. ISSN 0304-3975. Available from DOI: `10.1016/j.tcs.2017.07.004`.
5. BIRMELÉ, Etienne E.; MONTGOLFIER, Fabien de; PLANCHE, Léo; VIENNOT, Laurent. *Decomposing a Graph into Shortest Paths with Bounded Eccentricity*. 2017. Available also from: `https://hal.archives-ouvertes.fr/hal-01511357`. working paper or preprint.
6. BOULAND, Adam M. *Parameterized Complexity and Graph Isomorphism*. 2011. Available also from: `https://www.cl.cam.ac.uk/~ad260/bouland.pdf`. Master's thesis. University of Cambridge.

7. CHEN, Jianer; KANJ, Iyad A.; XIA, Ge. Improved Parameterized Upper Bounds for Vertex Cover. In: KRÁLOVIČ, Rastislav; URZYCZYN, Paweł (eds.). *Mathematical Foundations of Computer Science 2006*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 238–249. ISBN 978-3-540-37793-1. Available from DOI: `10.1007/11821069_21`.

8. GANIAN, Robert. Twin-Cover: Beyond Vertex Cover in Parameterized Algorithmics. In: MARX, Dániel; ROSSMANITH, Peter (eds.). *Parameterized and Exact Computation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 259–271. ISBN 978-3-642-28050-4. Available from DOI: `10.1007/978-3-642-28050-4_21`.

9. BORAL, Anudhyan; CYGAN, Marek; KOCIUMAKA, Tomasz; PILIPCZUK, Marcin. A Fast Branching Algorithm for Cluster Vertex Deletion. *Theory of Computing Systems*. 2016, vol. 58, no. 2, pp. 357–376. ISSN 1433-0490. Available from DOI: `10.1007/s00224-015-9631-7`.

10. TAIPEI RAPID TRANSIT CORPORATION. *2014 Annual Report*. Taipei Rapid Transit Corporation, 2015. Available also from: `https://www-ws.gov.taipei/001/Upload/public/Attachment/583115232012.pdf`.

11. IMPAGLIAZZO, Russell; PATURI, Ramamohan. On the Complexity of k-SAT. *Journal of Computer and System Sciences*. 2001, vol. 62, no. 2, pp. 367–375. ISSN 0022-0000. Available from DOI: `https://doi.org/10.1006/jcss.2000.1727`.

Appendix A

Acronyms

BFS	breadth-first search
CSC	constrained set cover
FPT	fixed-parameter tractable
MESP	minimum eccentricity shortest path
MRT	mass rapid transit
VM	virtual machine
XP	slice-wise polynomial

Appendix B

Contents of enclosed CD

The CD is also available online at `http://mkucera.cz/bp.zip`.

```
bin ........................................ executables
 ├─ mesp ...................................... the MESP executable
 └─ paths ................. the modulator to disjoint paths executable
datasets ................. the datasets presented in this thesis
 ├─ comparison ........................... the Comparison Dataset
 ├─ evaluation ........................... the Evaluation Dataset
 └─ miscellaneous ..................... the Miscellaneous Dataset
res .................... outputs for all graphs from the datasets
 ├─ comparison ................ outputs for the Comparison Dataset
 ├─ evaluation ................ outputs for the Evaluation Dataset
 └─ miscellaneous ........... outputs for the Miscellaneous Dataset
src ......................................... source codes
 ├─ impl ................................ implementation sources
 ├─ text ....................... XeLaTeX source codes of this thesis
 └─ gen.py ............. the script for generating random graphs
readme.txt ........................... CD contents description
thesis.pdf ...................... the thesis text in PDF format
```