On the Complexity of the Regenerator Location Problem - Treewidth and Other Parameters

Itamar Hartstein
On the Complexity of the Regenerator Location Problem - Treewidth and Other Parameters

RESEARCH THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE

Itamar Hartstein

SUBMITTED TO THE SENATE OF THE TECHNION - ISRAEL INSTITUTE OF TECHNOLOGY
Adar, 5772 Haifa March 2012
THE RESEARCH THESIS WAS DONE UNDER THE SUPERVISION OF PROF. SHMUEL ZAKS AND DR. MORDECHAI SHALOM IN THE DEPARTMENT OF COMPUTER SCIENCE

THE GENEROUS FINANCIAL HELP OF THE TECHNION IS GRATEFULLY ACKNOWLEDGED
# Contents

1 Introduction ......................................................... 2  
   1.1 Background .................................................. 2  
      1.1.1 Optical Networks ...................................... 2  
      1.1.2 Regenerators .......................................... 2  
   1.2 Related Work ................................................ 4  
   1.3 Our Results ................................................ 5  

2 Preliminaries ..................................................... 7  

3 $RLP_{req}$ in rings ................................................ 12  

4 Hardness of $RLP_{req}$ in Bounded Treewidth Graphs .............. 16  
   5.1 $RLP_{path}$ is FPT in $tw(G)$ for $d = 2$ .................... 21  
   5.2 $RLP_{path}$ is NP-HARD for $tw(G) \leq 2$ and $d = 5$ ........ 22  
   5.3 $RLP_{path}$ is NP-HARD even when $L_v(P) \leq 4$ and $d = 3$ 26  
   5.4 $RLP_{path}$ is in P for bounded treewidth and vertex load .... 27  
      5.4.1 Definitions and Basic Observations .................... 27  
      5.4.2 A dynamic programming algorithm for $RLP_{path}$ .......... 30  
      5.4.3 Analysis of the algorithm ................................ 32  

5 $RLP_{path}$ with Treewidth and Vertex Load Parameters ........... 21  

6 Fixed Number of Connections ..................................... 36  
   6.1 $RLP_{path}$ with fixed number of paths ..................... 36  
   6.2 $RLP_{req}$ with fixed number of requests ................... 40  

7 Summary and Open Problems ..................................... 45  

Bibliography ......................................................... 47
List of Figures

1.1 A simplified optical network ............................................. 3
2.1 An example of a 2-satisfying regenerator location placement ........... 8
2.2 An instance of $RLP_{path}$ ................................................... 9
2.3 An instance of $RLP_{req}$ ................................................... 9
2.4 An example of a tree decomposition of a graph .......................... 10
3.1 Demonstration of the different cases for a request $(s, t)$ ................... 14
4.1 An example of the constructed graph $G$ and request set $Q$ for $\phi = R(x_1, 0, x_2) \land R(x_2, x_3, 1) \land R(x_4, x_1, x_2)$ ......................... 18
4.2 A tree decomposition for the graph $G$ .................................. 20
5.1 An example of the construction of $G$ and $P$ for $\phi = x_1 \lor x_2 \lor x_3$ ......... 23
5.2 The component $T_j$ in the tree decomposition $T$ of the graph $G$ ....... 25
6.1 (1) An example graph $G$. (2) The graph $G'$ ................................ 37
6.2 (a) An undirected graph $G$. (b) The constructed directed graph $G'$ ........ 41
Abstract

In this thesis we deal with the Regenerator Location Problem in optical networks. One of the main problems in optical network design is the loss of signal energy after light travels large distances. Optical amplifiers are used to amplify the signal, but unfortunately they also introduce noise. Therefore, at some point, the signal must be regenerated so it will remain above a certain SNR (Signal-to-Noise Ratio) threshold. Regenerators are placed after every fixed distance \(d\) in order to enable communication over large distances.

We model an optical network as an undirected graph \(G = (V, E)\), and a set \(Q\) of communication requests between pairs of terminals in \(V\). We investigate two variations of the regenerator location problem: one in which we are given a routing \(P\) of the requests in \(Q\), and one in which we are required to find also the routing. In both cases, each path in \(P\) must contain at least one regenerator in every \(d\) consecutive internal vertices. Our goal for both problems is to minimize the number of regenerator locations used by the solution, that is, the number of vertices containing at least one regenerator.

Both variations of the problem are \(NP\)-Hard in the general case, and existing works have attempted to overcome this problem in various ways, including heuristics, simulations, approximation algorithms and identification of polynomial-time cases. In this work, we investigate the parameterized complexity of the problem. That is, we investigate the inherent difficulty of the problems at hand with respect to multiple parameters of the input. Our results include fixed parameter tractability results, polynomial algorithms for fixed parameter values, and several \(NP\)-Hardness results.

Several parameters are considered in this work. The main parameter discussed is the treewidth of the input graph. Informally, the treewidth is a measure of the similarity of a graph to a tree. Another parameter discussed here is the vertex load of the path set when the routing is given - that is, the maximum number of paths in which any vertex serves as an intermediate node. Other parameters discussed are the distance between consecutive regenerators, \(d\), and the number of connection requests.
**List of Terms**

\[ G = (V, E) \]  
Graph representing the network

\[ n \]  
The number of vertices

\[ m \]  
The number of edges

\[ \mathcal{P} \]  
The set of paths

\[ Q \]  
The set of connection requests

\[ d \]  
The maximum distance between two consecutive regenerators on a path

\[ \pi \]  
A path in \( G \)

\[ R \]  
A set of regenerator locations over a vertex set \( V \)

\[ RLP_{path}(G, d, \mathcal{P}) \]  
The problem of finding a regenerator location assignment for a set of paths \( \mathcal{P} \) over \( G \)

\[ RLP_{req}(G, d, Q) \]  
The problem of finding a routing and regenerator location assignment for a set of requests \( Q \) over \( G \)

\[ \ell_{i,k} \]  
The \( k \)'th literal of the \( i \)'th clause in a SAT formula.

\[ f \]  
An assignment for boolean variables of a SAT formula.

\[ C \]  
A vertex cover of \( G \)

\[ \mathcal{T} = (B, F) \]  
A tree decomposition of a graph \( G \)

\[ tw(G) \]  
The treewidth of \( G \)

\[ L_v(\mathcal{P}) \]  
The vertex load of \( \mathcal{P} \)
Chapter 1

Introduction

1.1 Background

1.1.1 Optical Networks

The demand for high speed and capacity networks grows in an enormous rate. Applications such as high definition streaming and video conferencing, real-time medical imaging, high speed super-computing, distributed computing and database synchronization require fast and reliable networks with quick response times over large distances. Some of these applications already demand transfer rates of several gigabytes per second, and more are emerging every year. The need for larger and faster networks arises in all levels of networking: LANs (local area networks), MANs (metropolitan area networks) and WANs (wide area networks).

All-optical networks is the most promising technology on route to implementation of gigabit networks, and has thus been the subject of many research works in the past decade. The idea behind all-optical networks is to keep the signal in optical form, and avoid optical-to-electrical signal conversion in intermediate nodes.

Large modern networks are typically required to support thousands of active connections simultaneously. To deal with this requirement, optical networks employ the WDM (Wavelength Division Multiplexing) technology. In WDM, each source-destination pair is assigned a wavelength (“color”), and the route on which they communicate in the network is called a lightpath. Several lightpaths may share a common optical fiber given that each lightpath is assign a different wavelength. Optical fibers carrying around 80 wavelengths simultaneously are operational, and networks with several hundred wavelengths are already experimental. In case certain communication requests require only a fraction of the bandwidth of a single wavelength, several communication requests can be joined together to fit into a single wavelength, by using a technique called traffic grooming.

1.1.2 Regenerators

One of the main problems in optical network design is the loss of signal energy after light travels large distances. To overcome this problem, amplifiers are placed after every fixed
distance (a typical value being around 100km). However, amplifiers introduce noise to the signal, and at some point the SNR (Signal-to-Noise Ratio) is too low to communicate, and the signal needs to be regenerated. This is done in the following manner: An ROADM (Reconfigurable Optical Add-Drop Multiplexer) has the capability to insert/extract a certain number of wavelengths (typically, around 4) to/from an optical fiber (without the need to convert the signals on all the wavelengths to electronic signals and back again to optical signals). Then, for each extracted wavelength, an optical regenerator is needed to regenerate the signal carried by that wavelength. Therefore, a given optical node needs as many regenerators as the number of wavelengths that need to be regenerated, and several regenerators share a common ROADM. Figure 1.1 demonstrates a simplified optical network containing these devices.

A satisfying regenerator placement for a network is an assignment of ROADMS and optical regenerators which allows a predefined traffic pattern. A regenerator placement can be measured in two main ways:

- The number of optical regenerators placed in the network.
- The number of locations in which regenerators are placed.

In this work we deal with the second measure. The motivation behind the number of locations measure is to reduce the number of ROADMs used, as well as other common equipment and manpower required in an optical node.

We deal with two types of connectivity requirements. In the first, we are given fixed lightpaths between certain terminal vertices and a fixed integer \( d > 0 \), and we need to find a set of regenerator locations such that every set of \( d \) consecutive internal vertices of every path contains at least one regenerator location. We name this problem \( RLP_{\text{path}} \). In the second, we are only given pairs of terminals as well as a fixed integer \( d > 0 \), and we are required to find lightpaths which connect the given terminal pairs, and a set of regenerator locations such that every set of \( d \) consecutive internal vertices of every path contains at least one regenerator location. We name this problem \( RLP_{\text{req}} \). In both problems our goal is to...
minimize the number of regenerator locations being used. A formal definition of \( RLP_{\text{path}} \) and \( RLP_{\text{req}} \) is given in Chapter 2.

1.2 Related Work

For a general reference to optical networks, see [Gre92, Muk97, RSS09]. The issue of their current and future data transmission rates is widely discussed in [Bra90, Kla98]. As was mentioned in the previous section, there are many potential applications of optical networking. Examples of such applications are discussed in [DV93, Gre92].

Regenerator placement has been extensively studied from an engineering point of view (e.g. [KS01, SGSG04, YR05, CR07, PPK08, FGG\(^+\)10, SZF\(^+\)10]). Such works usually offer mainly simulations and heuristics. [CR07] contained the first theoretical result concerning regenerator placement, showing that the regenerator location problem with connection requests (i.e. without a given routing) is NP-HARD in the all-to-all case, i.e. when there is a request between every two vertices in the network.

[FMSM\(^+\)11] was the first thorough theoretical study of regenerator placement. They examined the problem of minimizing the number of locations of regenerators in the network. The focus of their work is in finding approximation algorithms, and/or proving approximation hardness for the general case and for specific topologies, such as rings, trees and paths. In addition, they investigated the behavior of the regenerator location problem when an upper bound is placed on the number of regenerators permitted in each node of the network.

[MSSZ12] is the first theoretical study of the regenerator location problem under the total number of regenerators cost function. In the problem discussed in this work, it is required to find a regenerator placement of minimum size, which satisfies multiple sets of traffic patterns (i.e. lightpaths).

The articles [FMM\(^+\)10a, FMM\(^+\)10b, FMM\(^+\)11] deal with minimizing the number of regenerators when traffic grooming is presented. That is, in networks which allow sharing of the same wavelength by at most \( g \) different paths with \( g \) being the grooming factor.

[MSWZ11] and [SVW\(^+\)12] consider regenerator placement in an online setting. That is, when input arrives on-line rather than all at once, and decisions need to be made such that the ratio between the cost of the solution and the cost of the (off-line) optimum (called the competitive ratio) at every point in time is as small as possible.

This work investigates the parameterized complexity of regenerator location placement. Parameterized complexity is a branch of computational complexity that focuses on classifying computational problems according to their inherent difficulty with respect to certain parameters of the input. A basic reference for parameterized complexity is [DF99]. The main parameter under investigation in this work is the treewidth of the input graph. The treewidth parameter, informally, represents the similarity of a graph to a tree. Graphs of bounded treewidth encapsulate many useful network topologies, and thus have been the center of many research works in the past three decades (e.g [RS86, Bod88, AP89, ZN98, FFL\(^+\)07, BKKS11, BHM11]). A formal definition of treewidth is given in Chapter 2.
1.3 Our Results

Both problems under consideration were shown to be NP-Hard in [FMSM+11]. Our goal is to better understand the parameters making these problems hard, and to separate the polynomial cases, from the NP-HARD cases depending on these parameters, and to show fixed parameter tractability where possible.

We first show a result regarding ring networks. It was proven in [FMSM+11] that $RLP_{path}$ is polynomial-time solvable in tree and ring networks. The problems $RLP_{path}$ and $RLP_{req}$ are identical in trees, and therefore $RLP_{req}$ is also polynomial-time solvable in this topology. We complete these results by showing that $RLP_{req}$ is polynomial-time solvable in ring networks.

We consider the treewidth of the network as one main parameter. The treewidth $tw(G)$ of a network $G$ is a measure for its structure, resembling the network’s level of similarity to a tree (an exact definition is given in Chapter 2). We show that: (1) $RLP_{req}$ is NP-HARD even when $d = 1$ and $tw(G) = 3$, (2) $RLP_{path}$ is fixed parameter tractable for $d = 2$ with the treewidth $tw(G)$ of the graph as the parameter, and (3) it is NP-HARD when $d = 5$ even for graphs of treewidth 2.

Next, we introduce the vertex load parameter for the problem $RLP_{path}$. The vertex load of a path set $P$, $L_v(P)$, is the maximum number of paths which share the same common vertex. We show that (1) limiting the vertex load to 4 leaves $RLP_{path}$ NP-HARD when $d = 3$, and (2) $RLP_{path}$ is polynomially solvable if both the treewidth and the vertex load are fixed (note that if only one of them is fixed the problem remains NP-HARD).

Finally, we consider the number of connections that need to be made. We show that (1) $RLP_{path}$ is NP-HARD even when $|P| = 2$ and $d = 2$, and (2) $RLP_{req}$ is fixed parameter tractable for $d = 1$.

To the best of our knowledge, this work is the first to consider the treewidth, the vertex load and the number of connections as parameters of $RLP_{path}$ and $RLP_{req}$.

The results are summarized in Table 1.1.

<table>
<thead>
<tr>
<th>$RLP_{path}$ $(G, d, P)$</th>
<th>$RLP_{req}$ $(G, d, Q)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP-HARD when $tw(G) = 2$ and $d = 5$</td>
<td>Polynomial in rings</td>
</tr>
<tr>
<td>NP-HARD when $L_v(P) = 4$ and $d = 3$</td>
<td>NP-HARD when $tw(G) = 3$ and $d = 1$.</td>
</tr>
<tr>
<td>$FPT$ in $tw(G)$ when $d = 2$</td>
<td>Polynomial when $tw(G)$ and $L_v(P)$ are constants for any $d$</td>
</tr>
<tr>
<td>$APX$-Hard when $</td>
<td>P</td>
</tr>
</tbody>
</table>

Table 1.1: Summary of results
The structure of the thesis is as follows. In Chapter 2 we give definitions of the terms used in this paper. In Chapter 3 we discuss $RLP_{req}$ on rings. In Chapter 4 we prove a hardness result concerning $RLP_{req}$ for graphs of treewidth at most 3. In Chapter 5 we provide results concerning $RLP_{path}$ on bounded treewidth and bounded vertex load (see Table 1.1). In Chapter 6 we deal with instances of $RLP_{path}$ and $RLP_{req}$ with limited number of paths and requests, respectively. Finally, in Chapter 7 we conclude by presenting several open problems and possible extensions to this work.
Chapter 2
Preliminaries

Given an undirected underlying graph $G = (V(G), E(G))$ that corresponds to the network topology, a lightpath is a simple path in $G$. $\mathcal{P} = \{\pi_1, \pi_2, \ldots, \pi_n\}$ is a set of paths in $G$ representing the lightpaths. $\mathcal{Q} = \{q_1, q_2, \ldots, q_n\}$ is a set of communication requests between the vertices of $G$, i.e. $q_i = \{s_i, t_i\}$ for some $s_i, t_i \in V$, for every $1 \leq i \leq n$. Given a subset $U \subseteq V(G)$, $G[U]$ denotes the subgraph of $G$ induced by $U$. For any subset $Q \in \mathcal{Q}$ of requests, $\text{term}(Q) \overset{\text{def}}{=} \bigcup Q$ denotes the set of terminals of $Q$. The length $\ell(\pi)$ of a lightpath $\pi$ is the number of its edges. The internal vertices (resp. edges) of a path $\pi$ are the vertices (resp. edges) in $\pi$ except the first and the last ones. Given a set $\mathcal{P}$ of paths, a vertex $v \in V$ and an edge $e \in E$, $\mathcal{P}_v$ denotes the set of paths of $\mathcal{P}$ traversing $v$ (i.e. having $v$ as internal vertex), and $\mathcal{P}_e$ is the set of paths of $\mathcal{P}$ containing $e$. The load induced by $\mathcal{P}$ on $e$ and $v$ are $L_v(\mathcal{P}, v) \overset{\text{def}}{=} |\mathcal{P}_v|$ and $L(\mathcal{P}, e) \overset{\text{def}}{=} |\mathcal{P}_e|$. Finally $L_v(\mathcal{P}) \overset{\text{def}}{=} \max \{L_v(\mathcal{P}, v) | v \in V\}$ and $L(\mathcal{P}) \overset{\text{def}}{=} \max \{L(\mathcal{P}, e) | e \in E\}$.

Given two vertices $u, v \in V$, we define $\text{dist}(u, v)$ as the distance of a shortest path between $u$ and $v$ in $G$. If $u$ and $v$ are vertices of a common path $\pi \in \mathcal{P}$, we define $\text{dist}_\pi(u, v)$ as the number of edges between $u$ and $v$ on $\pi$.

A routing of $\mathcal{Q}$ is a set of paths $\mathcal{P}$ such that for every request $q_i = \{s_i, t_i\} \in \mathcal{Q}$, there is a path $\pi_i \in \mathcal{P}$ between $s_i$ and $t_i$. A regenerator location assignment is a set $R \subseteq V$.

Given an integer $d$, a lightpath $\pi$ is $d$-satisfied by a set of regenerator locations $R$ if for any $d$ consecutive internal vertices $v_1, v_2, \ldots, v_d$ of $\pi$, $v_i \in R$ for some $1 \leq i \leq d$. When there is no ambiguity about the set of regenerator locations under consideration we simply say that $\pi$ is $d$-satisfied.

A set of lightpaths is $d$-satisfied if every lightpath in it is $d$-satisfied. Note that a path with at most $d$ edges is $d$-satisfied regardless of $R$, therefore we assume without loss of generality that every path $\pi \in \mathcal{P}$ has at least $d + 1$ edges. We assume, without loss of generality, that every edge of the graph is used by at least one path $\pi \in \mathcal{P}$. Figure 2.1 demonstrates the terms of regenerator location placement and $d$-satisfiability when $d = 2$. 

7

Technion - Computer Science Department - M.Sc. Thesis MSC-2012-11 - 2012
We consider two variants of the Regenerator Location Problem: (a) given a graph $G = (V, E)$, a set $\mathcal{P}$ of paths in $G$, and a distance $d \geq 1$, find a regenerator location assignment $R \subseteq V$ of minimum cardinality such that all the paths in $\mathcal{P}$ are $d$-satisfied. (b) given a graph $G = (V, E)$, a set $\mathcal{Q}$ of requests in $G$, and a distance $d \geq 1$ find a routing $\mathcal{P}$ of $\mathcal{Q}$ and a regenerator location assignment $R \subseteq V$ of minimum cardinality such that all the paths in $\mathcal{P}$ are $d$-satisfied. Formally:

**Regenerator Location Problem** ($RLP_{path}$)

**Input:** An undirected graph $G = (V, E)$, a set $\mathcal{P}$ of paths in $G$, $d \geq 1$

**Output:** A regenerator location assignment $R \subseteq V$ such that every path $P \in \mathcal{P}$ is $d$-satisfied.

**Objective:** Minimize $|R|$.

**Routing and Regenerator Location Problem** ($RLP_{req}$)

**Input:** An undirected graph $G = (V, E)$, a set $\mathcal{Q}$ of requests in $G$, $d \geq 1$

**Output:** A routing $\mathcal{P}$ of $\mathcal{Q}$, and a regenerator location assignment $R \subseteq V$ such that every path $P \in \mathcal{P}$ is $d$-satisfied.

**Objective:** Minimize $|R|$.

Figures 2.2 and 2.3 contain respectively examples of instances of $RLP_{path}$ and $RLP_{req}$.
Figure 2.2: An instance of $RLP_{path}$ when $d = 2$. In (a) we are given a network and 2 lightpaths, and in (b) we provide an optimal regenerator location assignment which 2-satisfies the paths in (a). The bold vertices are regenerator locations.

Figure 2.3: An instance of $RLP_{req}$ when $d = 2$. In (a) we are given a network and 2 pairs of vertices that need to be connected: $(s_1, t_1)$ and $(s_2, t_2)$. (b) contains an optimal choice of paths which connect the pairs in (a) and a regenerator location assignment which 2-satisfies these paths. Note that in this case only one regenerator location is needed.
Given a graph $G = (V, E)$, a tree decomposition of $G$ is a tree $\mathcal{T} = (\mathcal{B}, F)$, where $\mathcal{B} = \{B_1, B_2, ..., B_k\}$ is a set of subsets of $V$, such that the following three conditions are met:

1. $\bigcup_{1 \leq i \leq k} B_i = V$.

2. For every edge $\{u, v\} \in E$, $u, v \in B_i$ for some $1 \leq i \leq k$.

3. For every $1 \leq i, j, l \leq k$ such that $B_l$ is on the path from $B_i$ to $B_j$ in $\mathcal{T}$, $B_i \cap B_j \subseteq B_l$.

The elements $B_i$ are called \textit{bags}, to distinguish them from the elements of $V$. The first condition guarantees that every vertex of $V$ is in at least one bag of $\mathcal{B}$. The second condition provides the same guarantees for the edges of $E$. The third condition guarantees that for every $v \in V$, the set of bags containing $v$ forms a \textit{connected} component in $\mathcal{T}$.

The \textit{width} $\omega(\mathcal{T})$ of a tree decomposition $\mathcal{T} = (\mathcal{B}, F)$ is defined as the size of its largest bag plus 1, i.e., $\omega(\mathcal{T}) = \max \{|B| \mid B \in \mathcal{B}\} + 1$. The treewidth of a graph $G$, denoted as $tw(G)$, is defined as the width of the minimum-width tree decomposition of $G$. Many efficient algorithms for generally \textsc{NP-Hard} problems are known when the treewidth of the input graph is bounded. The concepts of tree decomposition and treewidth are demonstrated in Figure 2.4.

![Figure 2.4: An example of a tree decomposition of a graph. The tree in (b) is a tree decomposition of the graph in (a). Note that the width of the tree decomposition in (b) is 2, since every bag contains at most 3 vertices. Therefore the treewidth of the graph in (a) is at most 2.](image)

**Parameterized complexity:** In parameterized complexity theory, the complexity of an algorithm is expressed by two variables: (1) the size $n$ of the input, (2) a parameter $k$ depending on the input. A problem is called \textit{fixed parameter tractable}, or \textsc{FPT} in short, if it can be solved in time $f(k) \cdot p(n)$, where $f$ is a function depending solely on $k$ and $p$ is a polynomial in $n$. In this work we consider as parameters the treewidth $tw(G)$ of the graph, the vertex load $L_v(P)$ of the paths, and the number $|P|$ of paths (resp. $|Q|$ of requests) for $RLP_{\text{path}}$ (resp. $RLP_{\text{req}}$).
**Approximation Theory:** Given a number $\alpha > 1$, we say that a minimization problem (resp. maximization problem) $\Pi$ is $\alpha$-approximable if there exists a polynomial-time algorithm which finds for every instance of $\Pi$ a solution which is at most $\alpha$ (resp. at least $\frac{1}{\alpha}$) times the optimal solution. The class $\mathsf{Apx}$ stands for all the optimization problems which are approximable by a constant factor. We say that $\Pi$ has a *Polynomial-Time Approximation Scheme* (PTAS) if it is $(1 + \epsilon)$-approximable for every $\epsilon > 0$. Finally, we say that $\Pi$ is $\mathsf{Apx}$-Hard, if a PTAS for $\Pi$ implies a PTAS for every $\Pi' \in \mathsf{Apx}$. It is known that if $P \neq NP$, every $\mathsf{Apx}$-Hard problem does not admit a PTAS.

**Two fundamental problems:** We mention here two fundamental problems used in this thesis: the 3Sat problem and the Vertex Cover problem.

An instance $(X, \phi)$ of 3Sat (or any of its variants that we use in this work) is a set $\phi = \{\phi_i| 1 \leq i \leq m\}$ of clauses over a set of boolean variables $X = \{x_j| 1 \leq j \leq n\}$. $\bar{X} = \{\bar{x}_j| 1 \leq j \leq n\}$ is the set of all negative literals over $X$. Each clause $\phi_i = \ell_{i,1} \lor \ell_{i,2} \lor \ell_{i,3}$ is a conjunction of 3 literals from $X \cup \bar{X}$. The output for such an instance is a satisfying assignment $f : X \mapsto \{0,1\}$ such that all the clauses $\phi_i$ of $\phi$ are satisfied, i.e. at for least one literal $\ell_{i,k}(k \in \{1,2,3\})$ either $\ell_{i,k} = x_j \land f(x_j) = 1$ for some $1 \leq j \leq n$ or $\ell_{i,k} = \bar{x}_j \land f(x_j) = 0$ for some $1 \leq j \leq n$.

Given an undirected graph $G = (V,E)$, $C \subseteq V$ is a vertex cover of $G$ if for every $\{u,v\} \in E$, either $\{u,v\} \cap C \neq \emptyset$. The problem $\text{VERTEXCOVER}$ is defined as follows:

<table>
<thead>
<tr>
<th><strong>VERTEXCOVER</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> An undirected graph $G = (V,E)$.</td>
</tr>
<tr>
<td><strong>Output:</strong> A vertex cover $C$ of $G$.</td>
</tr>
<tr>
<td><strong>Objective:</strong> Minimize $</td>
</tr>
</tbody>
</table>
Chapter 3

\textbf{RLP}_{req} \textbf{ in rings}

In this chapter we show that \( \text{RLP}_{req} (G, d, Q) \) is polynomial-time solvable when the graph \( G \) is a ring.

It was proven in [FMSM+11] that \( \text{RLP}_{path} \) is polynomially solvable on rings. Note, that given a ring \( G = (V, E) \) and a request set \( Q \), there are exactly two routings for every \( q \in Q \). Therefore, given a request set \( Q \), by considering all the possible routings of \( Q \), and finding the optimal solution for \( \text{RLP}_{path} \) for each one of them, we can optimally solve \( \text{RLP}_{req} (G, d, Q) \) for every given positive integer \( d \). Unfortunately, there are \( 2^{|Q|} \) such possible routings. We show that it is actually sufficient to consider only \( O(|V|^3) \) routings in order to find an optimal solution.

We assume an arbitrary planar embedding of the graph \( G \), such that for every every \( u, v \in V \) there is one \textit{clockwise} path and one \textit{counterclockwise} path which connects \( u \) and \( v \). We denote by \( \rho(u, v) \) the clockwise path from \( u \) to \( v \), and by \( \ell (\rho(u, v)) \) the length of \( \rho(u, v) \). For convenience, we will abuse notation and also refer to \( \rho(u, v) \) as the set of its vertices.

Given \( r_1, r_2 \in V \), we define \( R_{r_1,r_2} \) as the set of all vertices except the internal vertices of \( \rho(r_1, r_2) \). We also define \( P_{r_1,r_2} \) as the set of all potential paths that are \( d \)-satisfied by \( R_{r_1,r_2} \), formally: \( P_{r_1,r_2} = \{ \rho(s, t) \mid \{s, t\} \in Q, \rho(s, t) \text{ is } d\text{-satisfied by } R_{r_1,r_2} \} \)

By the following observation one can determine in linear time whether \( \text{OPT}_{r_1,r_2} = \infty \).
Observation 3.1. $OPT_{r_1,r_2} \neq \infty$ if and only if for every for every $\{s,t\} \in Q$, either $\rho(s,t) \in \mathcal{P}_{r_1,r_2}$ or $\rho(t,s) \in \mathcal{P}_{r_1,r_2}$.

The next observation implies a connection between every feasible solution and $\mathcal{P}_{r_1,r_2}$.

Observation 3.2. For every feasible solution $S = (P,R)$ to RLP$_{req}$ $(G,d,Q)$ such that $r_1, r_2$ are consecutive in $R$, $P \subseteq \mathcal{P}_{r_1,r_2}$.

(This follows directly from the fact that $R \subseteq R_{r_1,r_2}$)

The following lemma implies a polynomial time algorithm for RLP$_{req}$ in ring topology.

Lemma 3.1. Let $r_1, r_2 \in V$, such that $\text{dist}(r_1, r_2) > d$ and $OPT_{r_1,r_2} \neq \infty$. A solution of size at most $OPT_{r_1,r_2}$ to RLP$_{req}$ $(G,d,Q)$ can be found in polynomial time.

Proof. Let $\pi = \rho(r_1, r_2)$, and let $(\{s,t\} \in Q$. Exactly one of the following three cases applies (see Figure 3):

1. $s, t \notin \pi$. Assume w.l.o.g that $\pi \subseteq \rho(s,t)$. As $\ell(\pi) > d$ and $\pi \cap R_{r_1,r_2} = \{r_1, r_2\}$, $\rho(s,t)$ is not d-satisfied by $R_{r_1,r_2}$. Therefore, $\rho(s,t) \notin \mathcal{P}_{r_1,r_2}$.

2. $s, t \in \pi$. Assume w.l.o.g that $\rho(s,t) \subseteq \pi$. By definition of RLP$_{req}$, $\text{dist}(s,t) > d$, and therefore $\ell(\rho(s,t)) > d$. However, $\rho(s,t)$ does not contain any internal vertices in $R_{r_1,r_2}$, and is therefore not d-satisfied by $R_{r_1,r_2}$. This implies that $\rho(s,t) \notin \mathcal{P}_{r_1,r_2}$.

3. $s \in \pi, t \notin \pi$. We consider three sub-cases:

   a) $\text{dist}(r_1, s) > d$. As $\rho(r_1,s) \subseteq \rho(t,s)$ does not contain any internal vertices in $R_{r_1,r_2}$, this implies that $\rho(s,t) \notin \mathcal{P}_{r_1,r_2}$.

   b) $\text{dist}(s, r_2) > d$. As $\rho(s,t)$ does not contain vertices in $R_{r_1,r_2}$ between $t$ and $r_2$, this implies that $\rho(s,t) \notin \mathcal{P}_{r_1,r_2}$.

   c) $\text{dist}(r_1, s) \leq d$ and $\text{dist}(s, r_2) \leq d$.

For all the cases except 3(c), either $\rho(s,t) \notin \mathcal{P}_{r_1,r_2}$ or $\rho(t,s) \notin \mathcal{P}_{r_1,r_2}$. As $OPT_{r_1,r_2} \neq \infty$, this implies that for every request $\{s,t\} \in Q$ which does not conform with case 3(c), exactly one of $\rho(t,s)$ and $\rho(s,t)$ is in $\mathcal{P}_{r_1,r_2}$ (that is, exactly one routing is possible for every feasible solution $(P,R)$ in which $r_1, r_2$ are consecutive in $R$). Define:

$$Q' \overset{df}{=} \{\{s,t\} \in Q | s \in \pi, t \notin \pi, \text{dist}(r_1, s) \leq d, \text{dist}(s, r_2) \leq d\}$$

(i.e. this is the set of requests in $Q$ which conform with case 3(c)).

For every $q \in Q \setminus Q'$, define $r(q)$ as the unique path in $P_{r_1,r_2}$ which connects the endpoints of $q$.

Let $S = (P,R)$ be a minimum size solution to RLP$_{req}$ $(G,d,Q)$ such that $r_1$ and $r_2$ are consecutive in $R$, i.e. $|R| = OPT_{r_1,r_2}$ and $R \subseteq R_{r_1,r_2}$. We show that another solution for RLP$_{req}$ $(G,d,Q)$ of size at most $|R|$ can be found in polynomial time.
Let $T \overset{\text{def}}{=} \cup Q' \cap R_{r_1,r_2}$ be the set of terminals of $Q'$ outside of $\pi$. Denote $T = \{t_1, t_2, ..., t_m\}$, with $\ell(\rho(r_2, t_1)) < \ell(\rho(r_2, t_2)) < ... < \ell(\rho(r_2, t_m))$.

Let $1 \leq j \leq m$ be the largest value such that for some $s \in \pi$, $\rho(s, t_j) \in \mathcal{P}$. If no such value exists, let $j = 0$. We compose a routing $P_j = P_{Q \setminus Q'} \cup P_{\leq j} \cup P_{> j}$ of $Q$ such that:

\[ P_{Q \setminus Q'} = \{r(q)|q \in Q \setminus Q'\} \]
\[ P_{\leq j} = \{\rho(s', t_{j'})|\{s', t_{j'}\} \in Q', s' \in \pi, j' \leq j\} \]
\[ P_{> j} = \{\rho(t_{j'}, s')|\{s', t_{j'}\} \in Q', s' \in \pi, j' > j\} \]

We now show that $P_j$ is $d$-satisfied by $R$. For every $q \in Q \setminus Q'$, $r(q)$ is the only path in $P_{r_1, r_2}$ which connects the endpoints of $q$. As $P \subseteq P_{r_1, r_2}$ and $P$ is $d$-satisfied by $R$, clearly $r(q)$ is $d$-satisfied by $R$. For every $(s', t_{j'}) \in Q'$, we have two cases:

- $j' \leq j$. By choice of $j$, $\rho(s, t_j) \in \mathcal{P}$, and therefore $\rho(s, t_j)$ is $d$-satisfied by $R$. As $\rho(r_2, x_{j'})$ is a subpath of $\rho(s, t_j)$, it is $d$-satisfied by $R$ as well. Since $\ell(\rho(s, r_2)) \leq d$ and $r_2 \in R$, it follows that $\rho(s', t_{j'})$ is also $d$-satisfied by $R$.

- $j' > j$. By choice of $j$, $(s', t_{j'}) \notin \mathcal{P}$, and therefore $(t_{j'}, s') \in \mathcal{P}$, which implies that $(t_{j'}, s')$ is $d$-satisfied by $R$.

Then, $P_{Q \setminus Q'}$, $P_{\leq j}$ and $P_{> j}$ are all $d$-satisfied by $R$. Therefore, $P_j$ is $d$-satisfied by $R$, which means that $(P_j, R)$ is a feasible solution to $RLP_{\text{req}} (G, d, Q)$.

Let $R_j$ be an optimal solution to $RLP_{rt}(R, d, P_j)$. As $P_j$ is $d$-satisfied by $R$, $R$ is a feasible solution of $RLP_{\text{path}} (R, d, P_j)$, and therefore $|R_j| < |R| = OPT_{r_1, r_2}$. However, as $P_j$
is a routing of the requests in $Q$ and $P_j$ is $d$-satisfied by $R_j$. $(P_j, R_j)$ is a feasible solution for $RLP_{req} (G, d, Q)$.

Note, that $P_j$ and $R_j$ depend only on the problem instance $(G, d, Q)$ and on $j$. Moreover, for every $1 \leq j \leq m$, $(P_j, R_j)$ is a feasible solution for $RLP_{req} (G, d, Q)$, and for some $1 \leq j' \leq k$, $|R_{j'}| \leq OPT_{r_1,r_2}$. Therefore, a solution of size at most $OPT_{r_1,r_2}$ can be obtained by computing $P_j$ and $R_j$ for every $1 \leq j \leq m$, and choosing the solution with the minimum size.

For every $1 \leq j \leq m$, $P_j$ can be computed in time linear in the number of requests (as for every $q \in Q \setminus Q'$, $r(q)$ depends only on $P_{r_1,r_2}$, and for every $q \in Q'$, the routing of $q$ depends only on $j$). Moreover, according to [FMSM+11], the problem $RLP_{path}$ is polynomially solvable on rings and therefore $R_j$ can be computed in polynomial time. As there are $|m| = O(n)$ possible values for $j$, finding the required solution can be done in polynomial time.

Now we can conclude the following result:

**Theorem 3.1.** $RLP_{req} (G, d, Q)$ can be solved in polynomial time.

**Proof.** Let $OPT$ be the size of an optimal solution for $RLP_{req} (G, d, Q)$. One of the following cases applies:

- $OPT = |R_D|$. In that case, $S_D$ is an optimal solution for $RLP_{req} (G, d, Q)$.
- $OPT = OPT_{r_1,r_2}$ for some $r_1, r_2 \in V$ such that $\ell(\rho(r_1, r_2)) > d$. In that case, clearly $OPT_{r_1,r_2} \neq \infty$, as there exist feasible solutions to $RLP_{req} (G, d, Q)$ ($S_D$ is one).

Now, recall that we can determine in linear time for every $r_1, r_2 \in V$ such that $\ell(\rho(r_1, r_2)) > d$ whether $OPT_{r_1,r_2} = \infty$ or $OPT_{r_1,r_2} \neq \infty$. Moreover, in case $OPT_{r_1,r_2} \neq \infty$, according to Lemma 3.1, a solution $S_{r_1,r_2}$ of size at most $OPT_{r_1,r_2}$ can be found in polynomial time.

Therefore, we can compute an optimal solution by choosing the minimum size solution between $S_D$, and $S_{r_1,r_2}$ for every $r_1, r_2 \in V$ such that $\ell(\rho(r_1, r_2)) > d$ and $OPT_{r_1,r_2} \neq \infty$. As the number of such pairs $\{r_1, r_2\}$ is at most $O(n^2)$, the optimal solution can be computed in polynomial time. □
Chapter 4

Hardness of $RLP_{req}$ in Bounded Treewidth Graphs

In this chapter we show that the problem $RLP_{req}(G,d,Q)$ is hard even when $tw(G) = 3$ and $d = 1$. The proof follows a technique used in [BHM11] to show that the Steiner Forest problem is hard even for graphs of treewidth 3.

First, we define the problem R-SAT. Given three boolean variables $a_1, a_2, a_3$, an R-clause $R(a_1, a_2, a_3)$ stands for $(a_1 = a_3) \lor (a_2 = a_3)$. Given a set of boolean variables $X = \{x_1, ..., x_n\}$, we say that a formula $\phi$ is an R-formula over $X$ if $R = R_1 \land R_2 \land ... \land R_m$ where $m$ is a non-negative integer and for every $1 \leq i \leq m$, $R_i$ is an R-clause composed of literals over $X \cup \{0, 1\}$. We assume that for every $1 \leq j \leq n$, $x_j$ appears in some clause of $\phi$. For example, the R-formula $R(x, y, 1) \land R(x, 0, z)$ stands for $((x = 1) \lor (y = 1)) \land ((x = z) \lor (0 = z))$.

The problem of deciding whether a given R-formula $\phi$ is satisfiable is called R-SAT and is proven in [BHM11] to be Np-Complete.

A formula $\phi$ is non-trivial if it contains both constants 0 and 1. The problem Non-trivial R-SAT is the problem of deciding whether a non-trivial R-SAT formula is satisfiable. It is easy to see that non-trivial R-SAT is also Np-HARD by reduction from regular R-SAT. Indeed, any R-SAT formula $\phi$ can be transformed into an equivalent non-trivial R-formula $\phi' = \phi \land R(0, 1, x)$, where $x \in X$. Our main result follows:

**Theorem 4.1.** $RLP_{req}(G,d,Q)$ is NP-HARD even when $tw(G) \leq 3$ and $d = 1$.

**Proof.** By reduction from non-trivial R-SAT. Given a non-trivial R-formula $\phi$ with $n$ variables and $m$ clauses, we construct an instance $(G,d,Q)$ of $RLP_{req}$ as follows (see example in Figure 4.1). $G = (V,E)$ with $V = A_0 \cup A_1 \cup (\bigcup_{i=1}^m C_i) \cup (\bigcup_{j=1}^n V_j)$ and $E = E_0 \cup E_1 \cup (\bigcup_{i=1}^m E_{C_i}) \cup (\bigcup_{j=1}^n E_{V_j})$ where:

- $A_0 = \{w_0, w_1\}$ and $A_1 = \{0,1\}$.
- $C_i = \{a_{i,1}, a_{i,2}, a_{i,3}, v_{i,0}, v_{i,1}, u_{i,0}, u_{i,1}\}_{i=1}^m$.
- $V_j = \{x_j, w_{j,0}, w_{j,1}\}_{j=1}^n$. 

16
\begin{itemize}
  \item \( E_0 = \{(w_0, 0)\} \) and \( E_1 = \{(w_1, 1)\} \)
  \item \( E_{C_i} = \left\{ \begin{array}{l}
    \{a_{i,1}, u_{i,0}\}, \{a_{i,1}, u_{i,1}\}, \{a_{i,2}, v_{i,0}\}, \{a_{i,2}, v_{i,1}\}, \\
    \{u_{i,0}, 0\}, \{v_{i,0}, 0\}, \{u_{i,1}, 1\}, \{v_{i,1}, 1\}, \\
    \{a_{i,1}, a_{i,3}\}, \{a_{i,2}, a_{i,3}\}
  \end{array} \right\}^{m}_{i=1}.
  \item \( E_V = \{\{x_j, w_{j,0}\}, \{x_j, w_{j,1}\}, \{w_{j,0}, 0\}, \{w_{j,1}, 1\}\}^{n}_{j=1}. \)
\end{itemize}

For every \( 1 \leq i \leq m, 1 \leq k \leq 3 \) denote by \( \ell_{i,k} \) the \( k \)'th literal in the \( i \)'th clause of \( \phi \). Define \( b_{i,k} = \ell_{i,k} \) if \( \ell_{i,k} = x_j \) for some \( 1 \leq j \leq n \), and \( b_{i,k} = w_{\ell_{i,k}} \) otherwise. The request set \( Q \) is defined as \( Q = \{\{a_{i,k}, b_{i,k}\} | 1 \leq i \leq m, 1 \leq k \leq 3\} \). Finally we set \( d = 1 \).

We have to show that \( \phi \) is satisfiable if and only if there is a solution to \( RLP_{req} (G, 1, Q) \) of size (exactly) \( n + 3m + 2 \). We show both directions:

\begin{enumerate}
  \item [\( a \)] Assume that \( \phi \) is satisfiable, and let \( f \) be a satisfying assignment. For convenience, we extend the definition of \( f \), such that \( f(0) = 0 \) and \( f(1) = 1 \).

  We construct a solution \( S = (P, R) \) to \( RLP_{req} (G, 1, Q) \) s.t. \(|R| = n + 3m + 2 \). We construct the regenerator location set \( R \). Initially, \( R = \{0, 1\} \). For every \( 1 \leq j \leq n \), if \( f(x_j) = 0 \), add \( w_{j,0} \) to \( R \), otherwise add \( w_{j,1} \). For every \( 1 \leq i \leq m \), if \( f(a_{i,1}) = 0 \), add \( u_{i,0} \) to the solution, otherwise add \( u_{i,1} \). Similarly, if \( f(a_{i,2}) = 0 \), add \( v_{i,0} \) to the solution, otherwise add \( v_{i,1} \). Finally, since \( f \) is a satisfying assignment, either \( f(\ell_{i,3}) = f(\ell_{i,1}) \) or \( f(\ell_{i,3}) = f(\ell_{i,2}) \). Add accordingly \( a_{i,1} \) or \( a_{i,2} \) to \( R \). Clearly, \(|R| = n + 3m + 2 \).

  Since \( d = 1 \), we need to show that we can find for each request a path, all of whose internal vertices belong to \( R \). For every two vertices \( s, t \in V \), we say that \( s \) and \( t \) are \( R \)-connected if there exists a path in \( G \) between \( s \) and \( t \) such that all its internal vertices are in \( R \). Then it suffices to see that for every request \( \{s, t\} \in Q \), \( s \) and \( t \) are \( R \)-connected.

  Let \( \{s, t\} \in Q \). Exactly one of the following cases applies:

  \begin{itemize}
    \item \( s = a_{i,1} \) (or equivalently \( s = a_{i,2} \)) and \( t = x_j \) where \( 1 \leq i \leq m, 1 \leq j \leq n \). Denote \( f(x_j) = c \). By the definition of \( R \), both \( u_{i,c}, w_{i,c} \in R \). Since also \( c \in R \), \( a_{i,1} \) and \( x_j \) are \( R \)-connected.
    \item \( s = a_{i,1} \) (or equivalently \( s = a_{i,2} \)) and \( t = w_c \) where \( 1 \leq i \leq m, c \in \{0, 1\} \). By construction, both \( u_{i,0} \) and \( w_c \) are in \( R \). Thus \( a_{i,1} \) and \( w_c \) are \( R \)-connected.
    \item \( s = a_{i,3} \). Assume that \( a_{i,1} \in R \) (otherwise, \( a_{i,2} \in R \) and the claim is similar). Then \( f(\ell_{i,1}) = f(\ell_{i,3}) \). Denote \( f(\ell_{i,1}) = c \). Then, similarly to the previous cases, \( a_{i,1} \) and \( R \)-connected with \( c \). As \( a_{i,1} \in R \), \( a_{i,3} \) is also \( R \) connected with \( c \). Since also \( f(\ell_{i,3}) = c \), by definition of \( R \), \( w_{i,c} \in R \) and therefore \( \ell_{i,3} \) is \( R \)-connected with \( c \). As \( c \in R \), it follows that \( a_{i,3} \) and \( \ell_{i,3} \) are also \( R \)-connected.
  \end{itemize}

  For every \( q \in Q \), Let \( \pi_q \) be a path in \( G \) connecting the endpoints of \( q \), such that all its internal vertices are in \( R \), and define \( P = \bigcup_{q \in Q} \{\pi_q\} \). Then \( S = (P, R) \) is a feasible solution to \( RLP_{req} (G, 1, Q) \) such that \(|R| = n + 3m + 2 \), as required.
Figure 4.1: An example of the constructed graph $G$ and request set $Q$ for $\phi = R(x_1, 0, x_2) \land R(x_2, x_3, 1) \land R(x_4, x_1, x_2)$
Assume that we have a solution $S = (P, R)$ such that $|R| = n + 3m + 2$. We define the assignment $f$ in the following manner for every $1 \leq j \leq n$:

$$f(x_i) = \begin{cases} 0 & \text{if } x \text{ is } R\text{-connected with } 0 \\ 1 & \text{otherwise} \end{cases}$$

Since $\phi$ is non-trivial, it contains as literals both the constants 0 and 1. Therefore, by the definition of $Q$, there are requests that contain $w_0$ and $w_1$ as endpoints. As $w_0$ and $w_1$ are only connected to 0 and 1 in $G$, respectively, it must be the case that 0, 1 $\in R$. For every $1 \leq j \leq n$, since we assumed that $x_j$ appears in some clause of $\phi$, $x_j$ must be an endpoint of some request in $Q$. Therefore, at least one of the vertices adjacent to $x_j$, $w_{j,0}$ or $w_{j,1}$, must be in $R$. For every $1 \leq i \leq m$, each of the vertices $a_{i,1}, a_{i,2}, a_{i,3}$ must be $R$-connected to either 0 or 1 (otherwise the corresponding request cannot be satisfied).

It is easy to see that at least three vertices in $\{a_{i,0}, u_{i,1}, v_{i,0}, v_{i,1}, a_{i,1}, a_{i,2}, a_{i,3}\}$ must host a regenerator for this condition to be true. Therefore, we need 2 regenerators for 0 and 1, at least 1 regenerator for every variable and at least 3 regenerators for every clause.

Now, since $|R| = n + 3m + 2$, it follows that exactly one of $w_{j,0}$ and $w_{j,1}$ is in $R$, and exactly three vertices in $\{u_{i,0}, u_{i,1}, v_{i,0}, v_{i,1}, a_{i,1}, a_{i,2}, a_{i,3}\}$ are in $R$. Therefore, 0 and 1 are not $R$-connected. Thus, for every $1 \leq i \leq m$, $a_{i,1}, a_{i,2}, a_{i,3}$ are $R$-connected to either 0 or 1 (but not both), and for every $1 \leq j \leq n$, $x_j$ is connected to either 0 or 1 (but not both).

Let $T = \{s|\{s, t\} \in Q \text{ for some } t \in V\}$. For every $c \in \{0, 1\}$, let $P_c$ be the set of vertices in $T$ which are $R$-connected to $c$. Then it is clear that $\{P_0, P_1\}$ is a partition of $T$. Moreover, for every request $(s, t) \in Q$, clearly $s$ and $t$ must be in the same part of the partition (otherwise, they cannot be $R$-connected). To conclude the proof, let $1 \leq i \leq m$. $a_{i,3}$ must be in the same part of the partition as either $a_{i,1}$ or $a_{i,2}$, since at least one of them must be in $R$. If it is $a_{i,1}$, then it follows that $\ell_{i,1}$ and $\ell_{i,3}$ are also in the same part of the partition. Therefore, by definition of $f$, we have that $f(\ell_{i,1}) = f(\ell_{i,3})$. In a similar manner, if $a_{i,3}$ and $a_{i,2}$ are in the same part of the partition, then $f(\ell_{i,2}) = f(\ell_{i,3})$. Therefore, for every $1 \leq i \leq m$, either $f(\ell_{i,1}) = f(\ell_{i,3})$ or $f(\ell_{i,2}) = f(\ell_{i,3})$. Thus, $f$ is a satisfying assignment for $\phi$.

It remains to prove that $tw(G) \leq 3$. We will construct a tree decomposition $T$ of the graph $G$ in the following manner as a union of several trees:

$$T = S \cup \bigcup_{1 \leq j \leq n} T_{v,j} \cup \bigcup_{1 \leq i \leq m} T_{c,i}$$

Where the trees $S$, $T_{v,j}$ and $T_{c,i}$ are defined as follows (see Figure 4.2):

$$S = (\{B_s, B_w\}, F_S)$$
$$T_{c,i} = (\{B_s, B_{c,i,1}, B_{c,i,2}, B_{c,i,3}, B_{c,i,4}, B_{c,i,5}, B_{c,i,6}\}, F_{c,i}) \quad \forall 1 \leq i \leq m$$
$$T_{v,j} = (\{B_s, B_{v,j,1}, B_{v,j,2}\}, F_{v,j}) \quad \forall 1 \leq j \leq n$$
such that:

\begin{align*}
B_s &= \{0,1\} & B_{c,i,1} &= \{0,1,u_{i_0},v_{i_0}\} & B_{v,j,1} &= \{0,1,w_{j,0},w_{j,1}\} \\
B_w &= \{0,1,w_0,w_1\} & B_{c,i,2} &= \{1,a_{i,1},u_{i_1},v_{i_0}\} & B_{v,j,2} &= \{w_{j,0},w_{j,1},x_j\} \\
B_{c,i,3} &= \{1,a_{i,1},a_{i_2},v_{i_0}\} \\
B_{c,i,4} &= \{1,a_{i,1},a_{i_2},a_{i_3}\} \\
B_{c,i,5} &= \{1,a_{i,1},a_{i_2},u_{i_1}\} \\
B_{c,i,6} &= \{1,a_{i_2},v_{i_1}\}
\end{align*}

and:

\begin{align*}
F_S &= \{\{B_s, B_u\}\} \\
F_{c,i} &= \left\{ \{B_s, B_{c,i,1}\}, \{B_{c,i,1}, B_{c,i,2}\}, \{B_{c,i,2}, B_{c,i,3}\}, \{B_{c,i,3}, B_{c,i,4}\}, \{B_{c,i,4}, B_{c,i,5}\}, \{B_{c,i,5}, B_{c,i,6}\} \right\} \\
F_{v,j} &= \{\{B_s, B_{v,j,1}\}, \{B_{v,j,1}, B_{v,j,2}\}\}
\end{align*}

(4.1)

It is easy to see that $T$ meets all the three conditions required from a tree decomposition. Moreover, every bag in $T$ contains at most 4 vertices, and therefore the decomposition has width 3. Therefore, the treewidth of $G$ is at most 3. 

\hfill $\square$

Figure 4.2: A tree decomposition for the graph $G$
Chapter 5

\textit{RLP}\textsubscript{path} with Treewidth and Vertex Load Parameters

In this chapter we consider two parameters of the input for \textit{RLP}\textsubscript{path} \((G,d,\mathcal{P})\): The treewidth of \(G\) and the vertex load of \(\mathcal{P}\). In Section 5.1 we show that when \(d = 2\), \textit{RLP}\textsubscript{path} is FPT when parameterized by \(tw(G)\). However, we show in Section 5.2 that when \(d = 5\), \textit{RLP}\textsubscript{path} is \textsc{NP-Hard} even when \(tw(G) = 2\).

In Section 5.3 we consider the vertex load parameter, and show that \textit{RLP}\textsubscript{path} is \textsc{NP-Hard} when \(L_v(\mathcal{P}) = 4\) and \(d = 3\). We conclude this chapter in Section 5.4 by showing that \textit{RLP}\textsubscript{path} is polynomial-time solvable when both the treewidth and the vertex load are fixed, for every value of \(d\).

5.1 \textit{RLP}\textsubscript{path} is FPT in \(tw(G)\) for \(d = 2\)

In this section we show that the problem \textit{RLP}\textsubscript{path} is FPT for \(d = 2\) when parameterized by the treewidth of the input graph (note that \textit{RLP}\textsubscript{path} is trivial when \(d = 1\) even for general graphs).

Theorem 5.1. \textit{RLP}\textsubscript{path} \((G,2,\mathcal{P})\) is FPT when parameterized by \(tw(G)\).

Proof. By reduction to \textsc{VertexCover} . Let \(E' \subseteq E\) be the set of edges \(e \in E\) such that \(e\) is an internal edge of some path \(\pi \in \mathcal{P}\), and let \(G'\) be the subgraph of \(G\) induced by \(E'\). We show that \(R \subseteq V\) is a feasible solution for \textit{RLP}\textsubscript{path} \((G,2,\mathcal{P})\) if and only if \(R\) is a vertex cover of \(G'\).

Let \(R\) be a feasible solution of \textit{RLP}\textsubscript{path} \((G,2,\mathcal{P})\). It is easy to see that every edge of \(E'\) is covered by \(R\). Indeed, consider an edge \(\{u,v\} \in E'\). \(e\) is an internal edge of some path \(\pi \in \mathcal{P}\). \(\pi\) is 2-satisfied by \(R\), therefore \(\{u,v\} \cap R \neq \emptyset\). It follows that \(R\) is a vertex cover of \(G'\).

Conversely, let \(R\) be a vertex cover of \(G'\). We will show that every path \(\pi \in \mathcal{P}\) is 2-satisfied. Assume, by contradiction that some \(\pi \in \mathcal{P}\) is not 2-satisfied. Then, there are two consecutive internal vertices \(u,v\) such that \(u,v \notin R\). Then the edge \(e = \{u,v\}\) of \(G'\) is not
covered by $R$. We conclude that every path of $\mathcal{P}$ is 2-satisfied, i.e. $R$ is a feasible solution for $RLP_{\text{path}}(G, 2, \mathcal{P})$.

Now, VertexCover is known to be FPT when parameterized by treewidth. Clearly, the size of $G'$ is linear in the size of $G$ and the reduction above can be done in polynomial time. Moreover, since $G'$ is a subgraph of $G$, $tw(G') \leq tw(G)$. Thus, we conclude that $RLP_{\text{path}}(G, 2, \mathcal{P})$ is FPT when parameterized by $tw(G)$.

\section{$RLP_{\text{path}}$ is Np-Hard for $tw(G) \leq 2$ and $d = 5$}

In Theorem 5.1 we proved that $RLP_{\text{path}}$ is polynomial time solvable for any fixed any treewidth when $d = 2$. In this section we show that this does not hold for every value of $d$. Specifically, we show that when $d = 5$, the problem becomes Np-HARD even for graphs of treewidth at most 2.

\begin{theorem}
$RLP_{\text{path}} (G, 5, \mathcal{P})$ is Np-HARD, even when $tw(G) = 2$.
\end{theorem}

\begin{proof}
We show this by reduction from Monotone One-in-Three 3SAT. Given a instance $(X, \phi)$ of 3SAT, we say that $\phi$ is monotone if it contains no negative literals. We say that a truth assignment to $\phi$ is one-in-three if exactly one literal is assigned a true value in each clause. We define Monotone One-in-Three 3SAT as the problem of finding a satisfying one-in-three truth assignment to a given monotone 3CNF formula. This variation of 3SAT was proven to be Np-HARD by Schaefer in [Sch78].

We now describe the reduction. Let $(X, \phi)$ be an instance of 3SAT such that $\phi$ is monotone 3CNF and denote $X = \{x_j|1 \leq j \leq n\}$ and $\phi = \bigvee_{i=1}^{m} \phi_i$ where $\phi_i = x_{\alpha_i} \lor x_{\beta_i} \lor x_{\gamma_i}$ for every $1 \leq i \leq m$. We construct a graph $G = (V, E)$ as follows:

$$V = \{v_{\text{start}}, v_{\text{end}}\} \cup X \cup \bar{X} \cup \{t_j^1, t_j^2, t_j^3, t_j^4, t_j^5, t_j^6, t_j^7, t_j^8, t_j^9, t_j^{10}|1 \leq j \leq n\}$$

where $\bar{X} = \{\bar{x}_j|1 \leq j \leq n\}$. The edge set of $G$ is:

$$E = \{\{v_{\text{start}}, x_j\}, \{x_j, \bar{x}_j\}, \{x_j, v_{\text{end}}\}, \{\bar{x}_j, v_{\text{end}}\},$$
$$\{t_j^1, t_j^2\}, \{t_j^2, x_j\}, \{\bar{x}_j, t_j^3\}, \{t_j^3, t_j^4\}, \{t_j^4, t_j^5\},$$
$$\{t_j^6, t_j^7\}, \{t_j^7, x_j\}, \{\bar{x}_j, t_j^8\}, \{t_j^8, t_j^9\}, \{t_j^9, t_j^{10}\}|1 \leq j \leq n\}. \tag{5.1}$$

$\mathcal{P}$ is the union of three sets

$$\mathcal{P}_1 = \{(t_j^1, t_j^2, x_j, \bar{x}_j, t_j^3, t_j^4, t_j^5)|1 \leq j \leq n\}$$
$$\mathcal{P}_2 = \{\pi_i|1 \leq i \leq m\}$$
$$\mathcal{P}_3 = \{\pi_{\alpha_i, \beta_i, \gamma_i}, \pi_{\alpha_i, \gamma_i}|1 \leq i \leq m\}$$

where $\pi_i$ is the path $(t_{\alpha_i}^1, x_{\alpha_i}, v_{\text{end}}, x_{\beta_i}, v_{\text{start}}, x_{\gamma_i}, t_{\gamma_i}^2)$ and $\pi_{k,l}$ is the path $(t_{k}^1, t_{k}^3, \bar{x}_k, v_{\text{end}}, \bar{x}_l, t_{l}^3, t_{l}^4)$. An example of the construction is shown in Figure 5.1.
First, we give some intuition on the roles of the paths. The paths in $P_1$ are responsible for assigning exactly one truth value to each variable. The paths in $P_2$ ensure that at least one variable in every clause is assigned a true value. Finally, the paths in $P_3$ ensure that at least one in every pair of variables of every clause is assigned a false value, i.e. at least two variables in every clause get a false value. Together, the paths guarantee a truth assignment in which exactly one variable in every clause is assigned a true value.

We claim that $\phi$ has a satisfying one-in-three assignment if and only if $RLP_{\text{path}} (G, 5, P)$ has solution of size at most $n$. We note that every path in $P$ has 5 internal vertices. Such a path is 5-satisfied by a solution $R$ if and only if it has at least one of its internal vertices is in $R$.

Assume that $\phi$ has a satisfying one-in-three assignment $f : X \rightarrow \{0, 1\}$. Let $R = \{\bar{x}_j \in \bar{X} | f(x_j) = 0\} \cup \{x_j \in X | f(x_j) = 1\}$. Clearly $|R| = n$. We now verify that $R$ is a feasible solution for $RLP_{\text{path}} (G, 5, P)$, i.e. every path of $P$ is 5-satisfied.

- Every path in $\pi \in P_1$ has both $x_j$ and $\bar{x}_j$ as an internal vertex for some $j$, and one of these two vertices is in $R$, thus $\pi$ is 5-satisfied.

- Consider a path $\pi_i \in P_2$. Since $\phi_i$ is satisfied by $f$, at least one of the variables $x_{\alpha_i}, x_{\beta_i}, x_{\gamma_i}$ is assigned a true value. Therefore $\pi_i$ is 5-satisfied.

- For every $1 \leq i \leq m$, since $f$ is a one-in-three assignment, two of $x_{\alpha_i}, x_{\beta_i}, x_{\gamma_i}$ are assigned a false value by $f$. Thus at least one of every pair of these variables has a false value. Therefore, the paths $\pi_{\alpha_i, \beta_i}, \pi_{\beta_i, \gamma_i}, \pi_{\alpha_i, \gamma_i}$ are 5-satisfied.

We conclude that $R$ is indeed a feasible solution.

Now assume that there exists a feasible solution $R$ of $RLP_{\text{path}} (G, 5, P)$ with $|R| \leq n$. We note that the $P$ consists of $n$ pairs of paths, where the set of internal vertices of these
pairs are mutually disjoint. Thus using the pigeonhole principle, each such pair has exactly one vertex in R. Moreover, this vertex has to be on both paths. The set of internal vertices of the two paths of a pair intersect in \{x_j, \bar{x}_j\} for some 1 ≤ j ≤ n. Therefore, either \(x_j \in R\) or \(\bar{x}_j \in R\). As \(|R| = n\), clearly \(R\) contains only these vertices, i.e. \(R \subseteq X \cup \bar{X}\).

Now, we define a truth assignment \(f\). For every 1 ≤ j ≤ n:

\[
f(x_j) = \begin{cases} 
1 & \text{if } x_j \in R \\
0 & \text{otherwise.}
\end{cases}
\]

We verify that \(f\) is a satisfying one-in-three assignment:

- \(f\) satisfies \(\phi\): For every 1 ≤ i ≤ m, the path \(\pi_i \in P_2\) has an internal vertex in \(R\). However, \(R \subseteq X \cup \bar{X}\). Therefore \(R\) contains a vertex in at least one of the vertices \(x_{\alpha_i}, x_{\beta_i}, x_{\gamma_i}\). This means that at least one of the variables \(x_{\alpha_i}, x_{\beta_i}, x_{\gamma_i}\) is assigned a true value by \(f\).

- \(f\) is a one-in-three assignment: For every 1 ≤ i ≤ m, the path \(\pi_{\alpha_i, \beta_i}\) intersects \(R \subseteq X \cup \bar{X}\). Therefore, either \(x_{\alpha_i}\) or \(x_{\beta_i}\) is in \(R\). In other words \(f(x_{\alpha_i}) = 0\) or \(f(x_{\beta_i}) = 0\). We conclude similarly that any two of \(f(x_{\alpha_i}), f(x_{\beta_i}), f(x_{\gamma_i})\) is 0, i.e. at most one of them is 1.

To conclude the proof, it remains to see that \(tw(G) \leq 2\). We construct a tree decomposition \(\mathcal{T}\) of \(G\) in the following manner as a union of graphs:

\[
\mathcal{T} = \bigcup_{1 \leq j \leq n} \mathcal{T}_j
\]

where the graph \(\mathcal{T}_j = \{B_j, F_j\}\) is described as follows (see Figure 5.2 for every 1 ≤ j ≤ n:

\[
B_j = \{B_1, B_2, B_3, B_{a1}, B_{a2}, B_{b1}, B_{b2}, B_{c1}, B_{c2}, B_{c3}, B_{d1}, B_{d2}, B_{d3}\}
\]

\[
F_j = \begin{cases} 
\{B_1, B_2\}, \{B_2, B_3\}, \\
\{B_3, B_{a1}\}, \{B_{a1}, B_{a2}\}, \\
\{B_3, B_{b1}\}, \{B_{b1}, B_{b2}\}, \\
\{B_3, B_{c1}\}, \{B_{c1}, B_{c2}\}, \{B_{c2}, B_{c3}\}, \\
\{B_3, B_{d1}\}, \{B_{d1}, B_{d2}\}, \{B_{d2}, B_{d3}\},
\end{cases}
\]

such that:

\[
B_1 = \{v_{start}, v_{end}\} \quad B_2 = \{v_{start}, v_{end}, x_j\} \quad B_3 = \{v_{start}, x_j, \bar{x}_j\}
\]

\[
B_{a1} = \{x_j, t_2^2\} \quad B_{a2} = \{t_2^2, t_1^1\}
\]

\[
B_{b1} = \{x_j, t_2^1\} \quad B_{b2} = \{t_2^1, t_2^6\}
\]

\[
B_{c1} = \{x_j, t_3^3\} \quad B_{c2} = \{t_3^3, t_1^4\} \quad B_{c3} = \{t_4^1, t_5^2\}
\]

\[
B_{d1} = \{x_j, t_8^8\} \quad B_{d2} = \{t_8^8, t_9^9\} \quad B_{d3} = \{t_j^6, t_1^{10}\}
\]

It is easy to see that \(\mathcal{T}\) meets all the three conditions required from a tree decomposition. Moreover, every bag in \(\mathcal{T}\) contains at most 3 vertices, and therefore the with of the decomposition is 2, i.e. \(tw(G) \leq 2\).
Figure 5.2: The component $\mathcal{T}_j$ in the tree decomposition $\mathcal{T}$ of the graph G
5.3 \textit{RLP}_{\text{path}} \textbf{is} \textbf{Np-Hard} \textbf{even when} \( L_v(\mathcal{P}) \leq 4 \) \textbf{and} \( d = 3 \)

We have seen that limiting the treewidth still leaves \( \text{RLP}_{\text{path}} \) \textbf{NP-HARD}. We now see that limiting the vertex load does not help either.

\textbf{Theorem 5.3.} \textit{RLP}_{\text{path}} \textbf{is} \textbf{NP-HARD} \textbf{even when} \( L_v(\mathcal{P}) \leq 4 \) \textbf{and} \( d = 3 \)

\textit{Proof.} By a reduction from 3Sat where every variable appears at most 3 times. This variation of 3Sat is known to be \textbf{NP-COMPLETE} (see [GJ90], p. 259).

Let \((X, \phi)\) be an instance of 3Sat in which every variable appears at most 3 times. We construct an instance \((G, 3, \mathcal{P})\) of \textit{RLP}_{\text{path}} as follows:

\[
V = X \cup \bar{X} \cup \{v_{\text{start}}_j, z_j, v_{\text{end}}_j \mid 1 \leq j \leq n\} \cup \{c_{\text{start}}_i, c_{\text{end}}_i \mid 1 \leq i \leq m\}
\]

\[
\mathcal{P} = \{\pi(j) \mid 1 \leq j \leq n\} \cup \{\pi_i \mid 1 \leq i \leq m\}
\]

where \( \pi(j) = (v_{\text{start}}_j, x_j, z_j, v_{\text{end}}_j) \) and \( \pi_i = (c_{\text{start}}_i, \ell_{i,1}, \ell_{i,2}, \ell_{i,3}, c_{\text{end}}_i) \). \( E \) is the set of all edges contained in at least one path of \( \mathcal{P} \).

We will show that \( \phi \) is satisfiable if and only if \textit{RLP}_{\text{path}} \((G, 3, \mathcal{P})\) has a feasible solution \( R \) of size \( n \). Assume that \( \phi \) is satisfiable, and let \( f : X \to \{0, 1\} \) be a satisfying assignment for it. Let \( R = \{\bar{x}_j \in \bar{X} \mid f(x_j) = 0\} \cup \{x_j \in X \mid f(x_j) = 1\} \). Clearly \( |R| = n \). We now verify that \( R \) is a feasible solution for \textit{RLP}_{\text{path}} \((G, 3, \mathcal{P})\).

For every \( 1 \leq j \leq n \), \( x_j \in R \) or \( \bar{x}_j \in R \), thus \( \pi(j) \) is satisfied. Furthermore, as \( f \) is a satisfying assignment, for each \( \phi_i \), at least one of the vertices \( \ell_{i,1}, \ell_{i,2}, \ell_{i,3} \) is in \( R \). Therefore, \( \pi_i \) is satisfied.

Conversely, let \( R \) be a feasible solution to \textit{RLP}_{\text{path}} \((G, 3, \mathcal{P})\) of size \( n \). Every path \( \pi(j) \) hosts at least one regenerator in an internal vertex. As these \( n \) paths are pairwise disjoint, every such path hosts exactly one regenerator. We conclude that for every \( 1 \leq j \leq n \), exactly one of \( x_j, \bar{x}_j \) and \( z_j \) hosts a regenerator. Let \( f : X \to \{0, 1\} \) be the following assignment:

\[
f(x_j) = \begin{cases} 
1 & \text{if } x_j \in R \\
0 & \text{otherwise.}
\end{cases}
\]

We now show that \( f \) is a satisfying assignment. For every \( 1 \leq i \leq m \), \( \pi_i \) is 3-satisfied by \( R \). Then one of \( \ell_{i,1}, \ell_{i,2}, \ell_{i,3} \) is in \( R \). Assume w.l.o.g that \( \ell_{i,1} \in R \). If \( \ell_{i,1} = x_j \) for some \( 1 \leq j \leq n \), then \( f(x_j) = 1 \) and \( \phi_i \) is satisfied by \( f \). Otherwise, \( \ell_{i,1} = \bar{x}_j \) for some \( 1 \leq j \leq n \), then \( f(x_j) = 0 \) and again \( \phi_i \) is satisfied by \( f \).

It remains to see that the \( L_v(\mathcal{P}) \leq 4 \). Indeed the vertices \( x_j \) and \( \bar{x}_j \) appear in exactly one path \( \pi(j) \), and in at most 3 paths in \( \pi_i \) (as every variable appears in at most 3 clauses). All other vertices appear in a single path. \( \square \)
5.4 \textit{RLP}_{\text{path}} is in P for bounded treewidth and vertex load

We have seen that when either the treewidth of the graph or the vertex load of the path set are fixed, the problem \textit{RLP}_{\text{path}} is NP-HARD. In this section we show that the problem is solvable in polynomial time when both the treewidth and the vertex load are fixed.

\textbf{Theorem 5.4.} An optimal solution for \textit{RLP}_{\text{path}} \((G,d,P)\) can be computed in polynomial time, if both \textit{tw}(G) and \(L_v(P)\) are fixed.

In Section 5.4.1 we introduce definitions used in the algorithm and its analysis. In Section 5.4.2 we present an optimal algorithm for \textit{RLP}_{\text{path}} \((G,d,P)\). Finally, in Section 5.4.3 we prove the correctness of the algorithm, and show that it returns an optimal solution in polynomial time when \(\text{tw}(G) \leq w\) and \(L_v(P) \leq \ell\) for two constants \(\omega\) and \(\ell\).

5.4.1 Definitions and Basic Observations

In this section we assume that every path \(\pi \in P\) is assigned an arbitrary and fixed direction, designating one endpoint of \(\pi\) as its left endpoint, denoted as \(\text{left}(\pi)\), and the other as its right endpoint. Given a path \(\pi\), and two vertices \(u,v \in \pi\), we say that \(u\) is on the left of \(v\) in \(\pi\) if \(u\) is in the sub-path of \(\pi\) connecting \(\text{left}(\pi)\) and \(v\), and denote this by \(u \leq_{\pi} v\). If \(u \leq_{\pi} v\) and \(u\) is adjacent to \(v\) in \(\pi\), we say that \(u\) is the left neighbor of \(v\) in \(\pi\) and denote \(u\) as \(\text{left}_{\pi}(v)\).

Given a set \(R \subseteq V\) of regenerator locations, the function \(a_R\) indicates for each path and each internal node of it, the distance of the closest regenerator location when moving towards the left of the path. Formally, for a set \(U \subseteq V\) we define the set \(\mathcal{D}_U \overset{\text{def}}{=} \bigcup_{u \in U} \{u\} \times \mathcal{P}_u\), and \(a_R : \mathcal{D}_V \mapsto \mathbb{N}\) is an integer function \(\mathcal{D}_U\), such that:

\[
a_R(v,\pi) = \begin{cases} 
0 & \text{if } v \in R \\
1 & \text{if } v \notin R \text{ and } \text{left}_{\pi}(v) = \text{left}(\pi) \\
a_R(\text{left}_{\pi}(v),\pi) + 1 & \text{otherwise}
\end{cases}
\]

Clearly, if \(R\) \(d\)-satisfies \(\mathcal{P}\), then \(a_R(v,\pi) \in \mathbb{Z}_d = \{0,1,\ldots,d-1\}\). We denote by \(\mathcal{F}_U\) the set of all functions from \(\mathcal{D}_U\) into \(\mathbb{Z}_d\), i.e. \(\mathcal{F}_U \overset{\text{def}}{=} \mathbb{Z}_d^{\mathcal{D}_U}\). Clearly, \(|\mathcal{D}_U| = \sum_{u \in U} |\mathcal{P}_u| \leq \ell \cdot |U|\), thus \(|\mathcal{F}_U| \leq d^{\ell|U|}\).

A function \(f \in \mathcal{F}_V\) is a legal \(d\)-assignment if \(f = a_R\) for some \(R \subseteq V\) \(d\)-satisfying \(\mathcal{P}\). Define for every \(f \in \mathcal{F}_V\):

\[
R_f \overset{\text{def}}{=} \{ v \in V \mid \sum_{\pi \in \mathcal{P}_v} f(v,\pi) = 0 \}.
\]

\textbf{Claim 5.1.} \(f \in \mathcal{F}_V\) is a legal \(d\)-assignment if and only if for every \((v,\pi) \in \mathcal{D}_V\)

\[
\begin{align*}
f(v,\pi) = 0 & \Rightarrow \forall \pi' \in \mathcal{P}_v, f(v,\pi') = 0 \quad (5.3) \\
\text{left}_{\pi}(v) = \text{left}(\pi) & \Rightarrow f(v,\pi) \in \{0,1\} \quad (5.4) \\
\text{left}_{\pi}(v) \neq \text{left}(\pi) & \Rightarrow f(v,\pi) \in \{0,f(\text{left}_{\pi}(v),\pi) + 1\} \quad (5.5)
\end{align*}
\]
Proof. If $f$ is a legal $d$-assignment it is easy to see that conditions (5.3)-(5.5) are satisfied. Assume conversely that some $f \in \mathcal{F}_V$ meets these conditions, and let $R = R_f$. We first prove that $R$ $d$-satisfies $\mathcal{P}$. Indeed, let $\pi \in \mathcal{P}$ and let $u_1, u_2, \ldots, u_d$ be consecutive internal vertices on $\pi$ such that $u_1 \leq_\pi u_2 \leq_\pi \ldots \leq_\pi u_d$. Assume that $u_i \notin R$ for every $1 \leq i \leq d$. Then clearly $f(u_1) \geq 1$, and according to condition (5.5) $f(u_d) = f(u_{d-1}) + 1 = \ldots = f(u_1) + d - 1$, and therefore $f(u_d) \geq d$ which is a contradiction since $f(u_d) \notin Z_d$. Therefore, $R$ $d$-satisfies $\mathcal{P}$.

It is left to see that $f = a_R$. If $v \in R$, then by the definition of $R_f$ it follows that $f(u, \pi) = 0$. If $v \notin R$, then for some $\pi' \in P_u$, $f(u, \pi') \neq 0$, and therefore by condition (5.3), also $f(u, \pi) \neq 0$. If $\text{left}_\pi(v) = \text{left}(\pi)$, then by condition (5.4), $f(u, \pi) = 1$. Otherwise, by condition (5.5), $f(v, \pi) = f(\text{left}_\pi(v), \pi) + 1$. Therefore, $f$ can be defined in the exact same manner as $a_R$, which implies that $f = a_R$. \hfill \Box

If $f$ is a legal $d$-assignment such that $f = a_R$, then clearly $R = R_f$.

Corollary 5.1. Given a function $f \in \mathcal{F}_V$:

- It can be tested in polynomial time whether $f$ is a legal d-assignment.
- $R_f$ can be computed in polynomial time.

Given a $f \in \mathcal{F}_V$ and $U \subseteq V$, the restriction $f|_U$ of $f$ to $U$ is the function $f|_U : \mathcal{F}_V$ where $f|_U(v, \pi) = f(v, \pi)$ for every $(v, \pi) \in \mathcal{D}_U$. We say that $f, f' \in \mathcal{F}_V$ agree on $U \subseteq V$ (and denote by $f \cong_U f'$) if $f|_U = f'|_U$. When $f \in \mathcal{F}_U$ and $f' \in \mathcal{F}_{U'}$ agree on $U \cap U'$ we say simply that $f$ and $f'$ agree and denote this by $f \cong f'$. When these conditions do not hold, we write $f \not\cong_U f'$ and $f \not\cong f'$.

$g \in \mathcal{F}_U$ is a partial legal $d$-assignment over $U \subseteq V$ if $g = f|_U$, for some legal $d$-assignment $f \in \mathcal{F}_V$. We denote by $\mathcal{L}_U$ the set of all partial legal $d$-assignments over $U$. An optimal partial legal $d$-assignment over $U \subseteq V$ is a function $f^* \in \mathcal{L}_U$ minimizing $|R_{f^*}|$, i.e. $|R_{f^*}| = \min_{f \in \mathcal{L}_U} |R_f|$.

Claim 5.2. Let $U \subseteq V$ and $f \in \mathcal{F}_V$. $f \in \mathcal{L}_U$ if and only if the following conditions hold for every $u, v \in U$:

\begin{align}
&f(v, \pi) = 0 \Rightarrow \forall \pi' \in \mathcal{P}_v, f(v, \pi') = 0 \tag{5.6} \\
&f(v, \pi) \leq \text{dist}_\pi(\text{left}(\pi), v) \tag{5.7} \\
&u \leq_\pi v \Rightarrow f(v, \pi) < \text{dist}_\pi(u, v) \vee f(v, \pi) = f(u, \pi) + \text{dist}_\pi(u, v) \tag{5.8}
\end{align}

Proof. Assume that $f \in \mathcal{L}_U$. Then for some $f' \in \mathcal{F}_V$, $f = f'|_U$. Since $f'$ is a legal $d$-assignment, conditions (5.3)-(5.5) apply for $f'$. Let $u, v \in U$. First, if $f(v, \pi) = 0$, then also $f'(v, \pi) = 0$ and therefore according to condition (5.3) for every $\pi' \in \mathcal{P}_v$, $f'(v, \pi') = 0$ and thus also $f(v, \pi) = 0$. Second, according to condition (5.5), $f(v) = f'(v) \leq f(\text{left}_\pi(v)) + 1 \leq f(\text{left}_\pi(\text{left}_\pi(v))) + 2 \leq \ldots \leq f'(u, \pi) + \text{dist}_\pi(\text{left}(\pi), v) - 1$, such that $\text{left}_\pi(u) = \text{left}(\pi)$. According to condition (5.4), $f'(u, \pi) \leq 1$, and therefore $f(v) \leq \text{dist}_\pi(\text{left}(\pi), v)$. Finally, if $u \leq_\pi v$, we have two cases: (1) For some vertex $w$ on the subpath between $u$ and $v$,
In order to show that (5.6) is satisfied, let

\[ \text{Claim 5.4.} \]

If \( f(v, \pi) \leq f(w, \pi) + \text{dist}_\pi(w, v) = \text{dist}_\pi(w, v) \leq \text{dist}_\pi(u, v) \). (2) Otherwise, \( f(v, \pi) = f(\text{left}_\pi(v), \pi) + 1 = \ldots = f(u, \pi) + \text{dist}_\pi(u, v) \). Therefore conditions (5.6)-(5.8) are satisfied.

Conversely, assume that conditions (5.6)-(5.8) are all satisfied. For every \( (v, \pi) \in D_v \), we denote by \( \text{right}_{U,\pi}(v) \) the first internal vertex on the subpath from \( v \) to the right endpoint of \( \pi \) which is in \( U \). If no such vertex exists, \( \text{right}_{U,\pi}(v) \) is not defined. We define \( f' \in F_V \) in the following manner for every \( (v, \pi) \in D_v \):

\[
f'(v, \pi) = \begin{cases} 
  f(v, \pi) & \text{if } v \in U \\
  f(\text{right}_{U,\pi}(v), \pi) - \text{dist}_\pi(v, \text{right}_{U,\pi}(v)) & \text{if } v \notin U, \text{ right}_{U,\pi}(v) \text{ is defined and } f(\text{right}_{U,\pi}(v), \pi) \geq \text{dist}_\pi(v, \text{right}_{U,\pi}(v)) \\
  0 & \text{otherwise}
\end{cases}
\]

It is easy to see that conditions (5.3)-(5.5) are all satisfied by \( f' \). Thus, \( f' \) is a legal \( d \)-assignment, and \( f = f'|_U \), and therefore \( f \in L_U \).

\[ \square \]

\[ \text{Corollary 5.2. Given a function } f \in F_U \]

- it can be verified in polynomial time whether \( f \in L_U \),
- \( R_f \) can be computed in polynomial time.

\[ \text{Claim 5.3. Let } f \in F_U \text{ and } u \leq_\pi v \leq_\pi w. \text{ If both pairs } u, v \text{ and } v, w \text{ satisfy (5.8), then the pair } u, w \text{ satisfies (5.8) too.} \]

\[ \text{Proof. If } f(w, \pi) < \text{dist}_\pi(v, w) \text{ then clearly } f(w, \pi) < \text{dist}_\pi(u, w) \text{ and the condition holds for } u, w. \text{ Otherwise } f(w, \pi) = f(v, \pi) + \text{dist}_\pi(v, w). \text{ If } f(v, \pi) < \text{dist}_\pi(u, v) \text{ then } f(w, \pi) = f(v, \pi) + \text{dist}_\pi(v, w) < \text{dist}_\pi(u, v) + \text{dist}_\pi(v, w) = \text{dist}_\pi(u, v) \text{ and the condition holds for } u, v, \text{ otherwise } f(v, \pi) = f(u, \pi) + \text{dist}_\pi(u, v), \text{ therefore } f(w, \pi) = f(u, \pi) + \text{dist}_\pi(u, v) + \text{dist}_\pi(v, w) = f(u, \pi) + \text{dist}_\pi(u, w) \text{ and the condition holds.} \]}

Given \( U, U' \subseteq V, f \in L_U, f' \in L_{U'} \), such that \( f \equiv f' \), recall that their union \( f \cup f' \) is a function \( g \in F_{U \cup U'} \) such that

\[
g(v, \pi) = \begin{cases} 
  f(v, \pi) & \text{if } v \in U \\
  f'(v, \pi) & \text{otherwise.}
\end{cases}
\]

\[ \text{Claim 5.4. If } f \in L_U \text{ and } f' \in L_{U'}, \text{ then } g = f \cup f' \text{ satisfies conditions (5.6) and (5.7).} \]

\[ \text{Proof. In order to show that (5.6) is satisfied, let } g(v, \pi) = 0 \text{ and assume by contradiction that there is some } \pi' \in P, \text{ such that } g(v, \pi') \neq 0. \text{ Assume without loss of generality that } v \in U. \text{ Then } f(v, \pi) = g(v, \pi) = 0 \text{ and } f(v, \pi') = g(v, \pi') \neq 0, \text{ contradicting the assumption that } f \in L_U, \text{ and, in particular, that it satisfies (5.6). In order to show that (5.7) is satisfied, let } (v, \pi) \in D_{U \cup U'}. \text{ If } v \in U, \text{ then } g(v, \pi) = f(v, \pi) < \text{dist}_\pi(\text{left}(\pi), v) \text{ since } f \text{ satisfies (5.7). Otherwise, } g(v, \pi) = f'(v, \pi) < \text{dist}_\pi(\text{left}(\pi), v) \text{ since } f' \text{ satisfies (5.7). Therefore (5.7) is also satisfied in any case.} \]

\[ 29 \]
Note that \( f \cup f' \) is not necessarily a partial legal \( d \)-assignment as it might not satisfy (5.8).

**Observation 5.1.** Let \( U \subseteq V \), and \( f, f' \) be partial legal \( d \)-assignments over two sets that both include \( U \), and \( f \equiv f' \). Then

\[
R_{f \cup f'} = R_f \cup R_{f'} \\
R_f \mid_U = R_f \cap U.
\]

We end up this section by stating and proving a claim about tree decompositions.

**Claim 5.5.** Let \( B_i \) and \( B_k \) two adjacent bags in a tree decomposition \( \mathcal{T} \) of a graph \( G = (V, E) \), and let \( \mathcal{T}_1 \) and \( \mathcal{T}_2 \) be the subtrees obtained by the removal of the edge \( \{B_i, B_k\} \) from \( \mathcal{T} \). Let \( V_i \defeq \cup \mathcal{T}_i \) be the set of all nodes that reside in some bag of \( \mathcal{T}_i \). Let \( v_1 \in \mathcal{T}_1 \setminus \mathcal{T}_2 \) and \( v_2 \in \mathcal{T}_2 \setminus \mathcal{T}_1 \) two non-adjacent nodes of \( G \). Then every path \( \pi \) connecting \( v_1 \) and \( v_2 \) contains at least one node \( w \in B_i \cap B_k \).

**Proof.** Let \( \pi \) be a path in \( G \) connecting \( v_1 \) and \( v_2 \). We denote by \( w_1, w_2, \ldots, w_t \) the internal vertices of \( \pi \), ordered by distance from \( v_1 \), and for convenience we also say that \( w_0 = v_1 \) and \( w_{t+1} = v_2 \). Let \( B_1 \) be a bag in \( \mathcal{T}_1 \) which contains \( v_1 \) and let \( B_2 \) be a bag in \( \mathcal{T}_2 \) which contains \( v_2 \). For every \( 0 \leq i \leq t \), since \( (w_i, w_{i+1}) \in E \), then by the definition of a tree decomposition there exists a bag \( B_{i,i+1} \) in \( \mathcal{T} \) which contains both \( w_i \) and \( w_{i+1} \). Moreover, for every \( 1 \leq i \leq t \), both the bags \( B_{i-1,i} \) and \( B_{i,i+1} \) contain \( w_i \), and therefore by the definition of a tree decomposition there exists a path of bags \( \pi_i \) between \( B_{i-1,i} \) and \( B_{i,i+1} \) such that all the intermediate bags contain \( w_i \). Moreover, by similar arguments, there exists a path \( \pi_0 \) between \( B_1 \) and \( B_{0,1} \) such that all its intermediate bags contain \( w_0 = v_1 \), and there exists a path \( \pi_{t+1} \) between \( B_{t,t+1} \) and \( B_2 \) such that all its intermediate bags contain \( w_{t+1} = v_2 \). Let \( \pi_{B_1,B_2} \) be the path between \( B_1 \) and \( B_2 \) in \( \mathcal{T} \) formed by concatenating the paths \( \pi_0, \pi_1, \ldots, \pi_{t+1} \). Since \( B_1 \) is in \( \mathcal{T}_1 \) and \( B_2 \) is in \( \mathcal{T}_2 \), \( \pi_{B_1,B_2} \) must pass through the edge \( (B_1, B_k) \) in \( \mathcal{T} \). Therefore, for some \( 0 \leq i \leq t + 1 \), the path \( \pi_i \) passes through the edge \( (B_1, B_k) \). Therefore both \( B_i \) and \( B_k \) contain \( w_i \). \( \square \)

### 5.4.2 A dynamic programming algorithm for RLP\(_{\text{path}}\)

We now present a polynomial time dynamic programming algorithm solving optimally \( \text{RLP}_{\text{path}} (G, d, \mathcal{P}) \).

The algorithm works in three phases. In the initialization phase, we compute an optimal tree decomposition \( \mathcal{T} = (\mathcal{B}, \mathcal{F}) \) of \( G \), and for every bag \( B_i \in \mathcal{B} \) we set \( A_i \) to be the set of all possible partial legal \( d \)-assignments over \( B_i \). During the dynamic programming phase, we update \( A_i \) such that eventually it contains a set of legal \( d \)-assignments over \( V_i \), where \( V_i \) is the set of vertices appearing in some bag in the subtree of \( \mathcal{T} \) rooted at \( B_i \) (when \( \mathcal{T} \), without loss of generality, is assumed to be rooted at \( B_1 \)). One of these, as we will see, is a minimum-size \( d \)-assignment over \( V_i \). In the final phase, we obtain the minimum size assignment in \( A_1 \), \( f_{\text{min}} \), and declare \( R_{f_{\text{min}}} \) as an optimal solution for \( \text{RLP}_{\text{path}} (G, d, \mathcal{P}) \).

The full algorithm is presented in Algorithm 1.
Algorithm 1 ALGRLP \((G,d,P)\)

**Phase 1: Initialization**
1: Direct arbitrarily the paths in \(\pi\).
2: Compute a tree decomposition of \(G\) of width \(tw(G)\), \(T = (B,F)\), and direct \(T\) such that \(B_1\) is its root.
3: Compute \(L_{B_i}\) for every \(B_i \in B\).

**Phase 2: Dynamic Programming**
4: for every bag \(B_i\), traversed in postorder fashion do
5: \(A_i \leftarrow \emptyset\)
6: for every \(\hat{f} \in L_{B_i}\) do
7: Denote \(Ch_i = \{B_{k_1}, B_{k_2}, \ldots, B_{k_t}\}\) as the set of children of \(B_i\)
8: for every \(1 \leq j \leq t\) do
9: \(M \leftarrow \{f' \in A_{k_j} \mid f' \cong \hat{f}\}\)
10: \(f_{k_j} \leftarrow \arg\min_{f' \in M} |R_{f'}|\)
11: end for
12: \(\alpha_i(\hat{f}) \leftarrow \hat{f} \cup \bigcup_{j=1}^{t} f_{k_j}\)
13: \(A_i \leftarrow A_i \cup \{\alpha_i(\hat{f})\}\)
14: end for
15: end for

**Phase 3: Obtain an Optimal Solution**
16: \(f_{\min} \leftarrow \arg\min_{f' \in A_1} |R_{f'}|\)
17: return \(R_{f_{\min}}\)
5.4.3 Analysis of the algorithm

In this section we prove three lemmata showing that the algorithm returns a feasible solution, runs in polynomial time and optimal, respectively.

First however, we state two claims showing that the algorithm is valid. Recall that we consider $T$ as a directed tree rooted at $B_1$. We denote by $V_i$ the union of all $B_j$ where $B_j$ is a descendant of $B_i$ in $T$.

Claim 5.6. The set $M$ computed in line 9 is never empty.

Proof. Consider an execution of line 9. We refer as $B_i$, $\hat{f}$ and $j$ to the values of the respective variables at the time of the execution.

By definition of $\mathcal{L}_{B_i}$ there exists a function $f \in \mathcal{L}_V$ such that $\hat{f} = f|_{B_i}$. Note that $f|_{B_{kj}} \in \mathcal{L}_{B_{kj}}$. Therefore the function $\alpha_{kj}(f|_{B_{kj}})$ was computed when $B_{kj}$ was traversed in the outermost loop, and $\alpha_{kj}(f|_{B_{kj}}) \in A_i$. Since also $\alpha_{kj}(f|_{B_{kj}}) \simeq f|_{B_{kj}}$, we conclude that $\alpha_{kj}(f|_{B_{kj}}) \in M$. Therefore $M$ is not empty.

Claim 5.7. The union in line 12 is defined.

Proof. Note that for every $1 \leq j \leq t$, by choice of $f_{kj}$, $f_{kj} \simeq \hat{f}$. Moreover, since $T$ is a tree decomposition, for every $1 \leq j_1 < j_2 \leq t$, $V_{j_1} \cap V_{j_2} \subseteq B_i$. Therefore, $f_{kj_1} \simeq f_{kj_2}$. Thus, every pair of functions in the union agree, and therefore the union is defined.

The feasibility of the algorithm follows from the following lemma.

Lemma 5.1. If the functions $\hat{f}$ and $f_{kj}$ for every $1 \leq j \leq t$ in line 12 are all partial legal $d$-assignments, so is $f$.

Given this lemma, it easy to conclude from the code, that after $B_i$ is processed, $A_i \subseteq \mathcal{L}_{V_i}$. Consequently, every $f \in A_1$, in particular $f_{min}$ is in $\mathcal{L}_{V_1} = \mathcal{L}_V$. Therefore the solution $R_{f_{min}}$ returned by the algorithm $d$-satisfies $\mathcal{P}$.

Proof. For every $1 \leq j \leq t$, $f_{kj} \in \mathcal{L}_{V_{kj}}$ because $B_{kj}$ is a child of $B_i$, thus processed before $B_i$. Also, $f \in \mathcal{F}_{V_i}$ since $V_i = B_i \cup \bigcup_{j=1}^t B_{kj}$.

By Claim 5.4 $f$ satisfies conditions (5.6) and (5.7). It remains to show that it satisfies condition (5.8). In order to show this, let $u, v \in V_i$ and $\pi \in \mathcal{P}$ such that $u \leq \pi v$. We have to show that (5.8) is satisfied by $f$ for the pair $u, v$.

We distinguish between the following cases:

- $u, v \in B_i$: Then $\hat{f}(u, \pi) = f(u, \pi)$ and $\hat{f}(v, \pi) = f(v, \pi)$. As $f$ satisfies condition (5.6), and in particular for $u, v$, $\hat{f}$ also satisfies it for $u, v$.

- $u \in V_{kj} \\setminus B_i$ for some $1 \leq j \leq t$: If also $v \in V_{kj}$, then the same proof as the previous case applies here, by using $f_{kj}$ instead of $f$. 

32
Otherwise, by Claim 5.5 there exists a node \( w \), such that \( u \leq \pi w \leq \pi v \) and \( w \in B_i \cap B_{k_j} \subseteq B_i \cap V_{k_j} \) (possibly \( w = u \) or \( w = v \)). We have \( u \leq \pi w \) and \( u, w \in V_{k_j} \), therefore \( f_{k_j} \) satisfies (5.8) for \( u, v \). Clearly this holds for \( f \) that agrees with \( f_{k_j} \) at these nodes.

If \( v \in B_i \), then since also \( w \in B_i \), \( \hat{f} \) holds (5.8) for \( w, v \) and therefore \( f \) holds (5.8) for \( w, v \) as well. Thus, since \( u \leq \pi w \leq \pi v \) and \( f \) holds (5.8) for both \( u, w \) and \( w, v \), by Claim 5.3 \( f \) holds (5.8) for \( u, v \) as well.

If \( v \notin B_i \), then clearly \( v \in B_{k_i} \setminus B_i \), such that \( 1 \leq l \leq t \) and \( l \neq j \). Then by Claim 5.5 there exists a node \( w' \), such that \( w \leq \pi w' \leq \pi v \) and \( w' \in B_i \cap B_{k_i} \subseteq B_i \cap V_{k_i} \). By similar arguments as before we can show that \( f \) holds (5.8) for the pairs \( w, w' \) and \( w', v \). Thus, \( u \leq \pi w \leq \pi w' \leq \pi v \) and \( f \) holds (5.8) for the pairs \( u, w; w', w'; v \), by applying Claim 5.3 twice we conclude that \( f \) holds (5.8) for \( u, v \) as well.

\[ \square \]

**Lemma 5.2.** Algorithm 1 runs in polynomial time.

**Proof.** According to Bodlaender’s theorem [Bod93], a tree decomposition \( T = (B, F) \) of width \( \omega \) can be computed in time \( O(2^{\omega(n)} \cdot n) \), for some polynomial \( p \), i.e. in time linear in the input size. Clearly the number of bags \( |B| \) is also polynomial in the input size.

For every bag \( B_i \), \( L_{B_i} \) is computed by trying each \( f \in F_{B_i} \) and testing in polynomial time (by Corollary 5.2) whether it is a partial legal \( d \)-assignment. Therefore this step is completed in time at most \( |B| \cdot q(n) \cdot d^{\ell \cdot |B_i|} \leq |B| \cdot q(n) \cdot d^{\ell (\omega + 1)} \) for some polynomial \( q \), i.e. in polynomial time.

Every iteration of the innermost loop of the dynamic programming phase runs in time \( O(|Ch_i| \cdot |A_{k_j}|) = O(\|B\| \cdot d^{\ell (\omega + 1)}) \), i.e. polynomial in the input size. The total number of iterations of the innermost loop is \( |B| \cdot O(d^{\ell (\omega + 1)}) \leq n^2 \cdot O(d^{\ell (\omega + 1)}) \), i.e. polynomial in the input size. Therefore the dynamic programming phase runs in polynomial time.

Finally, by Corollary 5.2 phase 3 runs polynomial time.

The optimality of the algorithm will follow from the next lemma.

**Lemma 5.3.** After \( B_i \) is processed, for every \( \hat{f} \in L_{B_i} \), \( \alpha_i(\hat{f}) \) is optimal among all the \( d \)-assignments in \( L_{V_i} \) that agree with \( \hat{f} \). Formally, \( |R_{\alpha_i(f)}| = \min_{f \in L_{V_i} \land f \equiv \hat{f}} |R_f| \).

The optimality follows from the lemma by the following argument: By the above Lemma, at the end of the second phase, i.e. after \( B_1 \) is processed, for every \( \hat{f} \in L_{B_1} \), \( |R_{\alpha_1(\hat{f})}| = \min_{f \in L_{V} \land f \equiv \hat{f}} |R_f| \). We now observe that for every \( f \in L_{V}, f |_{B_1} \in L_{B_1} \) and \( f \equiv f |_{B_1} \). Therefore \( L_V \) can be partitioned into \( \cup_{f \in L_{B_1}} \{ f \in L_{V} \mid f \equiv \hat{f} \} \). We conclude that \( \min_{f \in L_V} |R_f| = \min_{f \in L_{B_1}} \min_{f \in L_{V} \land f \equiv \hat{f}} |R_f| \leq \min_{f \in L_{B_1}} |R_{\alpha_1(f)}| \geq \min_{f \in A_1} |R_f| \), i.e. the solution returned in phase 3 is optimal.
Proof. We prove the lemma by induction on the tree decomposition structure. If \( B_i \) is a leaf, then \( A_i = \mathcal{L}_{B_i} \). In this case, \( \alpha_i(\hat{f}) = \hat{f} \) is the only assignment in \( \mathcal{L}_{V_i} \) that agrees with \( \hat{f} \) and therefore the lemma follows immediately.

Assume now that the lemma holds for all the children of \( B_i \). Consider the iteration of the outer loop that considers \( B_i \) and an iteration of the inner loop that considers some \( \hat{f} \in \mathcal{L}_{B_i} \). Let \( f_{kj} \) be value assigned in line 10. For every \( 1 \leq j \leq t \), by the definition of \( f_{kj} \):

\[
|R_{f_{kj}}| = \min_{f' \in A_{kj} \land f' \equiv \hat{f}} |R_{f'}| = \min_{\hat{f} \in A_{kj} \land f' \equiv \hat{f}} \left( \min_{f' \in A_{kj} \land f' \equiv \hat{f}} |R_{f'}| \right).
\]

Now observe that in the last term \( f' \equiv \hat{f} \) if and only if \( \hat{f} \equiv \hat{f} \), since by the definition of a tree decomposition \( V_{kj} \cup B_i = B_{kj} \cup B_i \). Therefore:

\[
|R_{f_{kj}}| = \min_{\hat{f} \in A_{kj} \land f' \equiv \hat{f}} \left( \min_{f' \in A_{kj} \land f' \equiv \hat{f}} |R_{f'}| \right) = \min_{\hat{f} \in A_{kj} \land f' \equiv \hat{f}} |R_{\alpha_j(\hat{f})}|
\]

where the last equality follows from the fact that \( \alpha_j(\hat{f}) \) is the only assignment in \( A_{kj} \) that agrees with \( \hat{f} \). Therefore, by the induction hypothesis:

\[
|R_{f_{kj}}| = \min_{\hat{f} \in A_{kj} \land f' \equiv \hat{f}} \left( \min_{f' \in A_{kj} \land f' \equiv \hat{f}} |R_{f'}| \right) = \min_{\hat{f} \in A_{kj} \land f' \equiv \hat{f}} \left( \min_{f' \in A_{kj} \land f' \equiv \hat{f}} |R_{f'}| \right) = \min_{\hat{f} \in A_{kj} \land f' \equiv \hat{f}} |R_{f'}|
\]

where the middle equality is due to the fact that \( f \equiv \hat{f} \) if and only if \( \hat{f} \equiv \hat{f} \). Now, for every \( f \in \mathcal{L}_{V_i} \) such that \( f \equiv \hat{f} \):

\[
R_f = R_{\hat{f}} + \bigcup_{j=1}^{t} R_{f|_{V_{kj}}} = R_{\hat{f}} + \bigcup_{j=1}^{t} R_{f|_{V_{kj}}} \setminus R_{f|_{B_{i} \cap B_{kj}}}.
\]

Because \( f \) and \( \hat{f} \) agree on \( B_i \cap B_k \), this is equivalent to:

\[
R_f = R_{\hat{f}} + \bigcup_{j=1}^{t} \left( R_{f|_{V_{kj}}} \setminus R_{f|_{B_{i} \cap B_{kj}}} \right).
\]

Since the sets in the union in the right hand side are mutually disjoint and \( R_{f|_{B_{i} \cap B_{kj}}} \subseteq R_{f|_{V_{kj}}} \) for every \( 1 \leq j \leq t \):

34
\[ |R_f| = |R_f| + \sum_{j=1}^{t} \left( |R_f|_{V_{kj}} - |R_f|_{B_i \cap B_{kj}} \right) \]

Denote \( c(\hat{f}) = |R_{\hat{f}}| - \sum_{j=1}^{t} |R_f|_{B_i \cap B_{kj}} \). Then:

\[ |R_f| = c(\hat{f}) + \sum_{j=1}^{t} |R_f|_{V_{kj}} \]

Now, according line 12 in the algorithm:

\[ \alpha_i(\hat{f}) = \hat{f} \cup \bigcup_{j=1}^{t} f_{kj} \]

and therefore:

\[ R_{\alpha_i(f)} = R_f \cup \bigcup_{j=1}^{t} R_{f_{kj}} = R_f \cup \bigcup_{j=1}^{t} \left( R_{f_{kj}} \setminus R_{f_{kj}}|_{B_i \cap B_{kj}} \right) \]

Similarly we can show:

\[ |R_{\alpha_i(f)}| = c(\hat{f}) + \sum_{j=1}^{t} |R_{f_{kj}}| \]

Recall that for every \( 1 \leq j \leq t \), \( |R_{f_{kj}}| = \min_{f \in \mathcal{L}_{V_{kj}}} \wedge f \neq f |R_f| \), and thus \( |R_{f_{kj}}| \leq |R_f|_{V_{kj}} \).

Therefore:

\[ |R_{\alpha_i(f)}| \leq c(\hat{f}) + \sum_{j=1}^{t} |R_{f_{kj}}| = |R_f| \]

Since the last inequality holds for every \( f \in \mathcal{L}_{V_i} \), we conclude that:

\[ |R_{\alpha_i(f)}| = \min_{f \in \mathcal{L}_{V_i} \land f \neq f} |R_f| \]

as required. \( \square \)
Chapter 6

Fixed Number of Connections

In this chapter we consider another important parameter relevant for both problems $RLP_{path}$ and $RLP_{req}$, namely the number of connection requests. More precisely the parameter under consideration is the number $|P|$ of paths for $RLP_{path}$, and the number $|Q|$ of requests for $RLP_{req}$.

6.1 $RLP_{path}$ with fixed number of paths

In this section we show that $RLP_{path}$ is $\text{NP-Hard}$ even when $d = |P| = 2$. To see that this result draws an exact hardness boundary, we note that when $d = 1$ or $|P| = 1$, $RLP_{path}$ can be solved in linear time. Indeed when $d = 1$, every internal vertex of a path in $P$ must host a regenerator, and this constitutes a feasible solution, therefore optimal. When $|P| = 1$, let $P = \{\pi\}$. Clearly any solution contains at least $\lceil \frac{|\pi|}{d} \rceil$ locations, and such a solution can be easily found by letting $R$ be the vertices $d, 2d, \ldots$ of $\pi$.

We now proceed with the main result of this part. In fact, we prove a stronger result: we show that $RLP_{path}$ does not admit a polynomial-time approximation scheme unless $P = NP$.

**Theorem 6.1.** $RLP_{path} (G, d, \mathcal{P})$ is APX-HARD when $|\mathcal{P}| = 2$ and $d = 2$.

**Proof.** By reduction from $\text{VertexCover}$ in graphs of degree of at most 3. Let $G = (V, E)$ be an instance of $\text{VertexCover}$ where $V = \{v_1, v_2, \ldots, v_n\}$, $E = \{e_1, e_2, \ldots, e_m\}$, and $e_i = \{v_{x_i}, v_{y_i}\}$ for $1 \leq i \leq m$. For every $1 \leq j \leq n$, denote $I_j = \{i | e_i$ is incident to $v_j\}$, and mark $I_j = \{i_1, i_2, \ldots, i_{k_j}\}$ such that $i_1 < i_2 < \ldots < i_{k_j}$. We construct an instance $(G', 2, \mathcal{P})$ of $RLP_{path}$ (see Figure 6.1) where $G' = (V', E')$, $\mathcal{P} = \{\pi_1, \pi_2\}$ and $V' = \{t_{start}, t_{end}\} \cup U \cup O \cup S \cup A \cup B$ where

\[
U = \bigcup_{j=1}^{n}(U_j \cup \bar{U}_j), U_j = \{w_{ij} \mid i \in I_j\}, \bar{U}_j = \{\bar{u}_{ij} \mid i \in I_j\}
\]

\[
O = \{o_j \mid 1 \leq j \leq n\}
\]

\[
S = \{s_k \mid 1 \leq k \leq n + m\}
\]

\[
A = \{a_i \mid 0 \leq i \leq m\}
\]

\[
B = \{b_i \mid 0 \leq j \leq n\}
\]
We first describe intuitively the role of the vertices of $V'$. The vertices of $U_j$ are used to determine whether $v_j$ is included in the vertex cover. The vertices $o_j$ are used to require an additional regenerator for every vertex $v_j$ in the vertex cover. We will refer to the vertices of $S$ as separator vertices. We will guarantee that there is an optimal solution that contains all the separator vertices. Finally, the vertices of $A \cup B$ are auxiliary vertices that keep a distance of 2 between the separator vertices.

We now define the paths $\pi_1, \pi_2$. $E'$ is the set of all edges used by these paths.
\[ \pi_1 = (t_{\text{start}}, u_{i_1}, u_{i_1}^1, u_{i_1}^2, u_{i_2}, u_{i_2}^2, \ldots, u_{i_k}, u_{i_k}^1, o_1, s_1, \
\quad u_{i_2}^2, \bar{u}_{i_2}^2, u_{i_2}^2, \bar{u}_{i_2}^2, \ldots, u_{i_k}^2, \bar{u}_{i_k}^2, o_2, s_2, \
\quad \ldots) \]
\[ \pi_2 = (t_{\text{start}}, u_{1x}, u_{1x}+1, u_{2x}, u_{2y}, u_{2y}^2, s_{n+2}, \ldots, u_{m+x}, u_{m+y}, s_{n+m}, \
\quad b_0, s_1, b_1, s_2, b_2, \ldots, s_n, b_n, t_{\text{end}}) \]

We now show the following claim:

**Claim 6.1.** \( G \) has a vertex cover of size \( c \) if and only if \( RLP_{\text{path}} (G', 2, \mathcal{P}) \) has a solution of size \( 3m + n + c \).

**Proof.** Let \( C \) be a vertex cover of size \( c \) of \( G \). We define a set \( R \subseteq V' \) of regenerator locations that \( 2 \)-satisfy both \( \pi_1 \) and \( \pi_2 \) as follows:

\[ R = \left( \bigcup_{v_j \in C} U_j \right) \cup \left( \bigcup_{v_j \in C} \bar{U}_j \right) \cup S \cup \{ o_j \in O \mid v_j \in C \} . \]

The number of vertices in the first two sets composing \( R \) is \( |U|/2 = |\bigcup_{j=1}^n I_j| = 2|E| = 2m, |S| = m + n \), and the size of the fourth set is \( |C| = c \). As these sets are disjoint, we have \( |R| = 2m + m + n + c = 3m + n + c \).

We now verify that both \( \pi_1, \pi_2 \) are \( 2 \)-satisfied. \( S \subseteq R \), therefore it suffices to show that every sub-path obtained by subdividing \( \pi_1 \) or \( \pi_2 \) at the vertices of \( S \) is \( 2 \)-satisfied. Clearly if the length such a sub-path is at most \( 2 \) the claim holds. Consider a sub-path of \( \pi_1 \) with length more than \( 2 \). Indeed, there is one regenerator in every at most \( 2 \) vertices of it because for every \( v_j \in V \) (a) if \( v_j \in C \) then \( u_{ij} \in R \) for every \( i \in I_j \), and also \( o_j \in R \) (b) otherwise \( v_j \notin C \) and \( \bar{u}_{ij} \in R \) for every \( i \in I_j \). Now consider a sub-path of \( \pi_2 \) with length at least \( 3 \). Note that such a sub-path has exactly two internal vertices. It suffices to show that at least one of them is in \( R \). Indeed, because \( C \) is a vertex cover, either \( v_{xi} \in C \) or \( v_{yi} \in C \). Therefore either \( u_{ixi} \in R \) or \( u_{iyi} \in R \).

To prove the other direction, let \( R \) be a feasible solution \((G', 2, \mathcal{P})\) such that \( |R| = 3m + n + c \). Note that for every \( 1 \leq i \leq m \), either \( s_{n+i} \) or \( a_i \) is in \( R \), otherwise \( \pi_1 \) is not \( 2 \)-satisfied. Similarly, for every \( 1 \leq j \leq n \), either \( s_j \) or \( b_j \) is in \( R \). Therefore the solution \( R' = (R \setminus A \setminus B) \cup S \) satisfies \( |R'| \leq |R| \). Note that \( R' \) also \( 2 \)-satisfies \( \pi_1 \) and \( \pi_2 \), and clearly \( S \subseteq R' \).

Now let \( C = \{ v_j \in V \mid U_j \cap R' \neq \emptyset \} \). We show that \( C \) is a vertex cover of \( G \). Indeed, consider an arbitrary edge \( e_i \) of \( E \). Then \( u_{ix} \) and \( u_{iy} \) are consecutive internal vertices of \( \pi_2 \). Therefore, either \( u_{ix} \) or \( u_{iy} \) is in \( R' \), implying that either \( v_{xi} \) or \( v_{yi} \) is in \( C \), i.e. \( e_i \) is covered by \( C \). As \( e_i \) an arbitrarily edge of \( E \), \( C \) is a vertex cover of \( G \).
For each $1 \leq j \leq n$, at least one of $u_{ij}, \bar{u}_{ij}$ is in $R'$ for every $i \in I_j$, otherwise $\pi_1$ is not 2-satisfied. We consider two cases:

- $v_j \notin C$: Then $U_j \cap R' = \emptyset$, therefore $\bar{U}_j \subseteq R'$. If $o_j \notin R'$ then let $R'' = R'$, otherwise $R'' = R' \setminus \{o_j\}$ still 2-satisfies $\mathcal{P}$. In both cases $|R'' \cap (U_j \cup \bar{U}_j)| = |\bar{U}_j| = m$ and $|R''| \leq |R'|$.

- $v_j \in C$: Then $U_j \cap R' \neq \emptyset$. Let $u_{ij} \in R' \cap U_j$. There are $2(i - 1)$ vertices between $u_{ij}$ and $\bar{u}_{(i-1)j}$ on $\pi_1$, therefore at least $i - 1$ vertices among them are in $R'$. There are $2(m - i + 1)$ vertices between $\bar{u}_{ij}$ and $o_j$ on $\pi_1$, therefore at least $m - i + 1$ vertices among them are in $R'$. We conclude that $|R'' \cap (U_j \cup \bar{U}_j)| \geq (i - 1) + 1 + (m - i + 1) = m + 1$. Let $R'' = R' \setminus \bar{U}_j \cup U_j \cup \{o_j\}$. Clearly $|R'' \cap (U_j \cup \bar{U}_j)| = m + 1$, thus $|R''| \leq |R'|$.

Now $R''$ obtained from $R$ in this way satisfies

$$R'' = \left( \bigcup_{v_j \in C} U_j \right) \cup \left( \bigcup_{v_j \notin C} \bar{U}_j \right) \cup S \cup \{o_j \in O | v_j \in C\}$$

therefore $|R''| = 2m + n + m + |C| = 3m + n + |C|$. On the other hand $|R''| \leq |R'| \leq |R| = 3m + n + c$. We conclude that $|C| \leq c$. By adding $c - |C|$ vertices of $V$ to $C$ we get a vertex cover of size $c$.

Now let $C^*$ be an optimal vertex cover of $G$. Assume that $RLP_{\text{path}}$ has a PTAS when $d = |\mathcal{P}| = 2$. According to Claim 6.1, there is a solution $R$ to $RLP_{\text{path}} (G', 2, \mathcal{P})$ such that $|R| = 3m + n + |C^*|$. Let $\epsilon > 0$ and define $\epsilon' = \frac{\epsilon}{34}$. Since $RLP_{\text{path}}$ admits a PTAS when $d = |\mathcal{P}| = 2$, we can calculate a solution to $RLP_{\text{path}} (G', 2, \mathcal{P})$ of size at most:

$$(1 + \epsilon') |R| \leq (1 + \epsilon')(3m + n + |C^*|)$$

Therefore, according to Claim 6.1 we can calculate in polynomial time a vertex cover $C$ of $G$ such that:

$$|C| \leq (1 + \epsilon'(3m + n + |C^*|)) - (3m + n) = |C^*| + (3m + n + |C^*|)\epsilon'$$

Recall that $\Delta(G) \leq 3$, and therefore $m \leq \frac{3n}{2}$. Then:

$$|C| \leq |C^*| + \left( \frac{11n}{2} + |C^*| \right) \epsilon'$$

On the other hand, since every vertex in $C^*$ can cover at most 3 edges, $|C^*| \geq \frac{m}{3}$. Assuming that every vertex $v \in V$ is incident to some edge in $E$, this implies that $|C^*| \geq \frac{n}{3}$, i.e., $n \leq 6|C^*|$. Therefore:

$$|C| \leq |C^*| + \left( \frac{66|C^*|}{2} + |C^*| \right) \epsilon' \leq |C^*| + 34|C^*|\epsilon' \leq (1 + \epsilon)|C^*|$$

39
Since we started from an arbitrary value of $\epsilon$, we conclude that if $RLP_{path}$ has a PTAS when $d = |P| = 2$, then $\text{VERTEXCOVER}$ has a PTAS when $\Delta(G) \leq 3$. As the latter problem is APX-HARD, this means that if $RLP_{path}$ has a PTAS, every problem $\Pi \in \text{APX}$ has a PTAS. Therefore, $RLP_{path}$ is APX-HARD when $|P| = 2$ and $d = 2$.

To conclude this section, we note that for every path set $P$, $L_v(P) \leq |P|$. Therefore the following corollary follows immediately.

**Corollary 6.1.** $RLP_{path}(G, d, P)$ is APX-HARD when $L_v(P) \leq 2$ and $d = 2$.

### 6.2 $RLP_{req}$ with fixed number of requests

In this section we show that $RLP_{req}$ is FPT in the number $|Q|$ of connection requests, when $d = 1$. We start by introducing two problems.

<table>
<thead>
<tr>
<th><strong>Minimum Directed Steiner Tree Problem</strong> ($DST$)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> An instance $(G, w, S, r)$ where $G = (V, E)$ is a digraph with a non-negative weight function $w : E \mapsto \mathbb{R}^+$ on its edges, $S \subseteq V$ is a set of terminal vertices, and $r \in V$ is the root vertex.</td>
</tr>
<tr>
<td><strong>Output:</strong> A subgraph $T$ of $G$ such that: (a) $T$ is a directed tree rooted at $r$, (b) $S \subseteq V(T)$.</td>
</tr>
<tr>
<td><strong>Objective:</strong> Minimize $w(T) \overset{\text{def}}{=} \sum_{e \in E(T)} w(e)$.</td>
</tr>
</tbody>
</table>

Given a graph $G = (V, E)$ and a subset $S \subseteq V$ of its vertices, a subset $D \subseteq V$ is said to dominate $S$ in $G$ if every vertex $s$ of $S$ is either in $D$ or adjacent to a vertex of $D$.

<table>
<thead>
<tr>
<th><strong>Minimum Steiner Connected Dominating Set Problem</strong> ($SCDS$)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> An instance $(G, w, S)$ where $G = (V, E)$ is a graph with a non-negative weight function $w : V \mapsto \mathbb{R}^+$ on its vertices, and $S \subseteq V$ is a subset of vertices to be dominated.</td>
</tr>
<tr>
<td><strong>Output:</strong> A set of vertices $D \subseteq V$ such that: (a) $G[D]$ is connected, (b) $D$ dominates $S$ in $G$.</td>
</tr>
<tr>
<td><strong>Objective:</strong> Minimize $w(D) \overset{\text{def}}{=} \sum_{v \in D} w(v)$.</td>
</tr>
</tbody>
</table>

The $SCDS$ problem is a generalization of the well-known problem of finding a minimum connected dominating set. It was first defined in [GK98] in the context of approximation algorithms.

We denote by $OPT_{DST}(G, w, S, r)$ (resp. $OPT_{SCDS}(G, w, S)$, $OPT_{RLP_{req}}(G, d, Q)$) the optimum value of the instance $(G, w, S, r)$ (resp. $(G, w, S)$, $(G, d, Q)$) of problem $DST$ (resp. $SCDS$, $RLP_{req}$).

In [GNS09], it is noted that $DST$ is FPT in the number of terminals. We will use this result to show that (a) $SCDS$ is FPT in the size of the dominated set, and (b) $RLP_{req}$ is FPT in the number $|Q|$ of requests when $d = 1$. 

40
Figure 6.2: (a) An undirected graph $G$. (b) The constructed directed graph $G'$

Lemma 6.1. $SCDS$ is FPT in the size of the dominated set.

Proof. By reduction to $DST$. Let $(G, w, S)$ be an instance of $SCDS$. We build a digraph $G' = (V', E')$ where

$$V' = \{v_0, v_1 \mid v \in V\} \cup \{(u_1, v_0), (v_1, u_0) \mid \{u, v\} \in E\}. \tag{6.1}$$

An example of the construction is shown in Figure 6.2. The weight function $w'$ on the edges of $E'$ is

$$w'(u_0, u_1) = w(u) \quad \forall u \in V$$
$$w'(u_1, v_0) = w'(v_1, u_0) = 0 \quad \forall \{u, v\} \in E$$

Let $S_0 \overset{\text{def}}{=} \{v_0 \mid v \in S\}$. In the sequel we prove the following claim implying an FPT algorithm for $DST$:

Claim 6.2. $OPT_{SCDS}(G, w, S) = \min_{r \in V} OPT_{DST}(G', w', S_0, r_0)$. Moreover, given a solution $T$ of $(G', w', S_0, r_0)$ a solution $D$ of $(G, w, S)$ with $w(D) = w'(T)$ can be calculated in polynomial time.

The lemma concludes easily from the claim: Indeed, let $ALGDST(G', w', S_0, r_0)$ be an FPT algorithm calculating an optimal solution in time $f(|S_0|) \cdot p(n)$. Algorithm 2 calculates an optimal solution $D^*$ of $(G, w, S)$ in time $|V| \cdot f(|S_0|) \cdot p(q_2(n)) + q_1(n) + q_3(n) = |V| \cdot f(|S|) \cdot p(q_2(n)) + q_1(n) + q_3(n) \leq f(|S|) \cdot (n \cdot p(q_2(n)) + q_1(n) + q_3(n))$, therefore is an FPT algorithm for $DST$.

We proceed with the proof of Claim 6.2. Let $D^*$ an optimal solution of $SCDS$, i.e. $w(D^*) = OPT_{SCDS}(G, w, S)$. Let $G[D^*]$ be the connected graph induced by $D^*$, and let $T_1$ be a spanning tree of $G[D^*]$. As $D^*$ dominates $S$, every $s \in S \setminus D^*$ is adjacent to a vertex $d \in D^*$. Let $T_2$ be the tree obtained from $T_1$ by adding all the vertices $s \in S \setminus D^*$ and
Algorithm 2 ALGSCDS \((G, w, S)\)

1: Build \(G', w'\) and \(S_0\) from \(G, w, S\). \(\triangleright \) running time \(q_1(n)\)
2: \(r_0 \leftarrow \arg\min_{r_0 \in V} w(ALGDST(G', w', S_0, r_0)).\) \(\triangleright \) running time \(|V| \cdot f(S_0) \cdot p_2(n)\)
3: \(T^* \leftarrow ALGDST(G', w', S_0, r_0).\)
4: Calculate \(D^*\) from \(T^*\). \(\triangleright \) running time \(q_3(n)\)
5: return \(D^*\).

connecting each one of them to arbitrarily chosen neighbor in \(D^*\). We note that the vertices of \(s \setminus D^*\) are leaves of \(T_1\), therefore all the non-leaf vertices are vertices of \(D^*\). Let \(r\) be an arbitrarily chosen non-leaf vertex of \(T_1\) \((r \in D^*\)) and \(T_3\) be the rooted tree obtained from \(T_2\) by directing its edges so that \(r\) is the root of \(T_3\). We now construct a subtree \(T'\) of \(G'\) from \(T_3\) in the following way: we replace (a) every leaf vertex \(u\) by a (leaf) vertex \(u_0\), (b) every non-leaf vertex \(u\) by two vertices \(u_0, u_1\) with an edge \((u_0, u_1)\) connecting them, (c) every edge \((u, v)\) with an edge \((u_1, v_0)\). We claim that \(w'(T') \leq w(D^*)\). Indeed, the only non-zero weighted edges of \(T'\) are the edges \((u_0, u_1)\) having weight \(w'(u_0, u_1) = w(u)\), therefore \(w'(T') = \sum_u w(u)\) where the sum is taken over all internal vertices \(u\) of \(T_3\). These vertices are also in \(D^*\), thus implying \(w'(T') \leq \sum_{u \in D^*} w(u) = w(D^*)\). Moreover the root of \(T'\) is \(r_0\) (recall that \(r\) is the root of \(T_3\)) and \(S_0 \subseteq V(T')\), therefore \(T'\) is a solution of \((G', w', S_0, r_0)\). Therefore

\[
OPT_{DST}(G', w', S_0, r_0) \leq w'(T') \leq w(D^*) = OPT_{SCDS}(G, w, S),
\]

and clearly

\[
\min_{r \in V} OPT_{DST}(G', w', S_0, r_0) \leq OPT_{SCDS}(G, w, S).
\]

Now, conversely let \(T^*\) be an optimal solution of \(DST\) such that \(w'(T') = \min_{r \in V} OPT_{DST}(G', w', S_0, r_0)\). Without loss of generality all the leaves of \(T^*\) are in \(S_0\), because otherwise such leaves can be removed from \(T^*\) without increasing its cost. Consider the tree \(T_3\) obtained from \(T^*\) by contracting all the edges \((u_0, u_1)\) into a vertex \(u\), and replacing every leaf \(s_0 \in S_0\) by a leaf \(s \in S\). Let \(T_2\) be the underlying undirected tree of \(T_3\), and let \(D\) be the set of all non-leaf vertices of \(T_2\). \(T_2[D]\) is clearly connected, thus \(G[D]\) is connected. \(D\) dominates \(S\) because any vertex in \(S \setminus D\) is a leaf of \(T_2\) thus connected to a non-leaf vertex of \(T_2\), which is in turn a vertex of \(D\). Therefore \(D\) is a solution of the instance \((G, w, S)\) of \(SCDS\). We have \(w(D) \overset{\text{def}}{=} \sum_{d \in D} w(d) = \sum_{u_0} w'(u_0, u_1)\), where the sum is taken over all non-leaf vertices \(u_0\) of \(T^*\). As the leaves of \(T^*\) are in \(S_0\) the sum covers all the edges of type \((u_0, u_1)\) of \(T^*\). We conclude

\[
OPT_{SCDS}(G, w, S) \leq w(D) = \sum_{(u_0, u_1) \in T^*} w'(u_0, u_1) = w'(T^*) = \min_{r \in V} OPT_{DST}(G', w', S_0, r_0).
\]

Finally we note that all the steps described in the proof can be performed in polynomial time. \(\square\)
Theorem 6.2. $RLP_{req}$ is FPT in the number $|\mathcal{Q}|$ of the requests when $d = 1$.

Proof. Consider an instance $(G, 1, \mathcal{Q})$ of $RLP_{req}$. Without loss of generality, we assume that $\mathcal{Q}$ does not contain edges of $G$, i.e., for every $\{s_i, t_i\} \in \mathcal{Q}$, $\{s_i, t_i\} \notin E$, because otherwise this request is $d$-satisfied in every solution routing it through this edge. We denote by $\Omega_{\mathcal{Q}}$ the set of all partitions of the request set $\mathcal{Q}$.

The following claim implies an FPT algorithm for $RLP_{req}$:

Claim 6.3. $OPT_{RLP_{req}}(G, 1, \mathcal{Q}) = \min_{\Pi \in \Omega_{\mathcal{Q}}} \sum_{\mathcal{Q}_i \in \Pi} OPT_{SCDS}(G, 1, \text{term}(\mathcal{Q}_i)).$ \footnote{1} Moreover, given optimal solutions $D_i$ of instances $(G, 1, \text{term}(\mathcal{Q}_i))$ for every $\mathcal{Q}_i \in \Pi$, a solution $\mathcal{P}, R$ of $(G, 1, \mathcal{Q})$ with $|R| = \sum w(D_i) = \sum |D_i|$ can be calculated in polynomial time.

The lemma concludes easily from the claim: Indeed, let $ALG_{SCDS}(G, 1, \text{term}(\mathcal{Q}_i))$ be an FPT algorithm calculating an optimal solution in time $f(|\text{term}(\mathcal{Q}_i)|) \cdot p(n)$. Algorithm 3 calculates an optimal solution $R^*$ of $(G, 1, \mathcal{Q})$. The dominant term of its running time is $\sum_{\Pi \in \Omega_{\mathcal{Q}}} \sum_{\mathcal{Q}_i \in \Pi} (f(|\text{term}(\mathcal{Q}_i)|) \cdot p(n)) \leq \sum_{\Pi \in \Omega_{\mathcal{Q}}} \sum_{\mathcal{Q}_i \in \Pi} (f(2|\mathcal{Q}|) \cdot p(n)) \leq |\Omega_{\mathcal{Q}}| \cdot |\mathcal{Q}| \cdot f(2|\mathcal{Q}|) \cdot p(n)$. As the first three factors depend solely on $|\mathcal{Q}|$, $RLP_{req}$ is FPT in $|\mathcal{Q}|$ when $d = 1$.

Algorithm 3 ALGRLPREQ $(G, 1, \mathcal{Q})$

1: $\Pi \leftarrow \arg\min_{\Pi \in \Omega_{\mathcal{Q}}} \sum_{\mathcal{Q}_i \in \Pi} |ALG_{SCDS}(G, 1, \text{term}(\mathcal{Q}_i))|.$ \hspace{1cm} $\triangleright$ running time

$\sum_{\Pi \in \Omega_{\mathcal{Q}}} \sum_{\mathcal{Q}_i \in \Pi} f(|\text{term}(\mathcal{Q}_i)|) \cdot p(n)$

2: for $\mathcal{Q}_i \in \Pi$ do
3: $R_i \leftarrow ALG_{SCDS}(G, 1, \text{term}(\mathcal{Q}_i))$
4: for $(s, t) \in \mathcal{Q}_i$ do
5: $s' \leftarrow$ an arbitrary neighbor of $s$ in $R_i$
6: $t' \leftarrow$ an arbitrary neighbor of $t$ in $R_i$
7: route $(s, t)$ through:
8: - the edge $\{s, s'\}$
9: - a path from $s'$ to $t'$ in $G[R_i]$
10: - the edge $\{t', t\}$
11: end for
12: end for
13: $R^* \leftarrow \cup_{\mathcal{Q}_i \in \Pi} R_i$.

We proceed with the proof of Claim 6.3. Let $(\mathcal{P}, R^*)$ be an optimal solution of $RLP_{req}$, i.e. $|R^*| = OPT_{RLP_{req}}(G, 1, \mathcal{Q})$. Let $R_1, R_2, \ldots$ be the connected components of $G[R^*]$. Consider a request $(s, t) \in \mathcal{Q}$ and its routing $P = (s, \ldots, t)$. As $d = 1$ all the vertices in $P$ contain regenerators except possibly $s$ and $t$. Therefore $P \cap R \subseteq R_i$ for some connected component $R_i$ (If this is true for more than one connected component, we choose one arbitrarily). Moreover $R_i$ dominates $\{s, t\}$, because of our assumption that $\{s, t\}$ is not an edge. We associate the request $(s, t)$ with $R_i$. Let $\mathcal{Q}_i$ be the set of requests associated with $R_i$, then $R_i$ dominates $\text{term}(\mathcal{Q}_i)$. We conclude that $R_i$ is a solution of the instance $(G, 1, \text{term}(\mathcal{Q}_i))$ of $SCDS$, thus
$OPT_{SCDS}(G, 1, term(Q_i)) \leq |R_i|$. Clearly the sets $Q_1, Q_2, \ldots$ constitute a partition $\Pi$ of $Q$. Summing up over all connected components we get

$$\sum_{Q_i \in \Pi} OPT_{SCDS}(G, 1, term(Q_i)) \leq \sum |R_i| = |R^*| = OPT_{RLP_{req}}(G, 1, Q),$$

and clearly

$$\min_{\Pi \in \Omega_Q} \sum_{Q_i \in \Pi} OPT_{SCDS}(G, 1, term(Q_i)) \leq \sum_{Q_i \in \Pi} OPT_{SCDS}(G, 1, term(Q_i)) \leq OPT_{RLP_{req}}(G, 1, Q).$$

Conversely let $\Pi \in \Omega_Q$ be a partition of $Q$ minimizing $\sum_{Q_i \in \Pi} OPT_{SCDS}(G, 1, term(Q_i))$, and for every $Q_i \in \Pi$ let $D_i$ be the connected dominating set with $|D_i| = OPT_{SCDS}(G, 1, term(Q_i))$. Consider a request $(s, t) \in Q_i \subseteq Q$. As $D_i$ dominates $Q_i$ and $R[D_i]$ is connected, $(s, t)$ can be routed as described in lines 7-10 of Algorithm 3. Therefore $R$ is a solution of the instance $(G, 1, Q)$ of $RLP_{req}$. We have

$$OPT_{RLP_{req}}(G, 1, Q) \leq |R| = \sum_{Q_i \in \Pi} |D_i| = \sum_{Q_i \in \Pi} OPT_{SCDS}(G, 1, term(Q_i))$$

$$= \min_{\Pi \in \Omega_Q} \sum_{Q_i \in \Pi} OPT_{SCDS}(G, 1, term(Q_i)).$$

Finally we note that all the steps described in the proof can be performed in polynomial time. \qed
Chapter 7

Summary and Open Problems

We have considered the role of several parameters in the $RLP_{path}$ and $RLP_{req}$ problems. For $RLP_{path}$ we considered the treewidth, the vertex load, and the number of paths as parameters. For $RLP_{req}$ we considered the treewidth and the number of requests as parameters. In each case, our goal was to determine whether: (1) the problem is fixed parameter tractable for that parameter, (2) the problem is polynomial for all or some fixed values of that parameter (but possibly not FPT in that parameter), or (3) the problem is hard for that parameter.

We have several remaining open cases:

- $RLP_{path} (G, d, \mathcal{P})$ parameterized by $tw(G)$ when $d = 3, 4$.
- $RLP_{req} (G, 1, \mathcal{Q})$ parameterized by $|\mathcal{Q}|$ when $d \geq 2$
- We have seen that $RLP_{path} (G, d, \mathcal{P})$ is polynomially solvable when $tw(G)$ and $L_v(P)$ are fixed. Is $RLP_{path}$ parameterized by $tw(G) + L_v(P)$ fixed parameter tractable? (our algorithm does not guarantee this in case $d$ is not constant)

This work can be extended in many ways, including:

- A full analysis of the above cases.
- Presenting new parameters and combinations of parameters. For example, we have considered the treewidth parameter as a structural parameter of the input graph. It may be interesting to consider related parameters like cliquewidth, pathwidth and local treewidth.
- Considering both problems, under the above-mentioned parameters, for special families of graphs; for example: planar graphs, and more generally bounded genus graphs or minor-free graphs.
- Providing approximation algorithms or proving approximation hardness for the NP-HARD cases.
• Considering parameters and special families of graphs for solving the problem of minimizing the total number of regenerators satisfying multiple sets of paths (This problem was discussed in [MSSZ12] in the context of approximation algorithms).

• Considering parameters and special families of graphs for $RLP_{\text{path}}$ and $RLP_{\text{req}}$ when an upper bound is placed on the number of regenerators permitted per vertex (This variant was discussed in [FMSM+11] in the context of approximation algorithms).
Bibliography


[YR05] Xi Yang and Byrav Ramamurthy. Sparse regeneration in translucent
wavelength-routed optical networks: Architecture, network design and wave-

[ZN98] X. Zhou and T. Nishizeki. The edge-disjoint paths problem is np-complete for
partial k-trees. In *Proceedings of the 9th International Symposium on Algo-
rithms and Computation (ISAAC)*, pages 417–426, Taejon, Korea, December
1998.
סיבוכיות של בעיית מיקום רגרנרטורים
ברשתות אופטיות -
רוחב עץ ופרמטרים נוספים

איתמר הרטשטיין

אתימר הרטשטיין
סיבוכיות של בעיית מיקום רגנרטורים
ברשתות אופטיות -
רוחב עץ ופרמטרים נוספים

חיבור על מחקר
לשם מילוי חלקי של הדרישות לקבלת התואר
מגיסטר למדעים במדעי המחשב

איתמר הרטשטיין

הוגש לסנט הטכניון – מכון טכנולוגי לישראל
מרץ 2012 חיפה אדר תש"ב

lesai מילוי חלקו של הדרישות לקבלת התואר
מגיסטר במדעים במדעי המחשב

איתמר הרטשטיין

הוגש לסנט הטכניון – מרכז סטטולוגי לישראל
מרץ 2012 חיפה אדר תש"ב
המחקר הושקע בהנחיית פרופ' שמואל זקס וד"ר מרדכי שלום מהפקולטה למדעי המחשב.

אני מודה לטכניון על התמיכה הכספית הנדרשת ומוקלטת למדעי המחשב.
迨 לשתיות אופטיות

הצורך ברשתות גדול, מהירות ואמין גדל בקצב עצום. יישומים רשתיים כגון שידור 
וידאו ושיחות ועידה ברזולוציה גבוהה, מחשוב על, חישוב מבוזר וدمיה תופעת ווריםי

ינטלקטלת שמירה על תקשורת בקצבים גבוהים במיוחד, זמן תגובה מיידי לעבר מרחקים עצומים,

ושירות לא רק שנדבר.

רשמה של כל-אופטיות מונה את ההבטחה המגיעה של הרשת המק厥ית בברך למומש רשתות במקורות

שה גניז ייבים לשנייה כלכלית. הרישום מאחריו רשתות כלולות ומספרים או את התשמית

במעשים אופטיק, לא להמוד לא להאקלטרוניות בתחום הצבתיי בים.

רשמה של מפורצות נודדות דרישה לזרב לשתיי אפשריות בשכבותיה של התמורות של כל-אופטיות

WDM – Wavelength Division Multiplexing

ונת מפורצות אחריה של (wavelength) WDM – ב. Division Multiplexing

קובר את הבشاش התנשלות בזום מברק לשתייה פזמה הקשהות בככל-זרב. רשתות

סטילס השדרים לעבר מספר בקשות תקשורת, בנべ ש שינויים בכל-בشاش התשמית מזמנת אורכי

גל לקיים.

ואז בתוכי,gles ברבע גלים הריסתי הש %= 12.5(מ"מ) עלIKE. ממודרメント ממסים יועץ לצבתי

אות-לונש (SNR – Signal To Noise Ratio)

Reconfigurable Optical Add-) ROADM

יוצר והנשה באומן מבא – י[source בשם

מסוף השלט ברור ונלמסי אוכרי של ממסים (בברך לכל עצים 4) מסיב

אופסי פנים להمهر את כל-ממסים שאור sürecin דרכו. עובר כל-אורכי של שפות

חותמן. דרוש רגרגרור אופסי על מת-לזרי כל-אורכי הtoLowerCase.
In this framework, our goal is to minimize the number of regenerators in the network, so we can provide a given set of communication requests. This function is interesting because regenerators placed at the same optical node share the same ROADM and may use additional equipment.

Complexity parameter

In this thesis, we will investigate the problem of placing regenerators from a parameteric complexity perspective. In parameteric complexity, the goal is to study the impact of different parameters of the input on the difficulty of various optimization problems.

Given an optimization problem and a parameter of the problem, we say that

- the problem is \( \text{FPT} \) if it can be solved in polynomial time

and

- \( \text{NP-Hard} \) for certain parameter values.
We consider paths in a graph that do not include the terminal vertex of the paths. During the work, we mark with \( L_v(P) \) the vertex load of a given subset of paths \( P \).

Our model consists of a graph that we study in this thesis, where we model an optical network as an unweighted graph \( G = (V, E) \), and we assume that the maximum distance \( d > 0 \) from a vertex to a path satisfies the condition that the distance between two consecutive regenerators is at most \( \pi \), and the distance between any two vertices is at most \( R \). We denote this by \( \pi \) and \( R \). We solve two problems:

1. **Path Assignment Problem:** Given a set of paths \( P \), we need to find a set of regenerators such that all paths in \( P \) are supported.
2. **Request Assignment Problem:** Given a set of communication requests (pairs of vertices), we need to find a path for each request in \( P \), and a set of regenerators such that all paths in \( P \) are supported.

Results of this thesis include four positive results and four negative results for the two problems presented above, while taking into account the following parameters of the input:

- **Width of the tree** \( |V| \).
- **Vertex load** \( L_v(P) \).
- **Maximum distance between consecutive regenerators** \( \pi \).
- **Number of paths** \( |P| \).
- **Number of requests** \( |Q| \).
הטבלה הבאה מסכמת את כלל התוצאות בעבודה:

<table>
<thead>
<tr>
<th>בעיית המסלולים</th>
<th>בעיית הבקשות</th>
</tr>
</thead>
<tbody>
<tr>
<td>פולינומית בשותרות והגרף מתגליל</td>
<td>d = 5 \text{ כazers} \quad \text{tw}(G) = 2 \text{ נפסל} \quad \text{NP-Hard}</td>
</tr>
<tr>
<td></td>
<td>d = 3 \text{ כazers} \quad \text{L}_v(P) = 4 \text{ נפסל} \quad \text{NP-Hard}</td>
</tr>
<tr>
<td></td>
<td>d = 2 \text{ כazers} \quad \text{tw}(G) = \text{כazers} \quad \text{FPT}</td>
</tr>
<tr>
<td>פולינומית בשותרות והגרף קעור</td>
<td>\text{כלל} \quad \text{כלל} \quad \text{APX-Hard}</td>
</tr>
</tbody>
</table>

**שאלות פתרונות וכיווני המשך**

המקרים הבאים נותרו פתוחים:

- הרואנו שעבור \( d = 2 \) בעיות המסלולים והן \text{FPT} ברוחב העץ. לעומת זאת \( \text{נפסל} \) \( d = 5 \text{ כazers} \quad \text{tw}(G) = 2 \text{ נפסל} \) \( ?d = 3,4 \)
- \( d = 1 \text{ כazers} \quad |Q| - \text{כazers} \quad \text{FPT} \)
- \( ?d \geq 2 \)
- \( \text{נפשה טעויות המסלולים והן פולינומית כazers} \quad \text{ולנוائز אחיו כazers} \quad \text{FPT} \) לכל \( d \).

ניתן לתרום את הטרובה במספר דרכים. למשל:

- \( \text{נפשה מצלא של המקרים הפתרונותシェינט} \)
- \( \text{נפח של פרמטור שלה יש היווי בינוויסי. למשל, חצרות את רובע הנקב וברחפ} \)
- \( \text{פרמטור לכל שאר הנקב. ליתיה מועטני לחקור פרמטורי ביניים אחריים קצינו צוגר רוחב local (cliquewidth), ורוחב מסלול (pathwidth) ורוחב עלמקומי } (\text{treewidth}) \)

Technion - Computer Science Department - M.Sc. Thesis  MSC-2012-11 - 2012
בחינת הביצועים של הבעיות הנ"ל,ductor התיאוריה לפרשנויות שטוחות יליל, עבור משפחות מינייהות של גרפים: למשל – גרפים מישוריים, ובאופי כליל לגרפים חסמי גנוס (Bounded-genus graphs) וגרפים נטולי מינור (Minor-free graphs).

בחינת הפרמטרים ומשפחות גרפים מיוחדות עבור הבעיות של מינייהות והמשפות של גרפים במינור ובגדות אלגוריתמים של כלל מינור השמד.

עלון עבורה מפריט מהגרפרים מהוגרים בכל הנכון.