Analysing flow-like problems parameterised by tree-depth

Jack de Jong, 5977843

Supervisor: Prof. dr. Hans Bodlaender
Second supervisor: Dr. Erik Jan van Leeuwen

Jan. 24 2024

Abstract

In the field of parameterised complexity, there has been a significant amount of research into the parameters treewidth and pathwidth, but not a comparable amount of research into the related *tree-depth* parameter. In this paper, we try to expand our knowledge about the hardness of a set of 'flow-like' graph problems when parameterised by tree-depth, similar to work done in [1] and [2] where the same set of problems was considered for pathwidth and treewidth respectively. We also provide hardness proofs for the class XSLP, which was defined in [3] by Bodlaender et al., and is intended to serve as a 'natural home' for problems parameterised by tree-depth, similar to how XALP and XNLP are intended as 'natural homes' for problems parameterised by treewidth and foor pathwidth respectively. Furthermore, by showing that a parameterised reduction exists between any two problems in the set of flow-like problems we consider when using the tree-depth parameter, we support a conjecture that this set of problems is in a different complexity class that is distinct from XSLP.

Contents

Chapter 1

Introduction

Parameterised complexity is the study of algorithmic complexity with regard to parameters other than the input size of a given problem. Often, we consider a problem where if a certain parameter k is fixed at small values, we will have that the time and space requirements of an algorithm to solve the problem will still be within 'tractable' limits. More formally, the class *FPT* contains those problems which can be solved in $f(k) \cdot n^c$ time for a given input with size n, parameter k, a computable f, and c a constant. Other well-known complexity classes are *XP*, which contains the 'slicewise polynomial' problems which can be solved in $n^{f(k)}$ time, and the complexity classes of the type $W[i]$ for natural i in the so-called *W hierarchy*.

Together, the complexity classes mentioned above are intended to differentiate problems according their complexity to a finer degree; by providing completeness proofs for various classes, we get a better intuition of their inherent 'hardness', as well as the likelihood of finding algorithms that are faster than what is currently available.

More recently, classes such as *XNLP* [4], [5] and *XALP* [2] have been defined. The class XNLP contains all problems which can be solved nondeterministically in time $f(k)n^c$ and space $f(k)\log n$ for a parameter k. The class XALP is similarly defined as containing the problems which can be solved in $f(k)n^c$ time and $f(k)\log n$ space, but with a non-deterministic Turing machine which has access to an auxiliary stack. Less formally, the classes XALP and XNLP can be seen as 'natural' homes for problems parameterised by the treewidth and pathwidth parameters respectively, as demonstrated by the great number of completeness proofs for problems using either parameter ([6], [1], [4], [2]).

In addition to XNLP and XALP, a third such class named *XSLP* has been defined in [3]. This complexity class is defined around the fundamental *Binary Constraints* graph problem, referred to as BinCSP. Similar to XALP and XNLP, the authors of [3] have tried to create a 'natural home' for graph problems which are parameterised by tree-depth. Since this complexity class was only recently defined, many open questions remain as to which existing graph problems can be shown to be complete for it, as well as its relationship to other pre-established complexity classes. Additionally, there has not been much research done into the hardness of problems parameterised by the tree-depth parameter relative to the parameters treewidth and pathwidth, with which it shares similarities.

In this paper, we provide some of the first hardness and membership proofs for XSLP. We also consider a class of 'flow-like' problems which were shown to be XALP-complete as well as XNLP-complete in [2] and [1] for the treewidth and pathwidth parameter respectively, and expand the class with a few more similar flow-like problems. Initially, it was expected that these problems would be suitable candidates for demonstrating XSLP-hardness and/or membership, but difficulty in providing these proofs gave rise to the conjecture that they are in fact contained in a complexity class that is distinct from the XSLP class, casting doubt on the notion that XSLP could be a natural home for problems parameterised by tree-depth in a manner analogous to XALP and XNLP.

Chapter 2

Preliminaries

Before we continue, we provide some definitions of terms and concepts that will be used in the sections that follow, although we will assume familiarity on the part of the reader with the basic concepts of algorithmic complexity.

2.1 Tree-depth

Given a graph $G = (V, E)$, the tree-depth of G is the minimum height of a rooted forest F which has the same vertex set as G, and which has the property that for any pair $(v, w) \in E$ we have that v and w have an ancestor-descendant relationship in F; that is, adjacent vertices in G do not have do be adjacent in F, but they cannot be on different branches. Note that, since F is always a connected tree if G is connected, we will usually denote it as T instead. In the sequel, we will call F (or T) the *tree-depth spanning forest* of G, or the *tree-depth spanning tree* of G if G is connected. Alternatively, a forest or tree with such properties is referred to as a *Trémaux tree*, named for the 19th century mathematician credited with early research into depth-first search.

Before the term tree-depth was introduced [7], it was alternatively known as the *vertex rank number* (introduced in [8]), the *ordered chromatic number*, or the minimum height of an *elimination tree* (e.g. in [3]) of a given graph (these properties are defined differently, but their equivalence is proven in [7]). However, for the purposes of understanding this paper, it suffices to understand the first definition that we have outlined here.

2.2 Treewidth

Given a graph $G = (V, E)$, a *tree decomposition* is defined as a tree W, whose vertices are subsets $X_1, ..., X_k$ of the vertices in V (referred to commonly as 'bags'), such that the following three properties hold:

1. The union of all sets X_i for $i \in [1, k]$ is V;
2. For every pair X_i, X_j with $i, j \in [1, k]$, if X_i and X_j both contain a vertex $v \in V$, then every bag on the path between X_i and X_j in W contains v as well;
3. If $\{u, v\} \in E$, then at least one set X_i contains both u and v.

We call the maximum cardinality of any bag of W, minus one, the *width* of W, and we call the minimum width of any tree decomposition of G the *treewidth* of G.

2.3 Pathwidth

Given a graph $G = (V, E)$, we define a *path decomposition* of G as a sequence of subsets $X_1, ..., X_k$ of V such that:

1. The union of all sets X_i for $i \in [1, k]$ is V;

2. If $\{u, v\} \in E$, then at least one set X_i contains both u and v;
3. For all $i, i', j \in [1, k]$ where $i \leq i' \leq j$ we have that $X_i \cap X_j \subseteq X_{i'}$.

Like with treewidth, the width of a path decomposition is the maximum cardinality of any bag X_i minus one. The pathwidth of G is then the minimum width across all path decomposition of G.

2.4 Binary Constraints Satisfaction Problem

For this paper, one of the central problems will be the Binary Constraints Satisfaction Problem (BinCSP). An instance of the Binary Constraints Satisfaction Problem consists of an undirected graph $G = (V, E)$ called the *primal* or the *Gaifman graph*, where each $v \in V$ represents a variable whose value must be assigned, and where each adjacent pair $v, w \in V$ has associated with it a list $C(v, w)$ of possible value combinations for v and w respectively. The problem is solved by deciding whether an assignment of values to every vertex $v \in V$ exists that satisfies all edge constraints.

Formally, we define:

Binary Constraints Satisfaction Problem (BinCSP)
Input: An undirected graph $G = (V, E)$, a set of values D, and for each adjacent pair $v, w \in V$ a set of constraints $C(v, w) \subseteq \{(c, c') : c, c' \in D\}$.
Question: Is there a value assignment $\gamma : V \to D$ such that for adjacent pair $u, v \in V$ we have that $(\gamma(u), \gamma(v)) \in C(u, v)$?

In the sequel, we will assume that, for any instance of BinCSP and for any edge $\{v, w\} \in E$, the constraint sets $C(v, w)$ and $C(w, v)$ are symmetrical, in the sense that $C(w, v) = \{(c', c) : (c, c') \in C(v, w)\}$. In an instance of BinCSP where this is not the case, we could easily obtain an equivalent instance of BinCSP where we define a new constraint set $C'(v, w)$ whose value pairs abide by both $C(v, w)$ and $C(w, v)$, and subsequently define $C'(w, v) := \{(c', c) : (c, c) \in C'(v, w)\}$.

2.5 pl-reductions

Given two decision problems $Q_1 \subseteq \Sigma_1^* \times \mathbb{N}$ and $Q_2 \subseteq \Sigma_2^* \times \mathbb{N}$, we say that a function $f : \Sigma_1^* \times \mathbb{N} \to \Sigma_2^* \times \mathbb{N}$ is a *reduction* from Q_1 to Q_2 if for any $(x, k) \in \Sigma_1^* \times \mathbb{N}$ we have that $(x, k) \in Q_1$ if and only if $(x', k') = f((x, k)) \in Q_2$; that is, for any problem instance of Q_1, the function f maps it to an equivalent problem in Q_2. Furthermore, we say that f is a *parameterised reduction* if a computable function g exists such that $k' \leq g(k)$ for any such (x, k).

Lastly, if the above function f has an associated algorithm that computes $f(x, k)$ in logarithmic space with regard to the input x (i.e., in $\mathcal{O}(h(k) + log(|x|))$ space with h a computable function), we say that f is a *parameterised logspace* reduction, or a *pl-reduction* for short.

Note that a pl-reduction is a weaker reduction than an *fpt-reduction*, in the sense that the existence of a pl-reduction given two parameterised problems implies that there exists an fpt-reduction as well. Here, an fpt-reduction is a parameterised reduction that can be executed in $h(k)|x|^{\mathcal{O}(1)}$ for a computable function h. In this paper, we refer to the reductions that are used as pl-reductions exclusively, as that matches the definition of XSLP in [3], as well as the definitions of XALP and XNLP.

2.6 Definition of XSLP

All of the results in this paper relate, in some way, to a complexity class of decision problems named XSLP, which is defined in [3] as the set of decision problems which can be pl-reduced to the Binary Constraints Satisfaction Problem when parameterised by the tree-depth of the associated Gaifman graph.

Apart from this definition, two separate characterisations of XSLP are outlined in [3], and equivalence is proven between the following statements:

1. A parameterised problem Q is in XSLP.

2. Q can be decided by an alternating read-once stack machine that for input (x,k) with $|x| = n$, uses working space at most $f(k) + O(\log n)$, stack size $f(k)\log n$, nondeterminism $f(k)\log n$, co-nondeterminism $f(k) + O(\log n)$, and alternation $f(k)$, for some computable function f.

3. Q can be pl-reduced to the problem of model-checking $\forall$-guided sentence son forest-shaped Σ-structures.

For further explanation of these characterisations of XSLP, and for proof of their equivalence, see [3].

2.7 List Colouring and Precolouring Extension

Aside from BinCSP, there are the List Colouring (LC) and Precolouring Extension (PE) problems, which are each closely related to BinCSP.

List Colouring (LC)
Input: An undirected graph $G = (V, E)$, a set of colours $\mathcal{C}$, and for each $u \in V$ a domain list $L(v) \subseteq C$.
Question: Is there a colouring $\gamma : V \to \mathcal{C}$ such that for each $\{u, v\} \in E$ we have that $\gamma(u) \in L(u)$ and $\gamma(v) \in L(v)$, while also that $\gamma(u) \neq \gamma(v)$?

Precolouring Extension (PE)
Input: An undirected graph $G = (V, E)$, a set of colours $\mathcal{C}$, as well as a subset $S \subseteq V$ with a precolouring function $p : S \to \mathcal{C}$.
Question: Is there a colouring $\gamma : V \to \mathcal{C}$ such that for each $u \in S$ we have that $\gamma(u) = p(u)$, and also that for each $\{u, v\} \in E$ we have that $\gamma(u) \neq \gamma(v)$

With both problems defined, we can prove their XSLP completeness for tree-depth by using well known reduction from BinCSP to LC, as well a reduction from LC to PC as outlined in [4].

Theorem 2.7.1. *Precolouring Extension and List Colouring are XSLP-complete when parameterised by tree-depth.*

Proof. Observe first that any instance of List Colouring (i.e. a graph $G = (V, E)$ and a set of colour lists $\{L(v)|v \in V\}$) is essentially a special case of BinCSP, where for every $\{u, v\} \in E$ we have that the set of constraints $C(u, v)$ is exactly equal to $\{(c, c')|c \in L(u), c' \in L(v), c \neq c'\}$. Similarly, we have that every instance of Precolouring Extension (i.e. a graph $G = (V, E)$, colour set $\mathcal{C}$ and precoloured subset $S \subseteq V$) is a special case of List Colouring, where for every $u \in V \setminus S$ we have that $L(u) = \mathcal{C}$, and for every $v \in S$ we have that $L(v) = p(v)$. Membership in XSLP for both List Colouring and Precolouring Extension follow trivially as a result.

To prove XSLP hardness for both problems, we will start with an instance of BinCSP parameterised by tree-depth, i.e. a graph $G = (V, E)$, a set of values D, and for adjacent pair $v, w \in V$ a constraints set $C(v, w)$, as well as a tree-depth spanning tree T of G with height td, and we will show how to reduce this problem instance to an instance of List Colouring, and subsequently to an instance of Precolouring Extension.

To reduce to List Colouring, we create a new graph $G' = (V', E')$ which contains every vertex in G, but in addition to that, for every edge $\{v, w\} \in E$ and for every tuple (c, c') which is *not* contained in $C(v, w)$, we create a new vertex $x_{v,w,c,c'}$ which we connect to both v and to w with an edge in E'. For every $v \in V$, we then let $L(v) = D$, and for every vertex $x_{v,w,c,c'}$ we let $L(x_{v,w,c,c'}) = \{c, c'\}$. It is easy to see that an assignment of values of G that abides by its binary constraints leads to a valid list colouring of G', and vice versa. Additionally, since every vertex of the type $x_{v,w,c,c'}$ is only adjacent to v and w, and since v and w are adjacent in G and therefore an ancestor-descendant pair in T, we can expand T into a tree-depth spanning tree T' of G' by connecting $x_{v,w,c,c'}$ to whichever vertex between v and w is a descendant of the other. Doing so will increase the height of the resulting tree to at most $td + 1$.

To subsequently reduce to Precolouring Extension, we let $\mathcal{C} = \bigcup_{v \in V} D(v)$, and we create a graph $G'' = (V'', E'')$ which contains every vertex and every edge in G'. In addition, for every $v \in V'$, and for every $c \in D \setminus L(v)$, we create a vertex $y_{v,c}$ which we precolour with c and which we connect to v via an edge. It is easy to see how the resulting instance of Precolouring Extension is equivalent to the instance of List Colouring, since the precoloured neighbours of v 'prohibit' v from selecting a colour that is outside of its domain. Additionally, since every vertex of the type $y_{v,c}$ is only adjacent to one vertex v in the original graph G', we can expand T' into a tree-depth spanning tree of G'' by simply connecting every vertex of the type $y_{v,c}$ below v in T'. The height of this tree is at most one greater than the height of T', and therefore the tree-depth of G'' is at most $td + 2$.

Having proven XSLP membership and hardness for both problems when parameterised by tree-depth, we conclude that they are both XSLP-complete when parameterised by tree-depth. $\square$

2.8 Flow problems

Although we assume familiarity on the part of the reader with typical flow problems, we will restate the definition of an integer flow problem here.

Given a directed graph $G = (V, E)$ with vertices $s, t \in V$, as well as a capacity function $c : E \to \mathbb{Z}_{>0}$, we define a *flow* on G as a function $f : E \to \mathbb{Z}_{\geq 0}$ that maps a flow value to every arc in E. Furthermore, we say that f is a *valid* or *feasible* flow on G given c, if, for every $e \in E$, we have that $f(e) \leq c(e)$, and, for every $v \in V \setminus \{s, t\}$, we have that the property of *flow conservation* holds, which means that the total flow into v is equal to the total flow out of v. Lastly, we define $|f|$ as the total flow that goes out of the source vertex s, and we call $|f|$ the *flow value* or simply *value* of f.

Note that this definition is different from the more 'classic' definition of flow problems as outlined for instance by Cormen et al. in [9], where, if the flow value from a vertex v to w is equal to $f(v, w)$, there is assumed to be an opposite flow of $-f(v, w)$ from w to v. In this definition, a feasible flow must have the property that for every vertex v that isn't a source or sink, the sum of all flow values moving in and out of v must be exactly 0.

2.9 Flow-like problem definitions

In addition to BinCSP and its related problems, we will consider a class of 'flow-like' graph problems which have a strong adjacency to one another via the tree-depth parameter. Most of the problems here, with the exception of Target Set Selection and f-Dominating Set, were proven complete for XNLP and XALP when parameterised by the pathwidth and treewidth parameters respectively [1], [2]. In this paper, we will show that most of the reductions used in [1] can be adapted to work for the tree-depth parameter as well, as well as introduce several novel reductions that create a strongly connected reduction graph between the problems when parameterised by tree-depth.

The first problem we define is All-or-Nothing Flow (AoNF), which will serve as the 'backbone' of many of our reductions, as it is used to reduce to and from many other flow-like problems.

All-or-Nothing Flow (AoNF)
Input: A directed graph $G = (V, E)$ with vertices $s, t \in V$, and with for each $e \in E$ a positive integer capacity $c(e) \in \mathbb{Z}_{>0}$, as well as a positive integer R.
Question: Does a flow f from s to t of value R exist such that for each $e \in E$ we have that $f(e) = 0$ or $f(e) = c(e)$?

In addition to AoNF, we consider five separate (but closely related) orientation problems for undirected graphs. For each of these problems, we define an *orientation* of an undirected graph $G = (V, E)$ as an assignment of direction for every edge in E.

Target Outdegree Orientation (TOO)
Input: An undirected graph $G = (V, E)$ with a weight function $w : E \to \mathbb{Z}_{>0}$, and for each $v \in V$ a non-negative integer d_v.
Question: Is there an orientation of G such that for each $v \in V$, the total weight of all edges directed outwards from v is equal to d_v?

Chosen Maximum Outdegree (CMO)
Input: An undirected graph $G = (V, E)$ with a weight function $w : E \to \mathbb{Z}_{>0}$, and for each vertex a non-negative integer m_v.
Question: Is there an orientation of G such that for each $v \in V$, the total weight of all edges directed outwards from v is at most m_v?

Minimum Maximum Outdegree (MMO)

Input: An undirected graph $G = (V, E)$ with a weight function $w : E \to \mathbb{Z}_{>0}$, as well as a non-negative integer m.

Question: Is there an orientation of G such that for each $v \in V$, the total weight of all edges directed outwards from v is at most m?

Outdegree Restricted Orientation (ORO)

Input: An undirected graph $G = (V, E)$ with a weight function $w : E \to \mathbb{Z}_{>0}$, and for each $v \in V$ an interval $D_v \subseteq \mathbb{Z}_{\geq 0}$.

Question: Is there an orientation of G such that for each $v \in V$, the total weight of all edges directed out of v is contained in D_v?

Circulating Orientation (CO)

Input: An undirected graph $G = (V, E)$ with a weight function $w : E \to \mathbb{Z}_{>0}$.

Question: Is there an orientation of G such that for each $v \in V$, the total weight of all edges directed out of v is equal to the total weight of all edges directed towards v.

We also consider two problems involving capacitated dominating sets, whose completeness for XNLP was proven in [6] when using pathwidth as the parameter, building on the results of [1].

Red-Blue Capacitated Dominating Set (RBCDS)

Input: An undirected bipartite graph $G = (R, B, E)$, as well as a capacity function $c : R \to \mathbb{N}$ and an integer k.

Question: Is there a subset $S \subseteq R$ with $|S| \leq k$, as well as a mapping $f : B \to S$ such that $\{u, f(u)\} \in E$ for all $u \in B$ and $|f^{-1}(v)| \leq c(v)$ for all $v \in S$?

Capacitated Dominating Set (CDS)

Input: An undirected graph $G = (V, E)$, as well as a capacity function $c : V \to \mathbb{N}$ and an integer k.

Question: Is there a subset $S \subseteq V$ with $|S| \leq k$, combined with a mapping $f : V \setminus S \to S$ such that for all $u \in S$ we have that $\{u, f(u)\} \in E$ and for all $v \in S$ we have that $|f^{-1}(v)| \leq c(v)$?

The concept of introducing a capacitated variant to a well-known graph problem also lends itself to Vertex Cover, which was first considered in TODO.

Capacitated Vertex Cover (CVS)

Input: An undirected graph $G = (V, E)$, as well as a capacity function $c : V \to \mathbb{N}$ and an integer k.

Question: Is there a subset $S \subseteq V$ with $|S| = k$, as well as a mapping $f : E \to S$ such that for all $e = \{u, v\} \in E$ we have that $f(e) = u$ or $f(e) = v$, and also that for all $v \in V$ we have that $|f^{-1}(v)| \leq c(v)$?

We consider another variant of a classical flow problem, namely Undirected Flow with Lower Bounds.

Undirected Flow with Lower Bounds (UFLB)

Input: An undirected graph $G = (V, E)$ with a source s and a sink t, as well as a positive integer R, and with with for each $e \in E$ a positive integer capacity $c(e) \in \mathbb{Z}_{>0}$ as well as a non-negative integer lower bound $l(e) \in \mathbb{Z}_{\geq 0}$

Question: Is there an orientation of G such that the resulting directed graph D allows for an s-t flow f with value R, and which abides by the capacities and lower bounds of each arc (i.e., $l(a) \leq f(a) \leq c(a)$ for every arc in D).

We also consider the Target Set Selection problem, which is a problem that has similarities to both flow and domination problems.

Target Set Selection (TSS

Input: An undirected graph $G = (V, E)$ with for each $v \in V$ a positive integer threshold $\tau(v)$, a positive integer k.

Question: Does a subset $S_0 \subseteq V$ exist with $|S_0| = k$ exist such that either $S_0 = |V|$, or in the case that $k < |V|$, we have that a disjoint partition $S_0, S_1, ..., S_t$ of V exists for some positive integer t where for every $i \in [1, t]$ we have that S_i contains exactly every vertex $v \in V \setminus S_0 \cup ... \cup S_i - 1$ which has at least a $\tau(v)$ number of neighbours in $S_0 \cup ... \cup S_{i-1}$.

A less formal but more intuitive description of the TSS problem goes as follows: given a graph with associated threshold values $\tau(v)$ for every vertex $v \in V$, we say that v is 'activated' when its number of activated neighbours is equal to or greater than $\tau(v)$. The problem is then to decide whether it is possible to activate an initial k vertices, such that every vertex in the graph is eventually activated in a finite number of steps.

Lastly we consider another variant of the dominating set problem, which is similar to Target Set Selection.

f-Dominating Set (f-Dom)

Input: An undirected graph $G = (V, E)$ with a function $f : V \to \mathbb{Z}_{>0}$, as well as a positive integer k.

Question: Is there a $D \subseteq V$ such that $|D| = k$, and each $u \in V \setminus D$ has at least an $f(u)$ number of neighbours in D?

2.10 Simple graphs vs. Multigraphs

All the problems defined in Section 2.9 are defined for simple graphs, i.e. graphs that have at most one edge between any given between every pair of vertices. However, in the sequel when providing reductions to the All-or-Nothing Flow problem parameterised by tree-depth, we will allow All-or-Nothing Flow to be defined on multigraphs as well for ease of use, i.e. graphs that can have multiple arcs between the same pair of vertices. The reason that this does not interfere with our results is that any instance of a flow problem defined on a directed multigraph $G = (V, A)$ can easily be reduced to an equivalent problem instance that is instead defined on a simple graph G' with bounded tree-depth relative to G. Figure 3.1 shows how to construct G' by, for every adjacent pair $u, v \in V$, subdividing each arc between u and v into two arcs with a connecting vertex in between, and which have the same capacities as the arc from which they were created. Clearly, this results in a flow network that is equivalent, in the sense that one flow network has a permissible flow with value R if and only if the other network does as well. Note that, for any new vertex x that was added to G to create G', we have that x is only a adjacent to an adjacent pair $u, v \in V$. We can therefore include every such vertex x in the tree-depth spanning tree T of G to obtain a tree-depth spanning tree of G', whose height is at most only one greater than that of T. Via a similar argument, one can also show that G' has bounded treewidth and pathwidth relative to G.

Chapter 3

Overview of the results

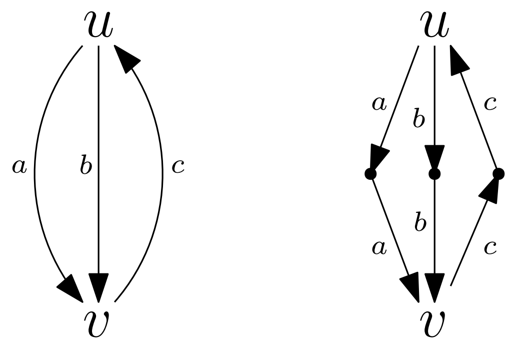

Figure 3.1: Example of an adjacent pair u, v in a multigraph with three directed arcs between them, and respective capacities a, b, and c under a given flow problem. On the right, an equivalent simple graph is constructed by subdividing each arc with a connecting vertex in between, and retaining the same capacities for every new pair of subsequent arcs.

$$\begin{tikzcd}
\text{AoNF} \arrow[r, "\text{p.13}"'] & \text{TOO} \arrow[r, "\text{p.15}"] \arrow[d, "\text{p.16}"] & \text{CO} \arrow[r, dotted] \arrow[dr, "\text{XNLP}"] & \text{UFLB} \\
\text{ORO} & \text{CMO} \arrow[d, "\text{p.15}"] & & \text{RBCDS} \arrow[d, "\text{XNLP}"] \\
& \text{MMO} \arrow[ul, dotted] & & \text{CDS}
\end{tikzcd}$$

Figure 3.2: Pre-existing reduction graph for flow-like problems when parameterised by treewidth, pathwidth, and tree-depth. A dotted arc indicates that the preceding problem is a special instance of the succeeding problem.

For this thesis project, the initial goal was to obtain hardness and/or membership results for existing problems relative to the XSLP class. Several such results were accomplished, with hardness results following for Independent Set and Dominating Set and a few related problems when parameterised by logarithmic tree-depth, as seen in Section 4.

The main body of work for this thesis project, however, is the analysis of the flow-related problems which were defined in Section 2.9. The initial motivation for studying these problems was to find an 'entry point' for XSLP-hardness, by providing a tree-depth bounded reduction from BinCSP or its related problems to one of the flow-related problems, in the hopes that such a result would lead to XSLP-hardness for the entire class of flow-like problems, similar to how XNLP-completeness was proven for the entire class in [1] when using the pathwidth parameter, and for the treewidth parameter

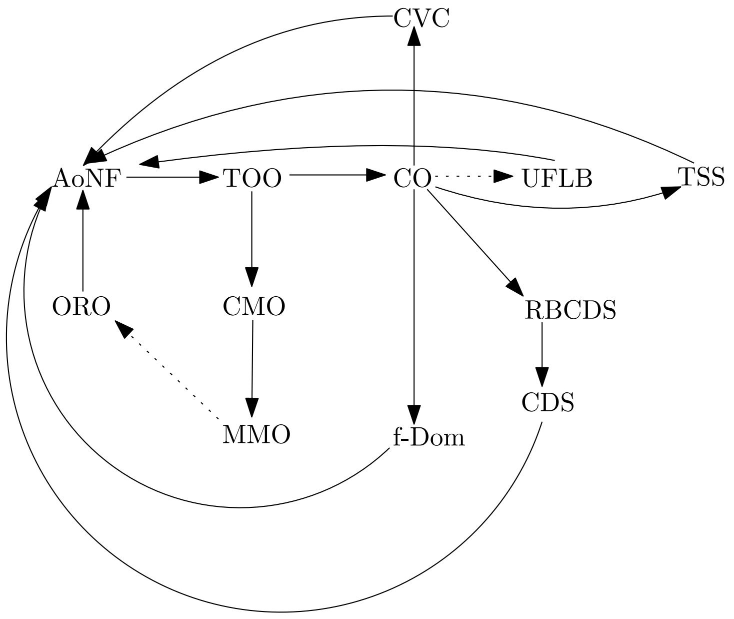

Figure 3.3: Reduction graph for flow-like problems when parameterised by tree-depth (and by treewidth and by pathwidth).

in [2]. Despite significant effort, however, no tree-depth bounded reduction was found to prove XSLP-hardness or membership for any of the flow-related problems, as there appears to be a fundamental barrier between the problems of assigning colours and of assigning flow to a given graph G when the tree-depth parameter is used. More specifically, there appears to be a fundamental difference between assigning a colour to every vertex and checking every edge constraint, and assigning a value to every edge of a graph such that every vertex abides by constraints that are analogous to flow equilibrium.

This intuition gave rise to the conjecture that the class of flow-like problems, when parameterised by tree-depth, are in a separate complexity class from XSLP. This conjecture is supported by the fact that the reduction graph for the flow-related problems parameterised by tree-depth can be shown to be strongly connected, as shown in Figure 3.3; that is, for every pair of flow-related problems parameterised by tree-depth, there exists a series of pl-reductions between the two. Section 5 contains the proofs of these reductions, and Section 6.1 contains a formal statement of this conjecture.

Chapter 4

Problems parameterised by logarithmic tree-depth

Given a graph problem parameterised by tree-depth where $G = (V, E)$ is the associated graph and td is the tree-depth of G, we call $td/\log|V|$ the *logarithmic tree-depth* of G. Similarly, if tw and pw are the treewidth and pathwidth of G respectively, then $tw/\log|V|$ and $tw/\log|V|$ represent the logarithmic treewidth and logarithmic pathwidth of G respectively.

In this chapter we will consider the Independent Set and Dominating Set problems when parameterised by logarithmic tree-depth. The reason that these problem variants are interesting is that these problems are FPT for tree-depth, and, by the definition of FPT, we therefore have that they are in XP for the logarithmic tree-depth parameter.

In [4] it was shown that Independent Set and Dominating Set are both complete for XNLP when parameterised by pathwidth. Similarly, [2] shows XALP-completeness for both problems when parameterised by treewidth. In [3], the end-of-paper discussion mentions the possibility of proving completeness for these two problems for XSLP under the logarithmic tree-depth parameter as well. Indeed, in the preliminary research for this project, it was found that hardness proofs for XSLP exists for both Independent Set and Dominating Set when parameterised by tree-depth. For these proofs, inspiration for the use of logarithmic-size 'selection gadgets' was taken from the proof of Independent Set's completeness for XNLP for logarithmic pathwidth in [4], which in turn used a gadget devised in [10].

4.1 Independent Set

Theorem 4.1.1. *Independent Set parameterised by logarithmic tree-depth is XSLP-hard.*

Proof. To demonstrate XSLP-hardness of Independent Set parameterised by tree-depth, we will provide a pl-reduction from the Precolouring Extension problem parameterised by tree-depth. To that end, we start with an instance of Precolouring Extension, i.e. an undirected graph $G = (V, E)$, a set $\mathcal{C}$ of colours, as well as a precolouring $p : S \to \mathcal{C}$ for a subset $S \subseteq V$, and a tree-depth spanning tree T of G with height td. We assume that no neighbouring two vertices are precoloured with different colours, since that would render the problem trivially unsolvable. We can also assume that $|\mathcal{C}| < n = |V|$, since every vertex has at most $n - 1$ neighbours, and so having access to n different colours would allow any vertex to always pick at least one colour, regardless of the colours picked by its neighbours, which in turn would make the problem trivial. Finally, we assume that $|\mathcal{C}|$ is exactly equal to 2^l for some integer l, since, if this is not the case, we can easily amend it by adding 'duplicate' colours to $\mathcal{C}$ until we have that $|\mathcal{C}| = 2^l$. Doing so only increases the cardinality of $\mathcal{C}$ by at most a factor 2, and so we still have that $|\mathcal{C}| < 2n$, and, given that $|\mathcal{C}| = 2^l$ for an integer l, we can use l binary bits to assign every colour $c \in \mathcal{C}$ its own unique binary bit string s_c.

We create a graph $G' = (V', E')$, on which to define an instance of Independent Set. Intuitively, we want to construct G' such that a sufficiently large independent set demands that for every $v \in V$ exactly one colour is picked from $\mathcal{C}$ via its associated bit string. To that end, we create for every $v \in V$ exactly $2l$ vertices $x_{v,1}, \bar{x}_{v,1}, ..., x_{v,l}, \bar{x}_{v,l}$, and for each pair $x_{v,i}, \bar{x}_{v,i}$ with $i \in [1, l]$ we add the edge $\{x_{v,i}, \bar{x}_{v,i}\}$ to E'. Figure 4.1 shows how this gadget looks for a given $v \in V$. Intuitively, since each pair $x_{v,i}, \bar{x}_{v,i}$ can contribute at most one vertex to an independent set on G', it follows that if we demand an independent set of size $|V| \cdot l$ on G', and we understand $x_{v,i}$ and $\bar{x}_{v,i}$ to represent the i-th bit with values 1 and 0 respectively, we will have that every vertex gadget 'selects' exactly one colour from the domain $\mathcal{C}$. Figure 4.2 shows an example of this. Note, however, that a certain subset of vertices have been precoloured. To that end, we consider every $w \in S$, and remove every bit vertex that does not correspond to the bit string associated with the colour $p(w)$; more formally, if $s_{p(w)} = b_1, ..., b_l$ is the bit string associated with $p(w)$, then for every $j \in [1, l]$ we remove from G' the vertex $x_{v,j}$ if $b_j = 0$, and we remove $\bar{x}_{v,j}$ if $b_j = 1$. Naturally, we remove the edge between the two vertices regardless of which vertex is removed.

Now let $\{v, w\}$ be an edge in E, and let $c, c' \in \mathcal{C}$ be two (possibly distinct) elements of $\mathcal{C}$ that indicate the same colour. To ensure that the gadgets of v and w do not pick c and c' respectively, we add a gadget $\Delta(v, w, c, c')$ to G which consists of $2l$ vertices that together form a clique. Then, for the bit strings s_c and $s_{c'}$, we have $2l$ of the type $x_{v,i}$, $\bar{x}_{v,i'}$, $x_{w,j}$, and $\bar{x}_{w,j'}$ which correspond to v and w selecting c and c' respectively. To complete our construction of the edge gadget, we connect every such corresponding bit vertex to a unique vertex in $\Delta(v, w, c, c')$ via an edge. Figure 4.3 shows an example of such an edge gadget, and Figures 4.4, 4.5, and 4.6 show an example of how such a gadget works in practice.

Lemma 4.1.2. *A colouring on G with a precolouring p exists if and only if G' has an independent set of size $|V|l + |E|R$, where R is the number of ordered same-colour pairs in $\mathcal{C}$.*

Proof. First we assume that a valid colouring exists on G; i.e., we have a colouring $\gamma : V \to \mathcal{C}$ such that $\gamma(u) = p(u)$ for every $u \in S$, and such that $\gamma(v) \neq \gamma(w)$ for all $\{v, w\} \in E$. Then, for every $u \in V$, we can 'pick' from every colour selection gadget in G' exactly l vertices for the independent set that correspond to the colour $\gamma(u)$. Additionally, for every $\{v, w\} \in E$, and for every same-colour pair $c, c' \in \mathcal{C}$, we know that at least one of the vertices v and w did not pick either one of the colours c and c', which means that among the vertices in $\Delta(v, w, c, c')$, there is at least one vertex y which does not neighbour any of the $2l$ vertices selected by the selection gadgets of v and w, and so we can add y to our independent set. Once we have iterated over every edge and every same-colour pairing in $\mathcal{C}$, the resulting independent set has a cardinality of exactly $|V|l + |E|R$.

To prove the lemma in the opposite direction, we can assume that an independent set I exists on G' with $|I| = |V|l + |E|R$. Then, since every colour selection gadget can contribute at most l vertices to I, and since every $\Delta(v, w, c, c')$ can contribute at most one vertex to I, and since together these gadgets comprise the entire graph G', we know that every one of these gadgets contributes its maximum amount. This means that every colour selection gadget for a given $u \in V$ 'points' to a single colour c in $\mathcal{C}$ via its accompanying bit string, and we can set $\gamma(u) = c$, knowing that $c = p(u)$ in the case that $u \in S$. Additionally, since for every $\{v, w\} \in E$ and for every same-colour pair $c, c' \in \mathcal{C}$ we have that $\Delta(v, w, c, c')$ is able to contribute one vertex, we know that the combined bit strings associated with $\gamma(v)$ and $\gamma(w)$ are different by at least one bit from the bit strings associated with c and c', meaning that $\gamma(v)$ cannot be the same colour as $\gamma(w)$. We conclude that γ is a valid colouring of G given the precolouring function p. □

By proving Lemma 4.1.2 we have shown that the problem instance of Independent Set we have constructed is indeed a valid reduction from the given problem instance of List Colouring. Furthermore, the construction can be performed in logarithmic space, as the graph G' contains a polynomial number of vertices and edges relative to the original graph G. What remains to be shown is that the logarithmic tree-depth of G' is within a constant factor of the tree-depth of G.

We can construct a tree-depth representation T' of G' by copying the structure of T in the following manner: for every $u \in V$, we sort the $2l$ vertices in its associated colour selection gadget in G' into a path p_u in arbitrary order. Then, for any pair $v, w \in V$ where v is a parent of w in T, we connect the end of v's path p_v to the root of p_w. Similarly, for every inequality gadget $\Delta(v, w, c, c')$, we can sort its $2l$ vertices into an arbitrary path, and connect its root to the end of p_w.

It is easy to verify that the resulting tree T' is a valid tree-depth representation of G'. Furthermore, since T' has a height of at most $2l \cdot td + 2l$, where $l = \log_2 |\mathcal{C}| = \mathcal{O}(\log |V|)$, we conclude that we have a valid pl-reduction from Precolouring Extension parameterised by tree-depth to Independent Set parameterised by logarithmic tree-depth. □

From the simple observation that a graph $G = (V, E)$ has an independent set of size k if and only if it has a vertex cover of size $|V| - k$, we can also conclude the following corollary result:

Corollary 4.1.2.1. *Vertex Cover is XSLP-hard when parameterised by tree-depth.*

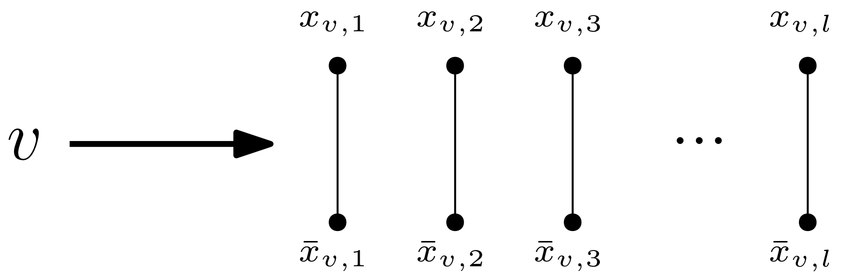

Figure 4.1: Example of a colour selection gadget for a vertex v in the reduction from Precolouring Extension to Independent Set.

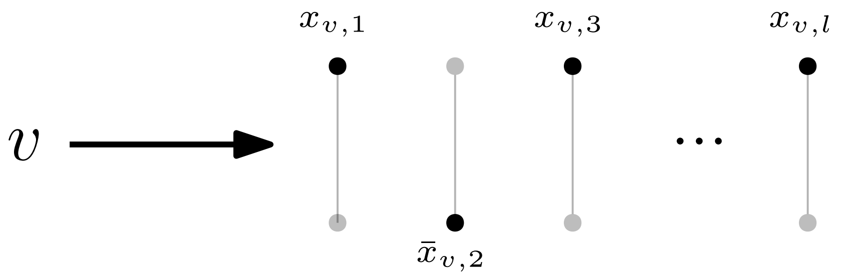

Figure 4.2: Example of a colour selection gadget for a vertex v in the reduction from Precolouring Extension to Independent Set, where the l vertices picked for the dominating set correspond to exactly one colour which can be applied to v.

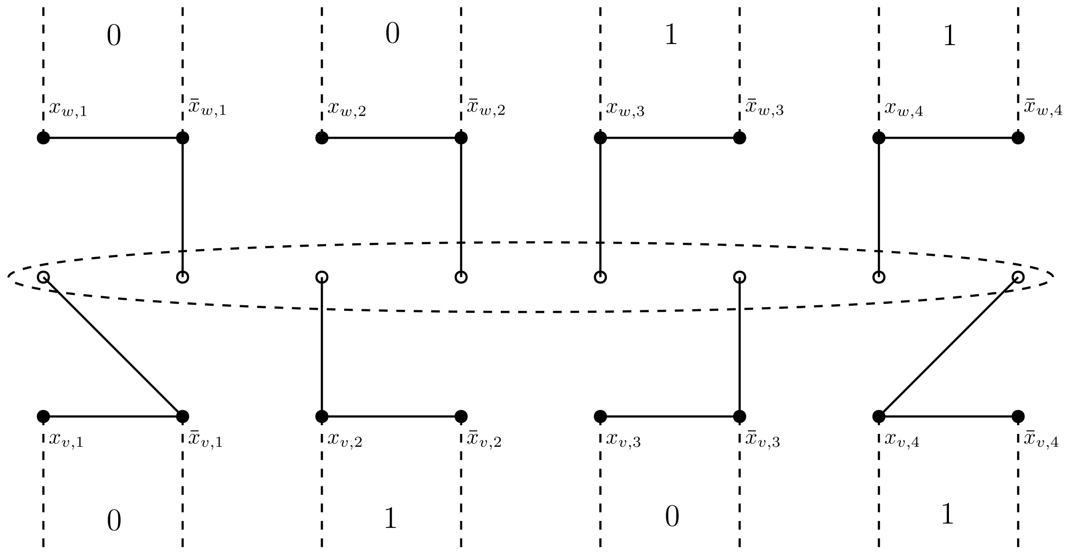

Figure 4.3: Example of a gadget $\Delta(v, w, c, c')$ where $l = 4$, and where $c, c' \in \mathcal{C}$ have bitstring representations 0011 and 1100 respectively. Note that all vertices within the dashed ellipse are part of one clique.

4.2 Dominating Set

Inspiration for the proof of the following result was also taken from [4], in this case the proof of Dominating Set's XNLP-hardness when parameterised by logarithmic tree-depth.

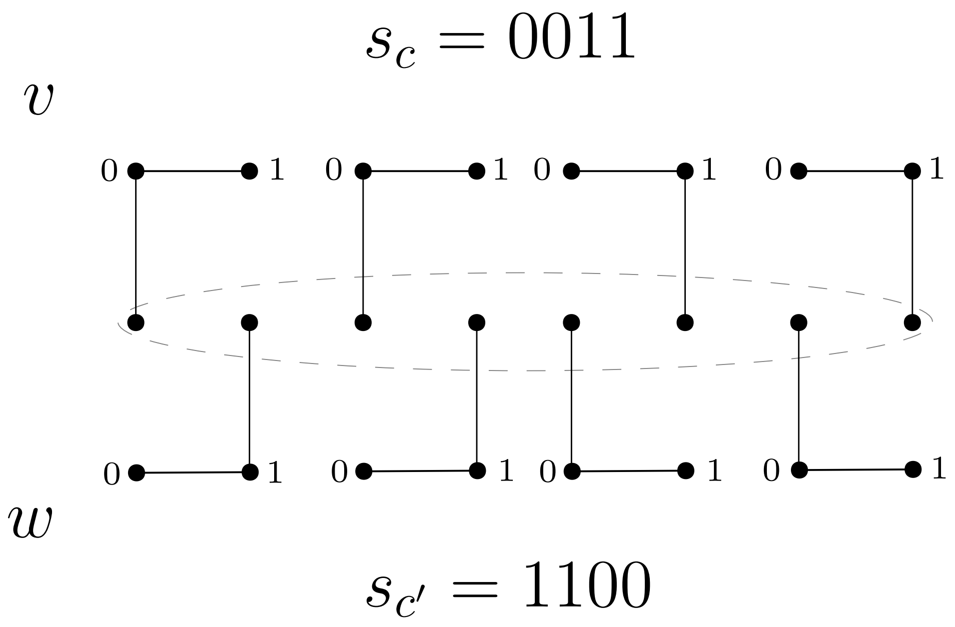

Figure 4.4: Another example of a gadget $\Delta(v, w, c, c')$ where $l = 4$, and where $c, c' \in \mathcal{C}$ have bitstring representations 0011 and 0011 respectively. Note that all vertices within the dashed ellipse are part of one clique.

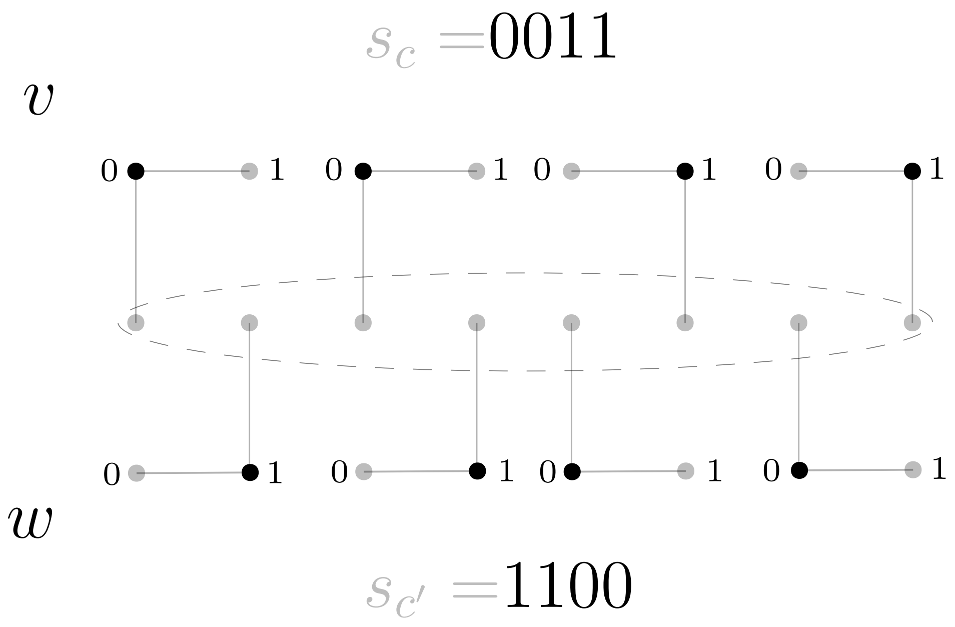

Figure 4.5: Example of a gadget $\Delta(v, w, c, c')$, where v and w pick the colours c and c' respectively. Note that the clique between the gadgets is unable to contribute any vertices to the independent set.

Theorem 4.2.1. *Dominating Set parameterised by tree-depth is XSLP-hard.*

Proof. We again provide a reduction from the Precolouring Extension problem, where we start with an undirected graph G, a colour set $\mathcal{C}$, and a precolouring function $p : S \to \mathcal{C}$ for some subset $S \subseteq V$. We also have T as the tree-depth spanning tree of G with height td. As with the proof of Theorem 4.1.1, we assume that $\mathcal{C} = 2^l$ for some integer l, and we give each colour $c \in \mathcal{C}$ its own unique bit string s_c. We can now begin defining an instance of Dominating Set by constructing the accompanying graph G'.

For every $v \in V$, we create l three-vertex cliques in G', such that for each $i \in [1, l]$ we have a triangle of vertices $x_{v,i}, \bar{x}_{v,i}$, and $y_{v,i}$. During the construction of G', we'll ensure that every $y_{v,i}$ is

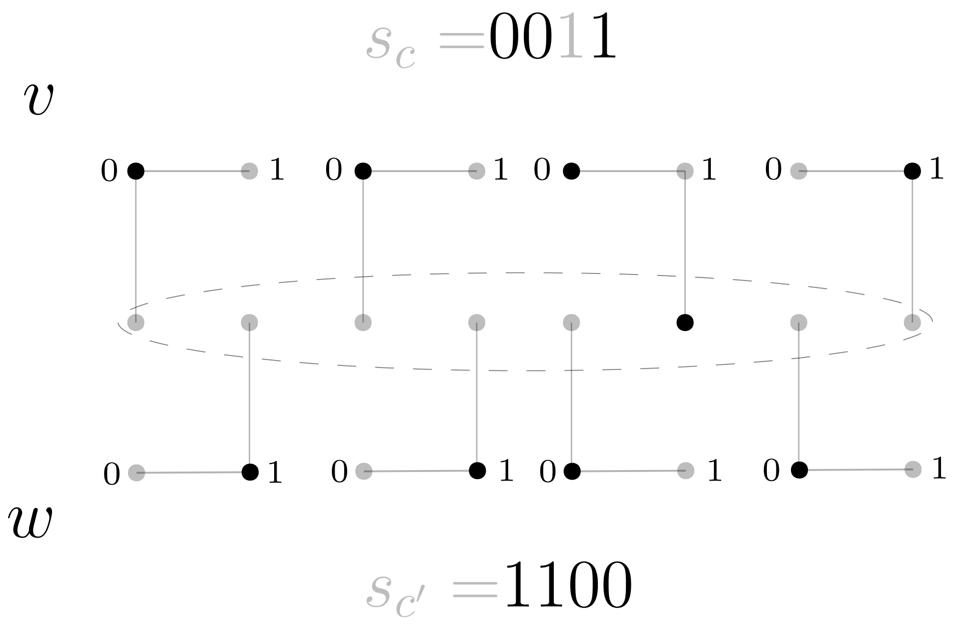

Figure 4.6: Example of a gadget $\Delta(v, w, c, c')$, where v and w do not pick the colours c and c' respectively. Note that the clique can now contribute exactly one vertex to the independent set.

only adjacent to $x_{v,i}$ and $\bar{x}_{v,i}$. The intuition for including such a 'dummy' vertex is that it ensures that any dominating set on G' must include at least one of the three vertices $x_{v,i}, \bar{x}_{v,i}$ and $y_{v,i}$ for every $v \in V$ and $i \in [1, l]$. Figure 4.8 shows an example for such a gadget, and Figure 4.9 shows how a dominating set of size l on the gadget corresponds to exactly one colour which can be applied to v.

Similar to with the proof of Theorem 4.1.1, we must consider the vertices in S which are precoloured; for those vertices $u \in S$ with $p(u) = c$, we remove any vertex $x_{u,i}$ or $\bar{x}_{u,i}$ which does not correspond with the i-th bit of s_c.

Then, for every $\{v, w\} \in E$ and for every same-colour pair $c, c' \in \mathcal{C}$ we create a vertex $z_{v,w,c,c'}$, which we connect to the 'complements' of the bit strings s_c and s_c' associated with c and c' respectively; that is, for every $i \in [1, l]$ we connect $z_{v,w,c,c'}$ to $x_{v,i}$ via an edge if the i-th bit of s_c equals 0, and otherwise we connect $z_{v,w,c,c'}$ to $\bar{x}_{v,i}$ via an edge. Similarly, we connect $z_{v,w,c,c'}$ to $x_{w,i}$ via an edge if the i-th bit of s_c' equals 0, and otherwise we connect $z_{v,w,c,c'}$ to $\bar{x}_{w,i}$ via an edge. Figure 4.10 shows an example of this. The intuition here is that, if both v and w have picked bitstrings via their associated gadgets that correspond to the same-colour pair c, c', the vertex $z_{v,w,c,c'}$ will not have any neighbours in the dominating set, and will therefore have to be included in the dominating set itself.

We are ready to state the following lemma:

Lemma 4.2.2. *There exists a valid colouring on G given the precolouring p if and only if a dominating set exists on G' of size $|V|l$.*

Proof. Assume first that the second statement holds, i.e. there exists a dominating set D on G' with $|D| = |V|l$. Since for every $u \in V$ and for every $i \in [1, l]$ we have that the clique consisting of $x_{u,i}, \bar{x}_{u,i}$ and $y_{u,i}$ must contribute one vertex to D, we know that every such clique contributes exactly one vertex, which we assume to be either $x_{u,i}$ or $\bar{x}_{u,i}$, since, in the case that $y_{u,i}$ was picked for D, we can arbitrarily swap it for one of the other two vertices, and retain a dominating set of size $|V|l$ on G'.

It follows then that D can be translated to a colouring $\gamma : V \to \mathcal{C}$ on G' by taking $\gamma(u) = c$ for the bit string s_c that matches the vertices of the type $x_{u,i}$ and $\bar{x}_{u,j}$ picked by the dominating set (in the case where $u \in S$, we know that $c = p(u)$ always holds). Furthermore, since for any $\{v, w\} \in E$ and any same-colour pair $c, c' \in \mathcal{C}$ we have that $z_{v,w,c,c'}$ is not in the dominating set and therefore adjacent to one of the bit vertices picked by v or w, we know that the bit strings associated with $\gamma(v)$ and $\gamma(w)$ cannot both be the same as the bit strings of c and c' respectively, and therefore $\gamma(v)$ and $\gamma(w)$ cannot indicate the same colour. We conclude that γ is a valid colouring of G given the precolouring function p.

For the converse proof, assume now that a valid colouring $\gamma : V \to \mathcal{C}$ exists on G given p. Then, for any $u \in V$ and $i \in [1, l]$, we include $x_{u,i}$ in D if the i-th bit of $s_{\gamma(u)}$ is 1, and we include $\bar{x}_{u,i}$ if the i-th bit is 0. Note that this is possible even when $u \in S$ given how we constructed G', and also note that the resulting set D has a cardinality that is exactly $|V|l$. Furthermore, we have that every vertex of the type $z_{v,w,c,c'}$ is adjacent to a vertex in D, since otherwise it would be the case that $\gamma(v) = c$ and $\gamma(w) = c'$ are both true, and therefore they are the same colour. We conclude that D is a dominating set on G'. □

With Lemma 4.2.2 proven, we have shown that for every instance of List Colouring an equivalent instance of Dominating Set exists. Since G' has a polonymial number of components relative to G, it follows that it is also a logspace reduction. The only remaning thing to do therefore is to show that the tree-depth of G' is within a factor $r \cdot \log(|V|)$ of td for some constant r. We will construct a tree-depth spanning tree T' of G' in an analogous manner as in the proof of Theorem 4.1.1.

For every pair $u \in V$, we sort the vertices $x_{v,1}, \bar{x}_{v,1}, y_{v,1}, ..., x_{v,l}, \bar{x}_{v,l}, y_{v,l}$ into a path p_u. Then, for any pair $v, w \in V$ where v is a parent of w in T, we connect the endpoint p_v to the root of p_w. Then, for every vertex of the type $w_{v,w,c,c'}$, we assume w.l.o.g. that v is an ancestor of w in T, and we connect $w_{v,w,c,c'}$ to the endpoint of p_w, which increases the total depth of T' by at most 1, so that T' is a tree-depth spanning tree of G' whose height is at most $3l \cdot td + 1$. □

Note that in the proof of Lemma 4.2.2 the dominating set D that is constructed on G' is an independent set as well. The reduction used is therefore also a valid pl-reduction from Precolouring Extension parameterised by tree-depth to Independent Dominating Set parameterised by tree-depth. Note also that the proof of the reduction can easily be adapted to work for the pathwidth and treewidth parameters, and furthemore that Precolouring Extension has already been proven to be XNLP-complete as well as XALP-complete for pathwidth and treewidth respectively ([4], [2]). We can conclude the following corollary result:

Corollary 4.2.2.1. *Independent Dominating Set is XSLP-hard when parameterised by logarithmic tree-depth, and XNLP-hard when parameterised by logarithmic pathwidth, and XALP-hard when parameterised by logarithmic treewidth.*

Another variant of the Dominating Set problem is *Roman Domination*, for which a problem instance consists of an undirected graph $G = (V, E)$, a nonnegative k, and the goal is to decide whether a function $f \to \{0, 1, 2\}$ with $\sum_{v \in V} f(v) = k$ exists such that for every $v \in V$ we have that $f(v) > 0$ or that $(v, w) \in E$ for some $w \in V$ with $f(w) = 2$. The problem is named as such because is it analogous to the problem of utilising a minimum number of Roman legions to pacify a given collection of cities, where a city is considered pacified if it either contains a legion, or is neighbouring to a city which has two legions stationed in it.

Theorem 4.2.3. *Roman Domination parameterised by logarithmic tree-depth is XSLP-hard.*

Proof. For our proof we will use the exact same reduction from Precolouring Extension as shown in Theorem 4.2.1, except that every bit selection gadget used will consist not of three but of four vertices $x_{v,i}$. $\bar{x}_{v,i}$, $y_{v,i}$, and $y'_{v,i}$, where the latter vertices are not adjacent to any vertices except the former two. The structure of the bit selection gadgets is shown in Figure 4.7. In the case that a given $u \in S$ is precoloured, we remove the bit vertices that do not correspond with the preselected colour of u. Note that no assignment of fewer than two legions is enough to cover every vertex in the gadget. It follows that an assignment of $|V| \cdot 2l$ legions exists on G' if and only if a colouring exists on G given the precolouring p, via the same proof as in Theorem 4.2.2. □

Given that Precolouring Extension is XALP-complete ([2]) and XNLP-complete ([4]) for the parameters treewidth and pathwidth respectively, we can also conclude two hardness results for Roman Domination:

Corollary 4.2.3.1. *Roman Dominating Set parameterised by logarithmic pathwidth is XNLP-hard, and Roman Dominating Set parameterised by logarithmic treewidth is XALP-hard.*

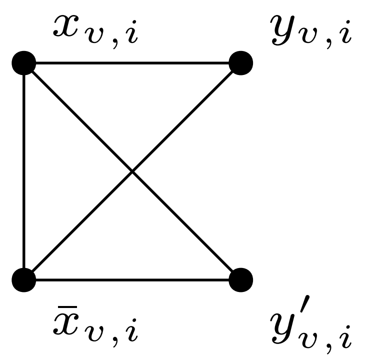

Figure 4.7: Bit selection gadget for the reduction graph from Precolouring Extension to Roman Domination.

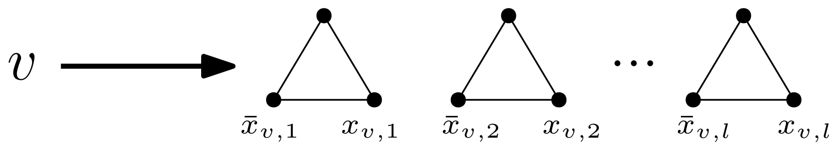

Figure 4.8: Colour selection gadget for a vertex v in the reduction graph from Precoloring Extension to Dominating Set.

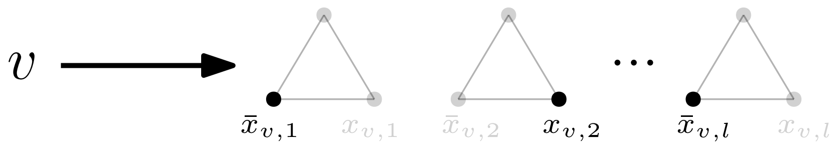

Figure 4.9: Colour selection gadget for a vertex v in the reduction graph from Precoloring Extension to Dominating Set, where a dominating set of size l corresponds to exactly one colour which can be applied to v.

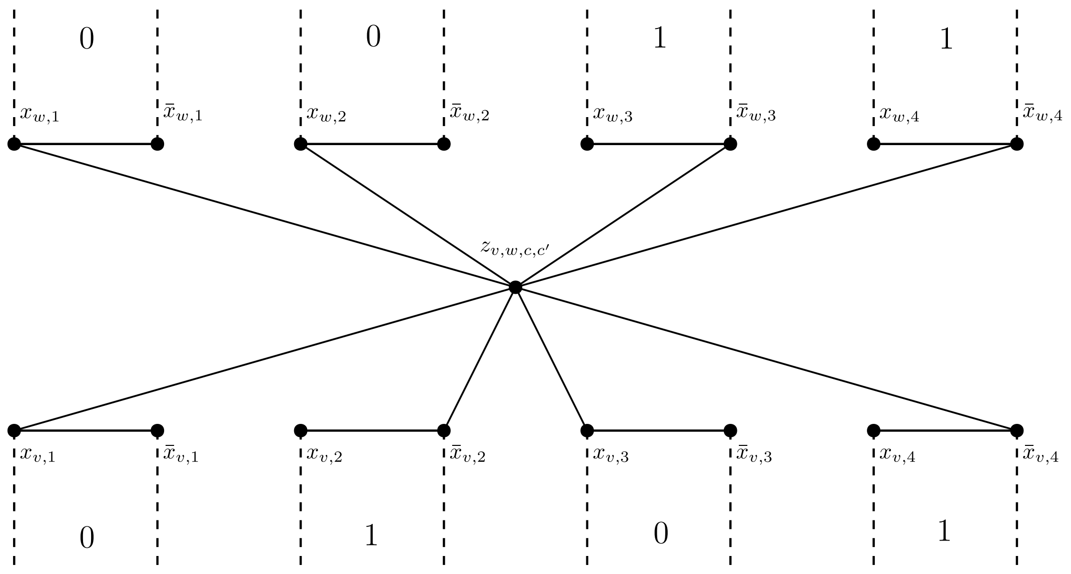

Figure 4.10: Example of a 'gadget' $z_{v,w,c,c'}$ where $l = 4$, and where $c, c' \in \mathcal{C}$ have bitstring representations 0101 and 0011 respectively.

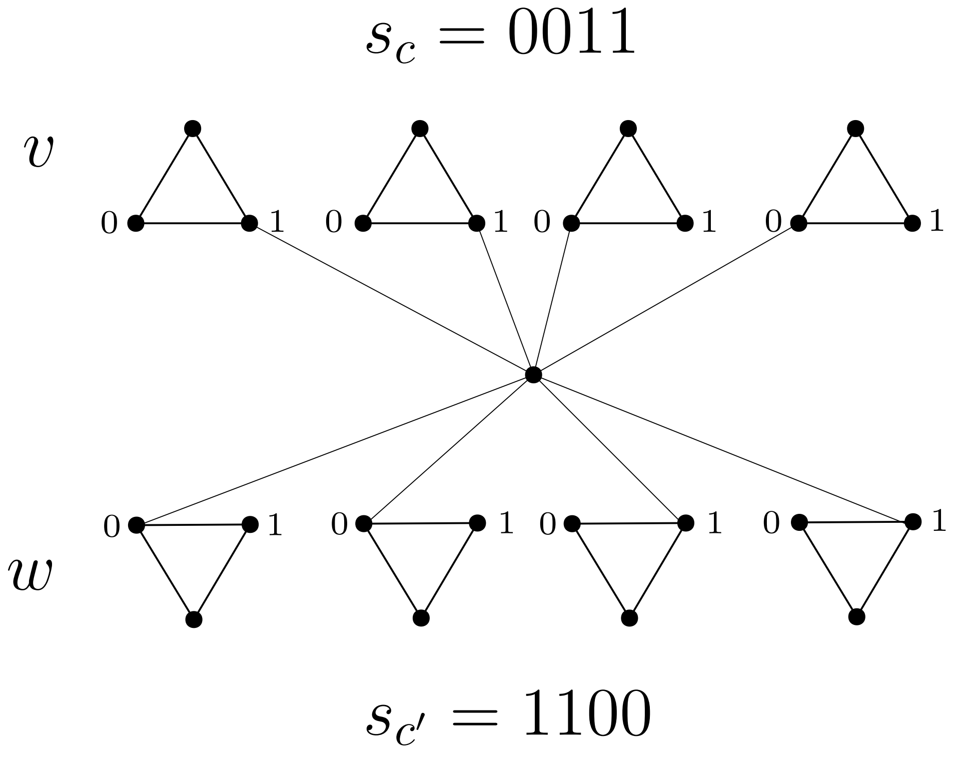

Figure 4.11: Another example of a vertex $z_{v,w,c,c'}$ where $l = 4$, and where $c, c' \in \mathcal{C}$ have bitstring representations 0011 and 1100 respectively.

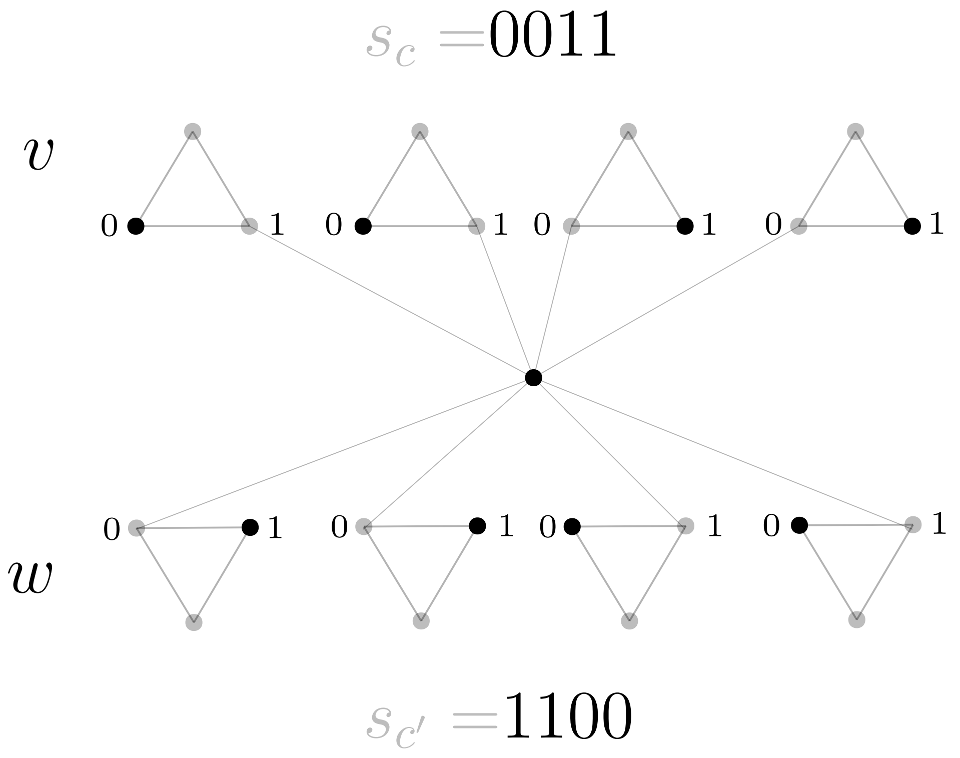

Figure 4.12: Example of a vertex $z_{v,w,c,c'}$, where v and w pick the colours c and c' respectively. Note that $z_{v,w,c,c'}$ has to be included in the dominating set on G'.

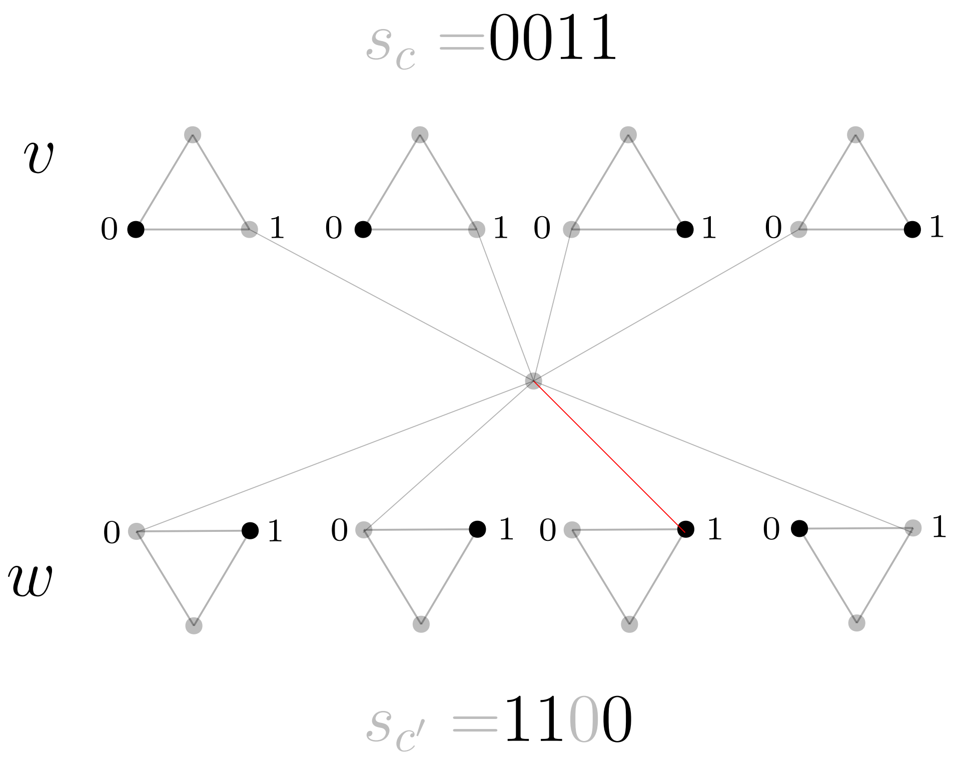

Figure 4.13: Example of a vertex $z_{v,w,c,c'}$, where v and w do not both pick the colours c and c' respectively. Note that $z_{v,w,c,c'}$ can now be excluded from the dominating set on G'.

Chapter 5

Completing the reduction graph for tree-depth

5.1 Existing reductions

We begin this section by proving that the impartial reduction graph shown in Figure 3.2 is correct when every problem is parameterised by tree-depth. For clarity, we give each reduction its separate theorem. Note that in the proofs of these theorems, we use $G = (V, E)$ to refer to the graph belonging to the problem instance from which the reduction is performed, and $G' = (V', E')$ to refer to the graph belonging to the problem instance to which the reduction is performed. We use T and T' to indicate the minimum-height tree-depth spanning trees of G and G' respectively, with td and td' the respective heights of both trees.

Theorem 5.1.1. *There exists a pl-reduction from All-or-Nothing Flow to Target Outdegree Orientation, with both problems parameterised by tree-depth.*

Proof. On p. 13 of [1], we are given a reduction from All-or-Nothing Flow to Target Outdegree Orientation. This reduction creates a graph G' identical to G, but with undirected edges, and where each arc in G has been subdivided into two edges with a connecting vertex in between. Doing so increased the tree-depth of G' by at most 1 relative to G. $\square$

Theorem 5.1.2. *There exists a pl-reduction from Target Outdegree Orientation to Chosen Maximum Outdegree, with both problems parameterised by tree-depth.*

Proof. On p. 16 of [1], we see a reduction where G remains unchanged in creating G', which means that tree-depth is unchanged. $\square$

Theorem 5.1.3. *There exists a pl-reduction from Chosen Maximum Outdegree to Minimum Maximum Outdegree, with both problems parameterised by tree-depth.*

Proof. On p. 15 of [1], we see a reduction where only two new vertices are added, which means that the tree-depth cannot increase by more than two. $\square$

Theorem 5.1.4. *There exists a pl-reduction from Target Outdegree Orientation to Circulating Orientation, with both problems parameterised by tree-depth.*

Proof. Also on p. 15, we see a reduction that only adds a source and sink node, and so does not increase the tree-depth by more than 2. $\square$

Theorem 5.1.5. *There exists a pl-reduction from Circulating Orientation to Red-Blue Capacitated Dominating set, with both problems parameterised by tree-depth.*

Proof. The reduction outlined on p. 18 of [6] copies every vertex of G to G', and gives each $v \in V$ a private neighbour v' which is only adjacent to v. For each edge $\{v, w\} \in E$, a gadget is created in G' whose vertices are only adjacent to each other and to v and w. It is easy to see how these vertices can be incorporated into T' so that T' has a height of at most $td + 5$. $\square$

Theorem 5.1.6. *There exists a pl-reduction from Red-Blue Capacitated Dominating Set to Capacitated Dominating Set, with both problems parameterised by tree-depth.*

Proof. In the same proof on p. 18 of [6], a standard reduction from RBCDS to CDS is given which adds only one new vertex to G', thereby only increasing the tree-depth by at most 1. □

The provided reduction for XNLP-hardness of Red-Blue Capacitated Dominating Set and subsequently Capacitated Dominating Set can also be easily shown to have bounded treewidth relative to the original graph. Since Circulating Orientation parameterised by treewidth is XALP-complete [2], it follows that Red-Blue Capacitated Dominating Set is also XALP-hard (although this result follows trivially from [2], it was not explicitly mentioned there).

Corollary 5.1.6.1. *Red-Blue Capacitated Dominating Set and Capacitated Dominating Set are XALP-hard when parameterised by treewidth.*

In this part of the section, we give the reductions necessary to complete the strongly connected reduction graph shown in Figure 3.3.

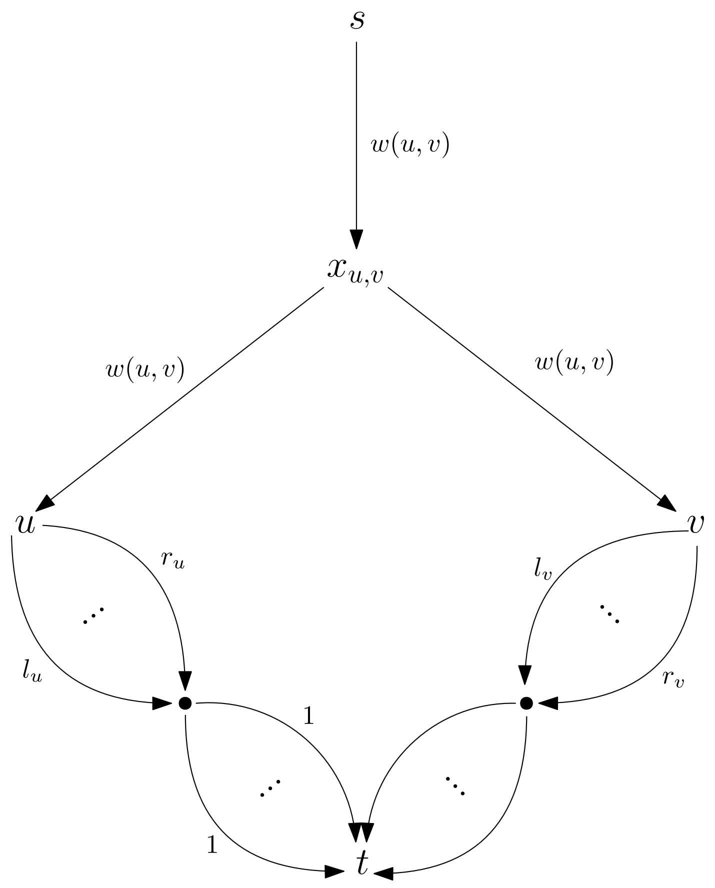

Figure 5.1: Edge gadget in the reduction graph from Outdegree Restricted Orientation to All or Nothing FLow

5.2 Outdegree Restricted Orientation to All-or-Nothing Flow

Theorem 5.2.1. *A pl-reduction exists from Outdegree Restricted Orientation to All-or-Nothing Flow, with both problems parameterised by tree-depth.*

Proof. We start with an instance of Outdegree Restricted Orientation, i.e. an undirected graph $G = (V, E)$, as well as a weight function $w : E \to \mathbb{Z}_{>0}$, and, for each $v \in V$, an interval $D_v \in \mathbb{Z}_{\geq 0}$.

We now create an instance of All-or-Nothing Flow. To that end, let $G' = (V \cup \{s, t\}, A)$ be a directed graph with the same vertex set as G, as well as a source s and a sink t. We will define the set of arcs A in the following paragraph.

For every $u \in V$, we create a vertex u', and for each $i \in [l_u, r_u] = D_u$ we create an arc (u, u') with capacity i. Additionally, we create a r_u arcs from u' to t, each with capacity 1. Then, for every edge $(u, v) \in E$, we create an extra vertex $x_{u,v}$, and we connect it in the graph with $(s, x_{u,v})$, $(x_{u,v}, u)$, $(x_{u,v}, v)$, with each arc having capacity $w(u, v)$. Let $D_u = [l_u, r_u]$, and $D_v = [l_v, r_v]$. Figure 5.1 shows this construction. The intuition here is that, for every edge $\{u, v\} \in E$, an orientation must be chosen by either sending flow from $x_{u,v}$ to u, or from $x_{u,v}$ to v. Subsequently, every vertex u must siphon its incoming flow to t, which, by the construction shown in Figure 5.1, is guaranteed to be within the permitted range $[l_u, r_u]$.

Lemma 5.2.2. *An orientation exists for G such that for every $v \in V$ the weight sum of its outgoing edges is contained in D_v, if and only if an all-or-nothing flow f exists for G' with flow value $\sum_{e \in E} w(e)$.*

Proof. First suppose we have an orientation G such that for every $u \in V$, the sum of weights of all its outgoing edges is contained in $D_u = [l_u, r_u]$. We create a flow f on G' by first setting $f(s, x_{u,v}) = w(u, v)$ for every $(u, v) \in E$. If (u, v) is directed to u in G, we set $f(x_{u,v}, v) = w(u, v)$; otherwise, we set $f(x_{u,v}, u) = w(u, v)$, so that for each $u \in V$, we have that u has an inflow in G' from f that is equal to the weight sum of its outgoing arcs in the orientation of G. It follows then, that the total inflow ι of u by f is contained in $[l, u, r_v]$, and therefore we can siphon its inflow to u' by the arc whose capacity is exactly ι, and then siphon the inflow from u' to t' via 1 capacity arcs, of which we are guaranteed to have at least ι. The flow f is therefore a valid all-or-nothing flow on G' which has a flow value of $\sum_{e \in E} w(e)$.

To prove the lemma in the opposite direction, suppose we start with a flow f on G' for which $|f| = \sum_e w(e)$. It follows then that for every $(u, v) \in E$ the flow 'selects' either u or v to siphon flow to via $x_{u,v}$, which is analogous to directing the edge $(u, v) \in G$ towards v or to u respectively; indeed, if we create such an orientation on G', we see that every vertex $u \in V$ has a weight sum of outgoing arcs that is equal to the inflow of u from f in G'. Since the lowest-capacity arc from u to u' has capacity l_u, and since the total sum of all capacities from u' to t is r_u, we must have that the inflow of u in G' (and therefore the weight sum of its outgoing edges in G) is contained in D_u. We conclude that a valid outdegree-restricted orientation exists for G. □

From the proof of Lemma 5.2 it follows that we have successfully reduced the instance of Outdegree Restricted Orientation to an instance of All-or-Nothing Flow; what remains to be proven is that the reduction graph G' has restricted tree-depth t' relative to the tree-depth t of G. To that end, let T be a tree-depth spanning tree of G which has depth t. We construct a tree-depth spanning tree T' of G' by connecting u and u' via an edge for every $u \in V$, and then by connecting v' to w for every pair $v, w \in V$ for which v is a parent of w in T. Clearly, this creates a tree-depth spanning tree T' of (part of) G' with depth $2t$. To complete the tree-depth spanning tree, we place s and t at the root of T', and we create an edge from $x_{u,v}$ to v for every $(u, v) \in E$, where assume w.l.o.g. that v is a descendant of u in T, which increases the depth of T' to at most $2t + 3$. □

5.3 Undirected Flow with Lower Bounds to All-or-Nothing Flow

Theorem 5.3.1. *A parameterised log-space reduction exists from Undirected Flow with Lower Bounds parameterised by tree-depth to All-or-Nothing Flow parameterised by tree-depth.*

Proof. We start with an instance of Undirected Flow with Lower Bounds parameterised by tree-depth, i.e. an undirected graph $G = (V, E)$ with sink and source nodes s and t, as well as a positive integer R, and capacity and lower bound functions $c : E \to \mathbb{Z}_{>0}$, as well as a tree-depth spanning tree T of G with height t. We will now construct a directed flow graph G' on which to define an instance of All-or-Nothing Flow.

To start with, we create a new sink and source s' and t', as well as a spare source s'_K which we connect to s' via (s', s_K) with capacity $|V|$, and, for every $v \in V$, we create two vertices v_{in} and v_{out}. The only exceptions are s and t, for which we only create s_{out} and t_{in} respectively. Additionally, for

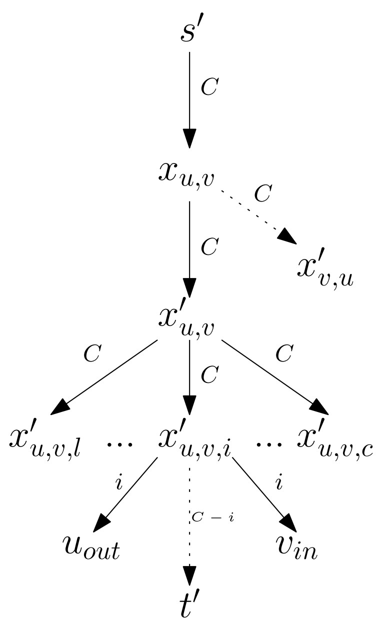

Figure 5.2: Edge gadget for the reduction graph of Undirected Flow with Lower Bounds to All-or-Nothing Flow. Note that we have substituted $l = l(u,v)$ and $c = c(u,v)$ for better visual clarity.

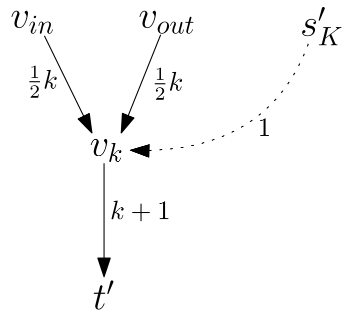

Figure 5.3: Vertex gadget for the reduction graph of Undirected Flow with Lower Bounds to All-or-Nothing Flow, to ensure that inflow and outflow are the same for every vertex v.

every $\{u,v\} \in E$ we create an edge gadget by creating three vertices $x_{u,v}$, $x'_{u,v}$ and $x'_{v,u}$, which we connect via the arcs $(s', x_{u,v})$, $(x_{u,v}, x'_{u,v})$ and $(x_{u,v}, x'_{v,u})$ with capacity C each, where we define as the maximum across all capacities $C = \sum_{e \in E} c(e)$. Then, for every $i \in [l(u,v), c(u,v)]$, we create a vertex $x'_{u,v,i}$ and a vertex $x'_{v,u,i}$, and we connect them via arcs $(x'_{u,v}, x'_{u,v,i})$ with capacity C, and $(x'_{v,u}, x'_{v,u,i})$ with capacity c. We also then connect $x'_{u,v,i}$ to u_{out} and to v_{in} via arcs with capacity i, and siphon its excess inflow to t' via an arc with capacity $C-i$. Similarly, we connect $x'_{v,u,i}$ to u_{in} and v_{out} via arcs with capacity i, and connect it via an arc to t' with capacity $C-i$. Figure 5.2 shows (part of) this construction. Intuitively, we have that sending C flow to $x'_{u,v,i}$ is equivalent to directing the edge between u and v in G towards v, and sending an $i \in [l(u,v), c(u,v)]$ amount of flow along it. The vertices v_{out} and v_{in} then represent the outflow and inflow that v receives respectively.

As a small amendment, for edges in G that include s or t as an endpoint, we only include the vertices in G' that correspond to sending flow outwards from s, and inwards to t, since any solution to the UFLB problem must be a valid flow from s to t, and can therefore not have incoming flow to s or outgoing flow from t.

In the following, let $W(v)$ be the sum of all capacities $c(e)$ of edges incident to v. To ensure that every $v \in V \setminus \{s\}$ receives an inflow that is equal to its outflow, we create for every $k \in \{0, 2, 4..., W(v)\}$ a vertex v_k, and we create an arc (v_{in}, v_k) and (v_{out}, v_k) each with capacity $\frac{1}{2}k$. Additionally, we create (s'_K, v_k) with capacity 1, and (v_k, t') with capacity $k+1$. Intuitively, we have that s'_K 'pre-selects' a vertex v_k for every $v \in V$, so that every v must have the same inflow and same outflow $\frac{1}{2}k$. Additionally, for s and t, we only create the arcs (s_{out}, t') and (t_{in}, t') with capacity R for each.

Lemma 5.3.2. *There exists an orientation of G such that a flow f of value R can exist on G given capacity and lower bound functions c and l, if and only if an all-or-nothing flow f' exists on G' with value $|V|K + |E|C$.*

Proof. We assume first that an orientation of G exists that allows a flow f of value R, given capacity and lower bound functions c and l. To create a flow f' on G', we direct a C amount of flow from s' to $x_{u,v}$ for every $\{u, v\} \in E$, and, assuming w.l.o.g. that the edge $\{u, v\}$ is directed towards v in the orientation of G, we use $i = f(u, v)$ to refer to the integer flow value that is sent from u to v in f. We then direct a C amount of flow from $x_{u,v}$ to $x'_{u,v,i}$, and we use every outgoing arc from $x'_{u,v,i}$ to its maximum capacity. Additionally, for every $v \in V$ we compute the value k as the total inflow of f towards v in G' (which is necessarily equal to its total outflow), and we direct a K amount of flow from s' to v_k in G'. It is easy to see that f' has a total flow value of $|V|K + |E|C$ as a result. Additionally, since every $v \in V$ has inflow and outflow values that are equal to $\frac{1}{2}k$, we have that every pair v_{in} and v_{out} is able to pass on its collective inflow to the vertex v_k, which can in turn pass the flow on to t'.

For the converse proof, we assume that an all-or-nothing flow f' exists on G' such that $|f'| = |V|K + |E|C$. From the fact that the total sum of capacities of outgoing arcs from s' is exactly $|V|K + |E|C$, we must conclude that for every $v \in V$ a single v_k is 'selected' by f' (since every $v \in V$ must have at least one v_k for which flow is provided by s'_K), as well as that for every $\{u, v\} \in E$, a direction of the edge $\{u, v\}$ with accompanying flow i across the directed arc is chosen via the vertex $x'_{u,v,i}$ or $x'_{v,u,i}$. If we translate this selection by f' into an orientation on G with accompanying flow f, we can easily see how f must have a total flow value $|f| = R$ (by the fact that both s_{out} and s_{in} were able to send R flow to t' in G'), and we can also see that every $v \in V$ must receive an equal amount of inflow as well as outflow k by the fact that every pair v_{in} and v_{out} was able to pass on its flow to a vertex v_k. □

With Lemma 5.3.2 proven, we only have to show that the reduction graph G' has bounded tree-depth relative to the original graph G. To that end, observe that we can copy the structure of T to a tree-depth spanning tree T' of G', where s', $s'K$, and t' are placed at the root of the tree, and where we connect every pair u_{in}, u_{out} in T' and by connecting u_{out} to v_{in} for any $v \in V$ which is a direct child of u in T. Then, for every vertex of the type v_k, we can add it as a separate child to v_{out} in T'. For the edge vertices of the type $x_{u,v}$, $x'_{u,v}$, and $x'_{v,u}$, observe that we can simply suspend $x_{u,v}$ below v in T' (assuming w.l.o.g. that u is the ancestor of v in T'), and that we can then add $x'_{u,v}$ and $x'_{v,u}$ as separate children of $x_{u,v}$. Lastly, we can connect every vertex of the type $x'_{u,v,i}$ and $x'_{v,u,i}$ as a separate child of $x'_{u,v}$ and $x'_{v,u}$ respectively.

By studying the structure of T', we can conclude that it has a tree-depth of at most $2t+6$, meaning it has bounded tree-depth relative to the original graph G. □

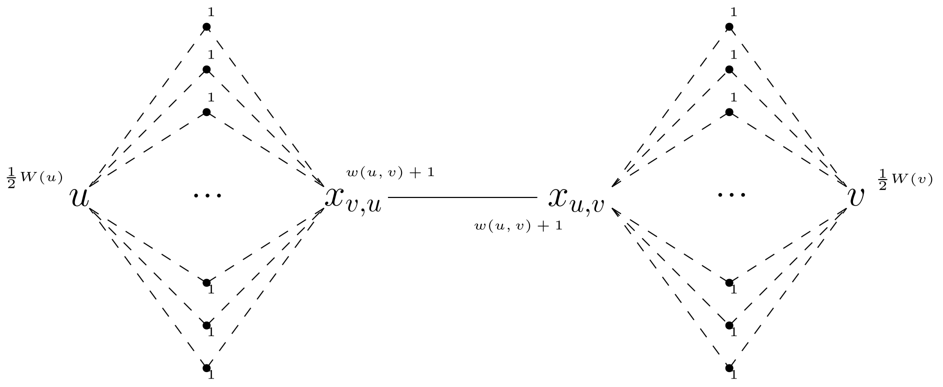

Figure 5.4: Edge gadget for the reduction graph of Circulating Orientation to Target Set Selection.

5.4 Circulating Orientation to Target Set Selection

Theorem 5.4.1. *A parameterised log-space reduction exists from Circulating Orientation to Target Set Selection, where each problem is parameterised by the tree-depth parameter.*

Proof. We start with an instance of Circulating Orientation parameterised by tree-depth, i.e. a graph $G = (V, E)$ with a weight function $w : E \to \mathbb{Z}_{>0}$, as well as a tree-depth spanning tree T of G with height t. We construct the reduction graph G' by moving every $v \in V$ to G', and by creating an additional edge gadget for every $\{u, v\} \in E$. This edge gadget is comprised of two adjacent vertices $x_{v,u}$ and $x_{u,v}$, as well as $2 \cdot w(u, v)$ unlabelled vertices, half of which we connect on individual paths between $x_{v,u}$ and u, and half of which we connect on individual paths between $x_{u,v}$ and v. Figure 5.4 shows this construction. We give $x_{v,u}$ and $x_{u,v}$ each a threshold value of $\tau(x_{v,u}) = \tau(x_{u,v}) = w(u, v) + 1$, and we give every other vertex in the edge gadget a threshold of 1. Lastly we set the threshold value of every $v \in V$ as $\frac{1}{2}W(v)$, where $W(v)$ is the summed total weight of edges incident to v.

Lemma 5.4.2. *There exists a circulating orientation of G given the weight function w, if and only if a target set S_0 of size $|E|$ exists for G' that activates every vertex given threshold function τ.*

Proof. We assume first that a circulating orientation exists for G, and we define our target set S_0 for G' to be comprised of every vertex $x_{v,u}$ for which the edge $\{u, v\} \in E$ is directed towards u, and of every vertex $x_{u,v}$ for which $\{u, v\}$ is directed towards v. Clearly, the resulting set S_0 has exactly $|E|$ elements as a result. Furthermore, we can define S_1 as the subset of V' that contains only those unlabelled vertices which are connected between u and $x_{v,u}$ for which $x_{v,u} \in S_0$, as well as those vertices connected between $x_{u,v}$ and v for which $x_{u,v} \in S_0$. We then define $S_2 = V$, and S_3 as the set of vertices with threshold 1 whose neighbour of the type $x_{u,v}$ was not picked in S_0, and S_4 as the set of vertices $x_{u,v}$ which were not picked in S_0.

From observing Figure 5.4, we have that the sequence S_0, S_1, S_2, S_3, S_4 defines a disjoint decomposition of the vertex set V' of G'. Furthermore, since every vertex in S_1 has a threshold of 1 and was picked to have a neighbour in S_0, we have that S_0 successfully activates S_1. Lastly, since we defined our target set to be analogous to the existing circulating orientation of G, it is easy to see how every $v \in V$ has exactly $\frac{1}{2}W(v)$ neighbours in S_1, which in turn allows it to activate the subsequent set S_3 whose vertices all have a threshold of 1. We conclude therefore that S_0 is a valid target set of G'.

Given the simple nature of the converse part of the proof, we eschew the formal definition of the target set problem for the more intuitive informal one. Assume therefore that a target set S_0 exists that successfully activates the rest of G' where $|S_0| = |E|$. If, for a given $\{u, v\} \in E$, neither $x_{u,v}$ or $x_{v,u}$ was picked for S_0, we quickly reach a contradiction by noting that neither vertex could then have been activated in a subsequent stage. We must conclude, therefore, that S_0 contains exactly one vertex $x_{u,v}$ or $x_{v,u}$ for every $\{u, v\} \in E$, which in turn activates the $w(u, v)$ vertices adjacent to the vertex u or v respectively. We can easily translate this to an orientation of G by directing $\{u, v\}$ towards u in the case that $x_{v,u}$ was picked for S_0, and by directing $\{u, v\}$ towards v if $x_{u,v}$ was picked. It must then be the case that the resulting orientation is a circulating orientation, since the total weighted indegree of any $u \in V$ is exactly equal to the number of neighbours that u has in S_1, and if any vertex $u \in V$ had fewer than $\frac{1}{2}W(u)$ neighbours in S_1 it would not have been activated, and if it had strictly more than $\frac{1}{2}W(v)$ neighbours in S_1, it would have to be the case that at least one vertex v does not have sufficient activated neighbours, which would contradict our assumption that S_0 is a valid target set of G'. □

To create a tree-depth decompositon T' of G', we can copy the structure of T into T', and then for any edge $\{u, v\} \in E$ we can connect the edge vertices $x_{v,u}$ and $x_{u,v}$ as separate children of v, assuming w.l.o.g. that v is the descendant of u in T. The remaining edge vertices can then be added as separate children of $x_{v,u}$ and $x_{u,v}$, so that the total height of T' is at most only $td + 2$. □

Circulating Orientation was already proven to be XNLP-complete when parameterised by pathwidth in [1], as well as XALP-complete when parameterised by treewidth in [2]. We can therefore conclude the following corollary result:

Corollary 5.4.2.1. *Target Set Selection is XNLP-hard when parameterised by pathwidth, and XALP-hard when parameterised by treewidth.*

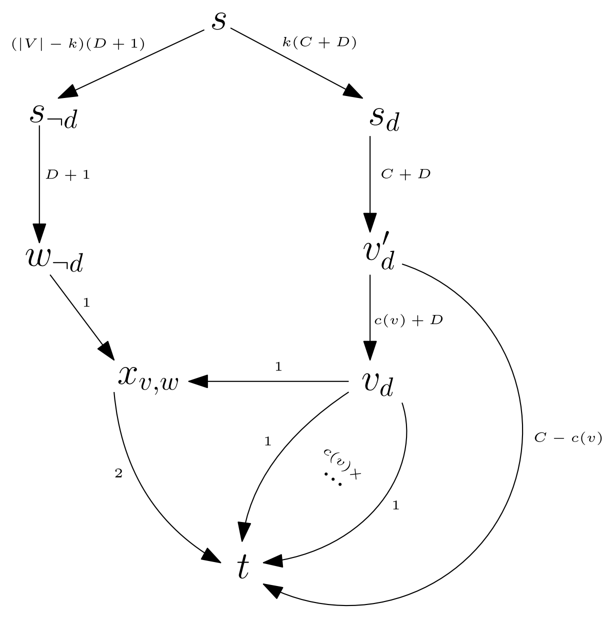

Figure 5.5: Reduction graph from Capacitated Dominating Set to All-or-Nothing Flow.

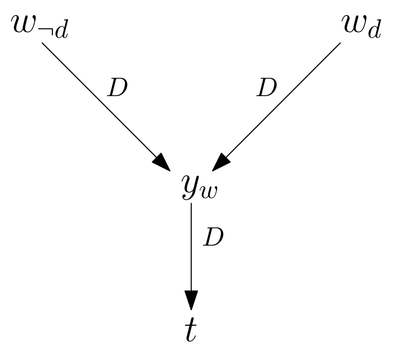

Figure 5.6: Vertex gadget for the reduction graph from Capacitated Dominating Set to All-or-Nothing Flow.

5.5 Capacitated Dominating Set to All-or-Nothing Flow

Theorem 5.5.1. *A pl-reduction exists from Capacitated Dominating Set to All-or-Nothing Flow, with both problems parameterised by tree-depth.*

Proof. We start with an instance of Capacitated Dominating set parameterised by tree-depth, i.e. an undirected graph $G = (V, E)$ with a tree-depth spanning tree T of depth t, as well as a capacity function $c : V \to \mathbb{N}$ and a target integer k. We construct a directed flow graph G' as follows: first we create source and sink vertices s and t, and then we create additional sink vertices s_d and $s_{\neg d}$, which we connect to s via the directed arcs (s, s_d) and $(s, s_{\neg d})$, which have the capacities $k(C + D)$ and $(|V| - k)(D + 1)$ respectively, where $C = \max_{v \in V} c(v)$ and $D = \max_{v \in V} d(v) + 1$. Then, for each $v \in V$, we create three vertices v'_d, v_d, and $v_{\neg d}$, and we create the arcs (s_d, v'_d), (v'_d, v_d), as well as $(s_{\neg d}, v_{\neg d})$, with capacities $C + D$, $c(v) + D$, and $D + 1$ respectively. Additionally, to siphon excess flow from v'_d and v_d, we create an arc (v'_d, t) with capacity $C - c(v)$, as well as $c(v)$ additional arcs from v_d to t, each with capacity 1.

Then, for every edge $(v, w) \in E$, we create a vertex $x_{v,w}$, and we connect it via an arc to t with capacity 2. Additionally, we create the arcs $(v_d, x_{v,w})$, $(v_{\neg d}, x_{v,w})$, $(w_d, x_{v,w})$, $(w_{\neg d}, x_{v,w})$, all with capacity 1. Figure 5.5 shows this construction.

Lastly, for every $w \in V$, we create a vertex y_w, which we connect via an arc to t with capacity D. Additionally, we create the arcs $(w_{\neg d}, y_w)$ and (w_d, y_w), each with capacity D. Figure 5.6 shows this part of the construction.

The intuition for why we built the graph this way goes as follows: if we set the required flow value high enough such that every arc going out of s has to be used, then the network is forced to 'select' a k number of vertices of the type v_d by directing flow from them from s_d, which in turn can be used to represent the dominating set. At the same time, the network has to select a $|V| - k$ number of vertices through $s_{\neg d}$, which together represent the vertices *not* chosen in the initial dominating set. If either a $w_{\neg d}$ or w_d is chosen, the integer D is sufficiently large that whichever vertex is chosen must send D flow to the vertex y_w, which ensures that no vertex is chosen to be both within and outside the dominating set. If we do have that $w_{\neg d}$ is chosen (and therefore w is not in the dominating set), then it can still release its inflow through a vertex $x_{v,w}$, provided that w has a neighbour v which has been dominated.

We are now ready to state the following lemma:

Lemma 5.5.2. *A dominating set of size k exists for G that abides by the capacity function c, if and only if a flow f exists on G' with flow value $|f| = (|V| - k)(D + 1) + k(C + D)$.*

Proof. Suppose first that a dominating set $S \subseteq V$ exists for G, i.e., a function $h : V \to S$ exists such that $\{w, f(w)\} \in E$ or $w = f(w)$ for all $w \in V$, while also the property holds that $|f^{-1}(v)| \leq c(v)$ for all $v \in S$.

We construct a flow f on G' as follows: we set $f(s, s_{\neg d}) = (|V| - k)(D+1)$, and we set $f(s, s_d) = k(C + D)$. Then, for each $w \in V \setminus S$, we set $f(s_{\neg d}, w_{\neg d}) = D + 1$, and for each $v \in S$, we set $f(s_d, v'_d) = C + D$. Since S and $V \setminus S$ are disjoint, every selected v_d and $w_{\neg d}$ will be able to send its excess inflow D to send to y_v and y_w. Additionally, since every $w_{\neg d}$ has a 'neighbouring' v_d which can send flow to $u_{v,w}$, it can siphon its excess flow of 1 through $u_{v,w}$ to t. Since we also have that $|f^{-1}(v)| \leq c(v)$ for all $v \in S$, we know that every selected v_d has sufficient inflow to supply its neighbours.

We conclude that a valid all-or-nothing flow exists on G' with the desired flow value.

For the converse proof, we assume that an all-or-nothing flow f exists on G with value $|f| = (|V| - k)(D + 1) + k(C + D)$. With similar reasoning as in the previous proof, it is easy to see how such a flow would 'select' exactly a k vertices of the type v_d, as well as $|V| - k$ vertices $v_{\neg d}$, which we can translate into a potentially dominating subset of V of size k. By the fact that every $w_{\neg d}$ is able to siphon its inflow to the sink t, we can conclude that every vertex w not selected to be in the initial dominating set has a neighbouring v that *is* selected, and by the fact that v_d can send out flow to at most $c(v)$ neighbouring edge vertices, we know that no dominating vertex exceeds its capacity for dominating neighbours. The set $S = \{v \in V : f(s_d, v'_d) > 0\}$ is therefore a valid capacitated dominating set of size k for G. □

From the proof of Lemma 5.5.2, it follows that the instance of Capacitated Dominating Set and the reduced instance of All-or-Nothing Flow are equivalent. What remains to be proven is that the graph G' has a tree-depth t' that is restricted with respect to t.

To that end, note that every vertex $v \in V$ has exactly four corresponding vertices $v'd$, v_d, $v_{\neg d}$, and y_w in G', and that for every edge vertex $x_{v,w}$ we have that it is only adjacent to vertices corresponding to v and w in G', which in turn are adjacent in G. We can therefore construct a tree-depth spanning tree of G' where s, s_d, $s_{\neg d}$, and t are placed at the root, such that its total height becomes $4td + 5$. □

5.6 Circulating Orientation to Capacitated Vertex Cover

Theorem 5.6.1. *There is a pl-reduction from Circulating Orientation to Capacitated Vertex Cover, with both problems parameterised by tree-depth.*

Proof. We start with an instance of Circulating Orientation parameterised by tree-depth, i.e. an undirected graph $G = (V, E)$ with a tree-depth spanning tree T of G whose height is td, as well as a weight function $w : E \to \mathbb{Z}_{>0}$. We construct our reduction graph G' by including in it every vertex $v \in V$, as well as an additional dummy vertex v' which we connect via an edge to v. We set the capacity of v as $\frac{1}{2}W(v) + 1$, and the capacity of v as 0. Here $W(v)$ is the total sum of the weights of edges incident to v. We also replace every edge $\{u, v\} \in E$ with an edge gadget as depicted in Figure 5.7. In this gadget there are two adjacent vertices $x_{v,u}$ and $x_{u,v}$, as well as $w(u, v)$ edges which are each on an individual path between u and v. We then create edges from $x_{v,u}$ to half of the endpoints of the $w(u, v)$ edges between u and v, and $x_{u,v}$ to the other half of the endpoints as depicted in

Figure 5.7. For the capacities of these vertices, we set $c(x_{v,u}) = c(x_{u,v}) = w(u,v)$, and we give all the remaining vertices in the edge gadget a capacity of 3.

With the construction out of the way, we are ready to state the following lemma:

Lemma 5.6.2. *There exists a circulating orientation for G given weight function w if and only if a vertex cover $S \in V$ exists for G' that abides by the capacity function c and where $|S| = |V| + |E| + \sum_{e \in E} w(e)$.*

Proof. Assume first that a circulating orientation exists on G. For our set $S \in V$, we pick every $v \in V$, and, for every $\{u,v\} \in E$ we pick $x_{v,u}$ if u is directed towards v, and $x_{u,v}$ if v is directed towards u. Additionally, we pick the $w(u,v)$ vertices in the edge vertex that are not adjacent to whichever vertex $x_{v,u}$ or $x_{u,v}$ we picked. Clearly, the set S has an $|V| + |E| + \sum_{e \in E} w(e)$ number of elements. Additionally, the way that we selected vertices from every edge gadget means that we can cover every arc in G' contained within every edge gadget in a way that abides by the capacity constraints of the vertices in the edge gadget; the only arcs left to be covered are the ones whose endpoints are the vertex u towards which the associated edge in G was oriented. By the fact that this orientation constitutes a circulating orientation, we have that every vertex u has exactly enough capacity (namely $\frac{1}{2}W(u)$) to cover these remaining arcs in G'. Therefore, S is a valid capacitated vertex cover.

For the converse proof, assume that a capacitated vertex cover $S \subseteq V$ exists for which $|S| = |V| + |E| + \sum_{e \in E} w(e)$. By the fact that every $v \in V$ has a 'dummy' vertex adjacent to it with capacity 0, and by the fact that every edge gadget for a given $\{u,v\}$ between vertices has $w(u,v)+1$ arcs that have to be covered (the ones that are not dashed in Figure 5.7)) we can conclude that every edge gadget for a given $e = u, v \in E$ contributes exactly $w(e)+1$ vertices to S, such that S contains exactly one of $x_{v,u}$ or $x_{u,v}$, and apart from that all the $w(e)$ vertices not adjacent to whichever of $x_{v,u}$ or $x_{u,v}$ was picked (any other selection of vertices of the edge gadget would not cover every arc in the edge gadget). We can then translate S to an orientation of G by directing the edge $\{u,v\}$ to u if $x_{v,u} \in S$, and towards v if $x_{u,v} \in S$. In such an orientation, every u has a total weighted indegree that is exactly equal to the number of edges that u has to cover in G'. This number is exactly equal to $\frac{1}{2}W(u)$, since otherwise at least one vertex v would have to cover more than its capacity $\frac{1}{2}W(v)$ number of edges in G', which contradicts our assumption. We conclude that we have found a circulating orientation on G. □

Now that we have equivalence between the two problem instances, it only remains to be proven that the reduction graph G' has bounded tree-depth with respect to G. We can construct a tree-depth spanning tree T' of G' by copying the structure of T, and expanding every $v \in V$ into v and v' so that T' has height $2 \cdot td$. Then, for every $\{u,v\} \in E$, we can suspend $x_{v,u}$ and $x_{u,v}$ beneath v and v' (assuming w.l.o.g. that v is a descendant of u in T), and attach every remaining adjacent pair of vertices in the edge gadget underneath $x_{u,v}$, so that the total height of T' becomes at most $2 \cdot td + 4$. Every vertex and every edge in G' is now accounted for in T', and so we conclude that G' has bounded tree-depth relative to G. □

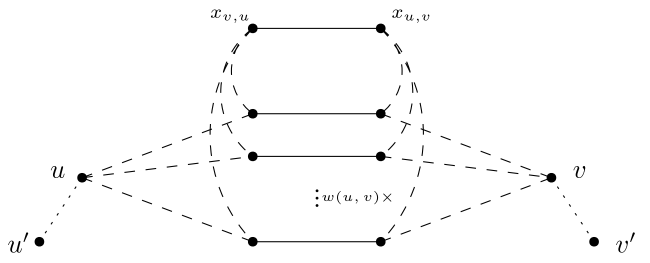

Figure 5.7: Edge gadget for the reduction graph from Circulating Orientation to Capacitated Vertex Cover. Here u' and v' both have capacity 0, u has capacity $\frac{1}{2}W(u)+1$, v has capacity $\frac{1}{2}W(v)+1$, $x_{v,u}$ and $x_{u,v}$ have the same capacity $w(u,v)+1$, and all other vertices have capacity 3.

5.7 Capacitated Vertex Cover to All-or-Nothing Flow

Theorem 5.7.1. *There is a pl-reduction from Capacitated Vertex Cover to All-or-Nothing Flow, with both problems parameterised by tree-depth.*

Proof. We begin with an instance of Capacitated Vertex Cover parameterised by tree-depth, i.e. an undirected graph $G = (V, E)$, a tree-depth spanning tree T of G with height td, a target integer k, and a capacity function $c : V \to \mathbb{N}$. We construct a flow graph G' by introducing source vertices s and s', as well as a sink vertex t, and we connect s to s' via an arc with capacity $(|V| - k)C$, where $C := \max_{v \in V} c(v)$. Then, for every $v \in V$, we add it to G' along with an additional vertex v', and we connect s' to v' via an arc with capacity C, and v' to v with capacity $c(v)$. To get rid of excess flow, we connect v' to t via an arc with capacity $C - c(v)$, and we also create $c(v)$ arcs with capacity 1 from v to t. Then, for every $\{v, w\} \in E$, we create a vertex $x_{v,w}$, and connect s to it via an arc with capacity 1. We also create arcs from $x_{v,w}$ to both v and to w with capacity 1. Figure 5.8 shows this construction.

The intuition is that a maximal flow on G' will be forced to 'choose' the $|V| - k$ vertices that are *not* part of the vertex cover. Each such vertex v will then have its entire capacity $c(v)$ directed towards it, so that it cannot receive any more flow from adjacent vertices edge $x_{v,w}$ for its neighbours w. At the same time, every $x_{v,w}$ corresponding to an edge in G must be able to get rid of its incoming flow to one of the endpoints v or w, meaning that for every edge $\{v, w\} \in E$, either one of its endpoints v or w must be included in the vertex cover.

Lemma 5.7.2. *There exists a capacitated vertex cover on G of size k given the capacity function c, if and only if an all-or-nothing flow with flow value $(|V| - k)C + |E|$ exists on G'.*

Proof. If a capacitated vertex cover S exists on G of size k, we can create a flow on G' by directing C flow from s' to v' for every $v \in V \setminus S$. If we then direct 1 flow from s to every edge vertex $x_{v,w}$, we have the desired total flow value of $(|V| - k)C + |E|$. Additionally, since every $\{v, w\} \in E$ has an endpoint in S, we can direct its inflow of 1 to that endpoint. By the fact that S is a capacitated vertex cover given c, we know that every $v \in S$ will have sufficient arc capacity between itself and t to siphon excess flow.

Conversely, if a valid all-or-nothing flow exists on G' with flow value $(|V| - k)C + |E|$, then we can take S to be the set of k vertices that were not 'picked' by s' to be in the dominating set. Since every edge $\{v, w\} \in E$ must have a corresponding $x_{v,w}$ in G' which was able to siphon its 1 flow to one of its corresponding endpoints in G, we must conclude that S is a valid vertex cover of G. Lastly, since every $v \in V$ could not receive more than $c(v)$ inflow, we can conclude that S is also a capacitated vertex cover given c. □

With Lemma 5.7.2 proven, we only have to show that G' has bounded tree-depth relative to G. To that end, we can construct a tree-depth composition T' of G' that copies the structure of T, but which expands every $v \in V$ into a path consisting of v and v'. For every edge vertex $x_{v,w}$, we can assume, w.l.o.g., that w is the ancestor of v, and suspend $x_{v,w}$ below w in T'. If we then place s, s', and t at the root of T', we have a valid tree-depth spanning tree T' of G' whose height is at most $2td + 4$. □

5.8 f-Dominating Set to All-or-Nothing Flow

Theorem 5.8.1. *There exists a pl-reduction from f-Dominating Set to All-or-Nothing Flow, with both problems parameterised by tree-depth.*

Proof. We start with an instance of f-Dominating Set parameterised by tree-depth, i.e. an undirected $G = (V, E)$ as well as a threshold function $f : V \to \mathbb{Z}_{>0}$, and a positive integer k, and a tree-depth spanning tree T of G with height t. We will now create a directed graph G' along with a capacity function $c : V \to \mathbb{N}$, on which we will define our instance of All-or-Nothing Flow.

To begin with, we create the source and sink vertices s and t, as well as an 'auxiliary' source s_D, which we connect via (s, s_D) with capacity kR, where we define $R = \max_{v \in V}(f(v) + d(v))$, with $d(v)$ being the number of neighbours that any $v \in V$ has. Then, for every vertex $v \in V$, we add v to G', and connect s to v via an arc with capacity 1, and we also connect v to t via an arc with capacity

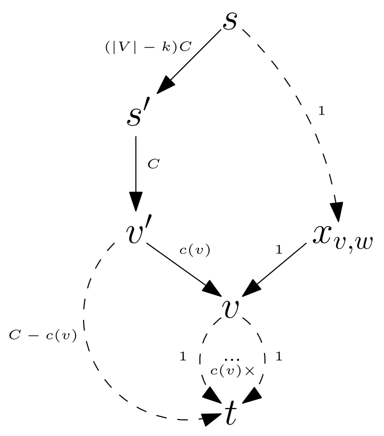

Figure 5.8: Part of the reduction graph from Capacitated Vertex Cover to All-or-Nothing Flow

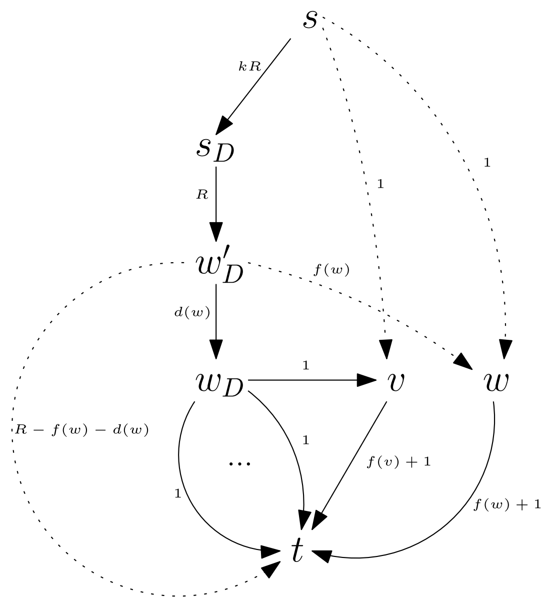

Figure 5.9: Reduction graph from f-Dominating Set to All-or-Nothing Flow.

$f(v) + 1$. Additionally, we create vertices v'_D and v_D, and we connect S_D to v'_D via an arc with capacity R, and we connect v'_D to w via an arc with capacity $f(v)$, and also we connect v'_D to v_D via an arc with capacity $d(v)$. Lastly, we create an arc from v'_D to t with capacity $R - f(v) - d(v)$ to get rid of excess flow, and we also create a $d(v)$ number of 1-capacity arcs from v_D to t.

Then, for any neighbouring pair v, w in G, we connect w_D to v via an arc with a capacity of 1.

So far, the intuition is that every vertex $v \in V$ 'needs' to receive its threshold amount of flow $f(v)$ in order to bridge the arc (v, t) which has capacity $f(v) + 1$. Via s_D, it is ensured that a k number of vertices receive their requisite inflow directly, while also receiving enough to supply every neighbour in G with one flow each. Figure 5.8.1 shows part of this construction for a adjacent pair v, w.

Lemma 5.8.2. *There exists an f-dominating set on G of size k if and only if an All-or-Nothing flow with value $kR + |V|$ exists on G' given the capacity function c.*

Proof. We first assume that the former statement holds, i.e. that a set $D \subseteq V$ exists with $|D| = k$ such that every $v \in V \setminus D$ has at least $f(v)$ neighbours in D. We begin constructing a flow g on G' by setting $g(s, v) = 1$ for every $v \in V$, and setting $g(s, s_D) = kR$. Clearly, this results in a total outflow from s of value $kR + |V|$; what remains to be done is to bring the rest of the flow network into equilibrium.

For every $u \in D$, we set $g(s_D, u'_D) = R$, which allows us to set $g(u'_D, u) = f(u)$, and in turn $g(u, t) = f(u) + 1$. We also set $g(u'_D, u_D) = d(u)$, and get rid off any excess flow by setting $g(u'_D, t) =$

$R - f(u) - d(u)$. Since u_D now has an inflow of $d(u)$, as well as $d(u)$ arcs connecting it to t, we have that every vertex in G' has achieved flow equilibrium, except for those $v \in V \setminus D$. For every such $v \in V \setminus D$, however, we know that $f(v)$ adjacent vertices w exists in D, and therefore we can set $g(w_D, v) = 1$ for every such $w \in D$, which gives v sufficient inflow such that we can set $g(v, t) = f(v) + 1$, and achieve equilibrium across the entire network G'.

For the converse proof, we assume that a flow g exists on G' which uses every arc of G' to full capacity or not at all, and which has a flow value $|g| = kR + |V|$. From the way that G' is constructed, it follows that exactly a k number of vertices v'_D receives R inflow from S_D. If we then take those k vertices to comprise the dominating set D, we can see that any $w \in V \setminus D$ has at least $f(w)$ neighbours in D, since every $w \in V \setminus D$ is able to receive $f(w)$ inflow in total from incident arcs in G'. □

From the proof of Lemma 5.8.2, we conclude that our reduction from the initial problem instance of f-Dominating Set is equivalent. What remains to be shown is that the graph G' has bounded tree-depth t' relative to t.

Note that, in G', every vertex $v \in V$ is represented by exactly three vertices v'_D, v_D and v, and that every arc in G' either has an endpoint in s or t (which we can place at the root), or has its endpoints in vertices corresponding to a pair $v, w \in V$ that is adjacent in G. A tree-depth representation T' of G' can therefore be created by copying the structure of T, and placing s, s_D, and t at the root of T', which creates a tree of height $3td + 3$. □

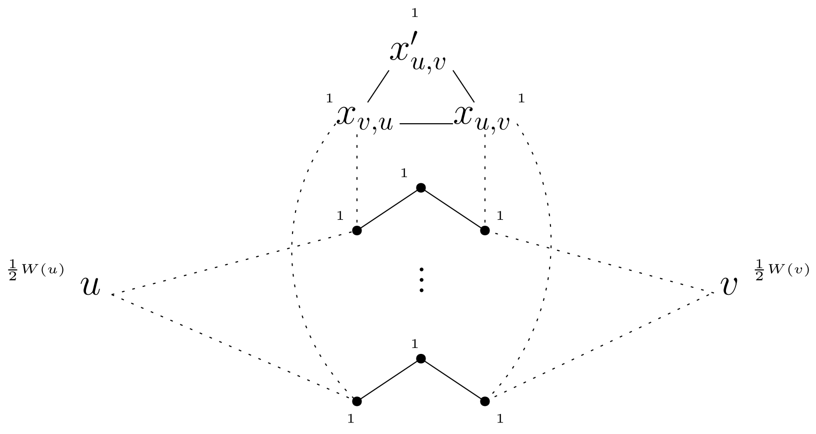

Figure 5.10: Edge gadget for the reduction graph of Circulating Orientation to f-Dominating Set

5.9 Circulating Orientation to f-Dominating Set

Theorem 5.9.1. *There exists a pl-reduction from Circulating Orientation to f-Dominating Set, with both problems parameterised by tree-depth.*

Proof. We begin with an instance of Circulating Orientation parameterised by tree-depth, i.e. an undirected graph $G = (V, E)$, and a weight function $w : E \to \mathbb{Z}_{>0}$. We will now construct an instance of f-dominating set by defining both an undirected graph $G' = (V', E')$ and a threshold function $f : V' \to \mathbb{Z}_{>0}$.

Firstly, we add every vertex $u \in V$ to G', and we set $f(u) = \frac{1}{2}W(u)$, where $W(u)$ is the sum total of weights of edges incident to u. Then, for every edge $\{u, v\} \in E$, we create two vertices $x_{v,u}$ and $x_{u,v}$, each with threshold $f(x_{v,u}) = f(x_{u,v}) = 1$, as well as a vertex $x'_{u,v}$ with threshold 1, and we connect the three total vertices in a clique. Additionally, for $i \in [1, w(u, v)]$, we create three vertices $y_{v,u,i}$, $y_{u,v,i}$ and $y'_{u,v,i}$, each with a threshold of 1, and we connect these vertices such that they form a path from u to v, with $y'_{u,v,i}$ in the middle. Additionally, we connect $x_{v,u}$ to $y_{v,u,i}$ via an edge, and $x_{u,v}$ to $y_{u,v}$ via an edge. Figure 5.9.1 shows this construction for a given edge $\{u, v\}$.

Lemma 5.9.2. *There exists a circulating orientation of G if and only if an f-dominating set D exists on G' with $|D| = |E| + \sum_{e \in E} w(e)$.*

Proof. Suppose first that a circulating orientation exists on G, that is, there exists an orientation of the edges of G such that every $u \in V$ has a total weighted outdegree of $\frac{1}{2}W(u)$, where $W(u)$ is the total weight sum of edges incident to u. We construct a dominating set D as follows: for each $\{u, v\} \in E$, if $\{u, v\}$ is oriented towards u, add $x_{v,u}$ to D, and, for every $i \in [1, w(u, v)]$, add $y_{v,u,i}$ to D. If however, $\{u, v\}$ is oriented towards v, we pick $x_{u,v}$, as well as $y_{u,v,i}$ for every $i \in [1, w(u, v)]$. Clearly, the resulting set D contains exactly $|E| + \sum_{e \in E} w(e)$ elements, and it is easy to verify that every vertex in G' whose threshold is 1 is either contained in D or has at least one neighbour in D. Additionally, we have that, for every $u \in V$, that u has exactly $w(u, v)$ neighbours in D for every neighbour v of u whose shared edge is directed towards u in the circulating orientation, which means that the total number of neighbours that u has in D is exactly $\frac{1}{2}W(u)$.

For the converse proof, we assume that an f-dominating set D exists on G with $|D| = |E| + \sum_{e \in E} w(e)$. Observe that, for every $\{u, v\}$, the triplet $x_{v,u}$, $x_{u,v}$, $x'_{u,v}$ must contribute at least one vertex to D, and that every triplet $y_{u,v,i}$, $y_{v,u,i}$, $y'_{u,v,i}$ must contribute at least one vertex to D. Given the cardinality of D, we can conclude that every such triplet contributes exactly one vertex to D. We can assume that every vertex in D is not a 'dummy' vertex of the type $x'_{u,v}$ or $y'_{u,v}$ since, if any such dummy vertex was picked for D, we can swap it out with either of the other vertices in its corresponding triplet and retain a dominating set that still has the same number of vertices. It follows that, for every $\{u, v\} \in E$, it is either the case that every vertex $x_{v,u,i}$ for every $i \in [1, w(u, v)]$ is contained in D, or it is the case that every vertex of the type $x_{u,v,i}$ is contained in D, since any other configuration of D would not actually be able to cover the entire edge gadget between u and v.

We create an orientation on G by orienting every edge $\{u, v\} \in E$ towards u if $x_{v,u} \in D$, and by orienting $\{u, v\}$ towards v if $x_{u,v} \in D$. From the fact that every $u \in V$ has at least $\frac{1}{2}W(u)$ neighbours in D, it follows that every $u \in V$ has at least $\frac{1}{2}W(u)$ total weighted indegree, and from that we can conclude that that every $u \in V$ must have exactly $\frac{1}{2}W(u)$ total weighted indegree, since, if any $u \in V$ had strictly more that $\frac{1}{2}W(u)$ weighted indegree, another $v \in V$ would necessarily have to have less than that. □

With Lemma 5.9.2 proven, all that remains to be shown is that G' has a tree-depth t' that is bounded relative to the tree-depth t of G.

Note that, in our construction of G', we retained the vertex set of G, and only added vertices for every edge $\{u, v\} \in V$. To construct a tree-depth representation T' of T', we can therefore copy the structure of T into T', and for every $\{u, v\}$ connect the edge-vertices of the type to v in G', assuming, w.l.o.g., that v is a descendant of u in T. The total height of T' only increases at most to $td + 6$ as a result. □

5.10 Target Set Selection to All-or-Nothing Flow

For the proof of the last reduction proof in the reduction graph shown in Figure 3.3, we must first introduce some new concepts that are going to be used in the sequel.

5.10.1 Graph orderings and branch orderings

Given a directed graph $H = (V, A)$, we say that a sorting s of (a subset of) the vertices V is a *topological ordering* if for any $v, w \in V$ where v precedes w in s we have that $(w, v) \neq A$.

In the following, let $G = (V, E)$ be a graph with a tree-depth spanning tree T of height t of G, and let $r \in V$ be the root of T. We will write td_v for every $v \in V$ to indicate the length of the unique path from the root of T to v (td_v can be seen as the depth of v in T). Lastly, we define $\sigma_T = (v)$ as the set containing v as well as every descendant of v in T.

We then call a function $\pi : \{\{v, w\} | v, w \in V, w \in \sigma_T(v)\} \to \{1, ..., td_w\}$ a *branch ordering* of G given T if, for any $u, v, w \in V$ where $u = v$ or u is an ancestor of v in T, and w is a direct child of v in T, we have that $\pi(u, w) = \pi(u, v)$ if $\pi(u, v) < \pi(w, w)$, and we have that $\pi(u, w) = \pi(u, v) + 1$ otherwise. Given a pair $u, v \in V$ where v is a descendant of u in T, we say that u *precedes* v in the branch ordering π if and only if $\pi(u, v) < \pi(v, v)$; otherwise, if $\pi(u, v) \geq \pi(v, v)$ we have that w precedes v in π. The intuition here is that, as will become clear through the following lemmas, we can use a branch ordering π to indicate for every ancestor-descendant pair $v, w \in V$ the index $i = \pi(v, w)$ that v has on a sorted path from the root of T to w, where the sorting is done via an ordering that is implicit on the entire graph.

Given a graph $G = (V, E)$ and a branch sorting π for a given tree-depth tree-decomposition T, we define a *branch topological ordering* as a sorting s of the vertices in V such that for any $v, w \in V$ we have that v precedes w in s if and only if we have that v precedes w in π.

With the definitions out of the way, we are ready to prove a series of lemmas.

Lemma 5.10.1. *Given a graph $G = (V, E)$ and a tree-depth spanning tree T of G, if a function $\phi : V \to \{1, ..., td\}$ is such that $\phi(v) \in [1, td_v]$ for all $v \in V$, then there exists exactly one branch ordering π of G such that $\pi(v, v) = \phi(v)$ for every $v \in V$.*

Proof. For every $u \in V$, we set $\pi(u, u) = \phi(u)$. Then, for every $u \in V$, we iterate over every v that is a direct child of u in T, and set $\pi(u, v) = \pi(v, v)$ if $\pi(u, v) < \pi(v, v)$, and $\pi(u, v) = \pi(v, v) + 1$ otherwise. Subsequently, we recursively apply the same logic to every child w of v in T to compute $\pi(u, w)$. By imagining the process visually, it is easy to see how this process is able to compute every value $\pi(u, v)$ for every ancestor-descendant pair u, v in T, resulting in a function π that abides by the criteria of being a branch ordering of G. Additionally, the process of computing π is entirely deterministic given ϕ, from which we can conclude that π is indeed unique. □

Lemma 5.10.1 essentially shows how a branch ordering π can be entirely defined by its values $\pi(v, v)$. In the following, we shall therefore refer to a function $\phi : V \to \{1, ..., td\}$ that has the property that $\phi(v) \in [1, td_v]$ for all $v \in V$ as a *reduced branch ordering* of G.

Lemma 5.10.2. *Given a graph $G = (V, E)$ with a tree-depth spanning tree T, and a sorting s of the vertices in V, there exists a branch ordering π of G given T, such that for all ancestor descendant pairs v, w in T we have that v precedes w in s if and only if v precedes w in π.*

Proof. In the following we will use ρ_v to indicate the sorted path $u_1, ..., u_{td_v}$ to indicate the td_v vertices on the unique path from r to t such that u_i precedes u_j on s for any $i, j \in [1, td_v]$ with $i < j$. Using this, we can define $\pi(u, v)$ for every ancestor-descendant pair u, v in T as the index that u has on the path ρ_v. It is easy to verify then that, for all u, v, w with $u = v$ or u an ancestor of v in T, and w a direct child of v, we have that $\pi(u, w) = \pi(u, v)$ if $\pi(u, v) < \pi(w, w)$, and $\pi(u, w) = \pi(u, v) + 1$ otherwise, since $\pi(w, w)$ represents the index of w on its own path from r to w, and so we have that w 'displaces' v by one position relative to its position on the path ρ_v if w appears before v on the path ρ_w. □

Lemma 5.10.3. *Let $G = (V, E)$ be a graph with tree-depth spanning tree T, and let pi be a branch ordering of G given T, and let $u, v \in V$ be such that u precedes v in T. Then, for any $w \in V$ where w is a descendant of both u and of v, we have that $\pi(u, w) \leq \pi(v, w)$.*

Proof. The proof follows from inductive logic, where we first observe that $\pi(u, v) \leq \pi(v, v)$ (the induction base) follows trivially from our assumption that u precedes v in π.

For the induction step, assume that $\pi(u, w) \leq \pi(v, w)$ is true for some $w \in V$ where w is a descendant of both u and of v (or, in the fringe case, $w = v$, assuming w.l.o.g. that u is an ancestor of v). Then, for any direct child w' of w, one of the following three cases must apply:

- $\pi(w', w') \leq \pi(u, w) \leq \pi(v, w)$, therefore $\pi(u, w') = \pi(u, w) + 1$ and $\pi(v, w') = \pi(v, w) + 1$;
- $\pi(u, w) < \pi(w', w') \leq \pi(v, w)$, therefore $\pi(u, w') = \pi(u, w)$ and $\pi(v, w') = \pi(v, w) + 1$;
- $\pi(u, w) \leq \pi(v, w) < \pi(w', w')$, therefore $\pi(u, w') = \pi(u, w)$ and $\pi(v, w') = \pi(v, w)$.

In all three cases, we have that $\pi(u, w') \leq \pi(v, w')$ follows. □

Lemma 5.10.4. *Given a branch ordering π of $G = (V, E)$ for a given tree-depth representation T, there exists a branch topological ordering s of G.*

Proof. Let $\{b_1, ..., b_l\}$ be the l distinct leaves of the tree T sorted in arbitrary order. For all $i \in [1, l]$ We write p_i to indicate the path from the root of T to b_i.

Claim 5.10.5. *If a topological branch ordering s_k exists for the vertices in $p_1 \cup ... \cup p_k$ for a given $k \in [1, l-1]$, then a branch topological ordering s_{k+1} exists for the vertices in $p_1 \cup ... \cup p_k \cup p_{k+1}$.*

Proof. Let $p_{k+1} = \{v_1, ..., v_m\}$, such that $v_1, ..., v_m$ is exactly the ordered path from the root of T to b_{k+1}. We initiate $s' = s_k$, and we add the vertices on p_{k+1} in order to s'. Note that to insert a vertex v_i into s' correctly, we need only to consider vertices that are already contained in s' which share the same branch as v_i in T. Note also that the following invariant holds: if s' is a branch topological sort for the vertices so far contained within it, then for any v_i with $i \in [1, m]$, if all the vertices $v_1, ..., v_{i-1}$ have already been inserted into s', we have that either v_i is already contained in s' (in which case we can skip it), or we have that v_i is not in s', but neither are any of v_i's descendants in T (if a descendant of v_i were contained in s', then v_i would necessarily have been placed in s' first). Let $u_1, ..., u_n$ be the sequence of ancestors of v_i as they appear in s'. We can now easily compute the partition $L \cup R = \{u_1, ..., u_n\}$ such that for all $u \in L$ we have that u precedes v_i in π, and that for all $u' \in R$ we have that v_i precedes u' in π. Note that for any pair u, u' where $u \in L$ and $u' \in R$, we must have that u precedes u' in π, since otherwise we would have that the contradiction $\pi(u, v_i) < \pi(v_i, v_i) \leq \pi(u', v_i) \leq \pi(u, v_i)$ holds (here the inequality $\pi(u', v_i) \leq \pi(u, v_i)$ follows from Lemma 5.10.3). We can therefore insert v_i into s' between L and R, so that s' remains a branch topological ordering of the vertices contained within it.

From the proof of the invariant it follows that we can insert all vertices $v_1, ..., v_m$ into s' and retain a topological branch ordering of the vertices within it. Once all vertices have been inserted, we let $s_{k+1} = s'$. □

From the proof of claim 5.10.5, and the fact that s_0 exists since the empty set trivially has a branch topological ordering, we can use inductive logic to conclude that a branch topological s exists that includes the vertices of every branch in T, and is therefore a branch topological ordering of the entire graph G. □

Lemma 5.10.6. *Given a branch ordering π of $G = (V, E)$ for a given tree-depth representation T, if we let $H = (V, A)$ be the directed orientation of G such that $(u, v) \in A$ if and only if $\{u, v\} \in E$ and u precedes v in π, then there exists a topological ordering s of H.*

Proof. It follows from Lemma 5.10.4 that a branch topological ordering s exists for G. Then, for any adjacent pair $v, w \in V$ in G for which v appears before w in s, we have that v must also precede w in π, which means that there cannot be an arc $(w, v) \in A$. We conclude that s is a topological ordering of H. □

5.10.2 Reduction proof of TSS to AoNF

Theorem 5.10.7. *A parameterised log-space reduction exists from Target Set Selection to All-or-Nothing Flow, where each problem is parameterised by the tree-depth parameter.*

Proof. We start with a given instance of Target Set Selection parameterised by tree-depth, i.e. an undirected graph $G = (V, E)$, a threshold function $\tau : V \to \mathbb{Z}_{>0}$, a positive integer k, and a tree-depth spanning tree T of G whose height is td.

We will construct a new directed flow-graph G' by creating vertices s and t, which represent the source and sink vertices of G' respectively. We also create an extra sink node labelled s_1, and we create an arc of capacity $|V|D$ from s to s_1, where $D := \max_{v \in V} d(v)$. We also create s_2, which we each connect to s via (s, s_2) with capacity kF, where $F := \max_{v \in V} \tau(v)$. Then, for every $v \in V$ we add to G' the vertices v_1, v_2, v_{in} and v_{out}, and we create the arcs (v_1, v_{in}), (v_2, v_{in}) and (v_{in}, v_{out}), which have capacities $c(v_1, v_{in}) = d(v)$, $c(v_2, v_{in}) = \tau(v)$, and $c(v_{in}, v_{out}) = \tau(v) + d(v)$ respectively, where $d(v)$ is the degree of v in G. Furthermore, we create an arc (v_1, t) with $c(v_1, t) = D - d(v)$ and an arc (v_2, t) with $c(v_2, t) = F - \tau(v)$, and additionally we create a $d(v) + 1$ number of arcs from v to t, whose capacities we increment such that there is an arc of capacity $\tau(v) + i$ between v and t for every $i \in [0, d(v)]$. Figure 5.11 shows this construction. Intuitively, a flow on the graph shown in Figure 5.11 must 'choose' exactly a k number of vertices via s_1 by siphoning F flow towards every vertex chosen for the initial target set. At the same time, every vertex that is not initially chosen must receive its necessary threshold flow $\tau(v)$ in order to be 'activated' by bridging the arc between v' and v. Once activated, a vertex v (through edge gadgets that we are about to define) will be able to siphon at most 1 flow to all of its neighbours, and siphon the flow it does not need directly to t.

A naive step would now be to introduce an arc (v, w) with capacity 1 for every neighbouring pair $v, w \in V$. Doing so, however, would not yield a valid reduction from target set selection. A simple way to demonstrate this is to suppose that $k = 1$, and that G is a clique of three vertices u, v, w each with threshold 2. Such a graph clearly does not have a target set of size 1, but the accompanying flow network G' would have a solution, as shown in Figure 5.15. Here we have that u is 'selected' for the initial target set by the flow network by having $d(u) + \tau(u) = 2 + 2$ flow directed towards u, while v and w both initially receive $d(v) = d(w) = 2$ inflow. Figure 5.15 shows how, given these initial values, we can reach flow equilibrium while using every arc that is used to full capacity. An intuitive explanation for the problem of our naive reduction method is that it allows for vertices v and w to activate each other in the reduction graph, even if they do not have sufficient neighbours among the previously activated vertices.

A possible solution would be to impose an arbitrary ordering on the vertices in V beforehand, and to subsequently impose restrictions on the flow network such that any vertex v could only siphon flow to a vertex w if v precedes w in said ordering. If a valid flow then existed on G', we would be certain that any vertex v has only received extra flow from the vertices that preceded it in the arbitrary ordering, which in turn would translate to a valid target set. By iterating over every such ordering, we would eventually find an ordering that works, or we would be able to conclude that no such ordering exists, and therefore that no target set exists for G of size k. We could accomplish this idea of iterating over every ordering of V relatively easily, by siphoning to every vertex a unique flow value from 1 to $|V|$. Doing so, however, would increase the tree-depth by too much, since we'd need a $|V|$ number of extra vertices v'_i per vertex $v \in V$, each of which would have to be connected to every edge gadget corresponding to every neighbour of v in G. The idea then, is to instead impose a reduced branch ordering ϕ on G, and use the mechanism of the flow network we create to derive from that an accompanying branch ordering π.

To that end, we create another sink node s_ϕ, and we connect (s, s_ϕ) with capacity $|V|$. Then, for every $v \in V$, we create ϕ_v, and for every $i \in [1, td_v]$, we create a vertex $\phi_{v,i}$, and we create an arc $(\phi_v, \phi_{v,i})$ with capacity 1. Figure 5.12 shows this construction. Intuitively, we choose a starting value $\phi(v) \in [1, td_v]$ for every $v \in V$.

In the following, for every $v \in V$, we write $\gamma(v)$ to denote the set of direct children that v has in T, and $\alpha(v)$ the set of ancestors of v in T. Lastly, we let P be a value that is sufficiently large by letting $P = \max_{v \in V}(|\gamma(v)| + |\alpha(v)| + 1)$. The choice of these numerical values will become more clear as we complete the reduction graph. In essence, they are chosen such that every vertex has the exact right amount of flow at its disposal.

To expand ϕ to a complete branch ordering π as described in Lemma 5.10.1, we consider every pair $u, v \in V$, where $u = v$ or u is an ancestor of v in T, as well as every corresponding $i \in [1, td_v]$, and we create three vertices $\pi''_{u,v,i}$, $\pi'_{u,v,i}$, and $\pi_{u,v,i}$. Intuitively, if any flow reaches $\pi_{u,v,i}$, we'll have that $\pi(u, v) = i$. The other two vertices $\pi''_{u,v,}$ and $\pi'_{u,v,}$ are there for the 'administrative' part of making sure the flow is correctly distributed throughout the network. Then in the case where $u = v$, we connect every ϕ_{u_i} to $\phi''_{u,u,i}$ via an arc with capacity 1, s to $\pi''_{u,u,i}$ with capacity P, $\pi''_{u,u,i}$ to $\pi'_{u,u,i}$ with capacity $P+1$, $\pi'_{u,u,i}$ to $\pi_{u,u,i}$ with capacity $P+1-|\gamma(u)|$, and $\pi_{u,u,i}$ to t with $\pi+1-|\gamma(u)|-|\alpha(u)|$. This construction is depicted in Figure 5.14.

Then, for all $u, v, w \in V$ where $u = v$ or u is an ancestor of v, and where w is a direct child of v, and for all $i \in [1, td_v]$, we create the construction depicted in Figure 5.14 with accompanying arc capacities. Additionally, for all $j, j' \in [1, td_w]$ where $j \geq i$ and $j' < i$, we connect $\pi_{w,w,j}$ and $\pi_{w,w,j'}$ to $\pi''_{u,w,i}$ and $\pi''u, w, j+1$ respectively via arcs, each with capacity 1. The intuition here is that, via a distribution of flow through the network, we can recursively compute every $\pi(u, w)$ as either $\pi(u, v)$ or $\pi(u, v) + 1$, where v is an ancestor of w in T, the same way as described in Lemma 5.10.1.

Lastly, we consider each edge $(v, w) \in E$. Assume, w.l.o.g., that v is an ancestor of w in T. Then, for each pair $i, j \in [1, td_w]$ for which we have that $i < j$, we create the vertices $x'_{v,w,i,j}$, and $x_{v,w,i,j}$, and we connect them between v_{out} and w_{in} as depicted in Figure 5.13. Additionally, we create the arcs $(\pi_{v,w,i}, x'_{v,w,i,j})$ and $(\pi_{w,w,j}, x'_{v,w,i,j})$, each with capacity 1. Intuitively, we have that v_{out} can only supply flow to w_{in} in G' if v precedes w in the branch ordering that we have imposed on G.

Then, for each pair $i', j' \in [1, td_w]$ for which $i' \geq j'$, we create $x'_{w,v,j',i'}$ and $x_{w,v,j',i'}$, and we connect them between w_{out} and w_{in} as depicted in Figure 5.13. Additionally, we create the arcs $(\pi_{v,w,i'}, x'_{w,v,j',i'})$ and $(\pi_{w,w,j'}, x'_{w,v,j',i'})$, each with capacity 1. Intuitively, we have that w_{out} can only supply flow to v_{in} in G' if w precedes v in the branch ordering that we have imposed on G.

Lemma 5.10.8. *There is a valid target set of size k for G with threshold function τ if and only if an All-or-Nothing flow exists on G' with value $|V|(D+1) + kF + (1 + \sum_{v \in V} |\sigma(v)|)P$, where $\sigma(v)$ is the set of descendants that v has in T.*

Proof. First we assume that a valid target set exists for G; i.e., there exists a subset $S_0 \subseteq V$ with $|S_0| = k$, as well as a partition $S_0, S_1, ..., S_l$ of non-empty, disjoint subsets of V where for every $i \in [1, l]$ we have that every $v \in s[i]$ has at least $\tau(v)$ neighbours in $S_0 \cup ... \cup S(i-1)$. By sorting the vertices of every S_i with $i \in [1, l]$ in arbitrary order, we can also state that an ordering $s = u_1, ..., u_k, v_1, ..., v_{|V|-k}$ exists such that $\{u_1, ..., u_k\} = S_0$ and $\{v_1, ..., v_{|V|-k}\} = V \setminus S_0$, and we have that for every $j \in [1, |V| - k]$, v_j has at least τ_{v_j} neighbours in the set $S_0 \cup \{v_1, ..., v_{j-1}\}$.

We can then create a flow f through G where we direct $|V|D$ from s to s_1, as well as kF from s to s_2, resulting (so far) in a flow value of $|V|D + kF$. This flow can then be siphoned off by directing the maximum possible flow towards v_1 and v_2 for every $v \in S_0$, which results in v_{in} having an inflow of $\tau(v) + d(v)$ which it can direct towards v_{out}.

From s we can generate an imposed branch ordering $\pi : V \to [1, td]$ as described in Lemma 5.10.2. We set $f(s, \pi_{u,v,i}) = P$ for every ancestor-descendant pair $u, v \in V$, where $i = \pi(u, v)$. Let $\phi : V \to \{1, ..., td\}$ be the reduced branch ordering of G for which $\phi(v) = \pi(v, v)$ for all $v \in V$. We then set $f(s, s_\phi) = |V|)$, and for every $v \in V$, we set $f(s_\phi, \phi_v) = 1$ and $f(\phi_v, \phi_{v,i}) = 1$ where $i = \phi(v)$. By looking at the structure of the graph in Figure 5.14, we can see that a vertex $\pi_{u,v,i}$ is activated by having flow directed towards it if and only if we have that $i = p(u, v)$. The flow graph G' therefore 'copies' the branch ordering π such that an activated u_{out} can give flow to a vertex v_{in} if they are adjacent, and u precedes v in the sorting s. The total flow directed out of s is now the desired $|V|(D + 1) + kF + (1 + \sum_{v \in V} |\sigma(v)|)P$. All that remains to be done is to show that the flow network can be brought into equilibrium given the flow values chosen so far.

Note that, for every $v \in V \setminus S_0$, there exist at least a $\tau(v)$ number of neighbours $u \in V$ that precede v in s. For each of these neighbours u we have, via Lemma 5.10.2, that u precedes v in π as well. Let b be the leaf of the branch of T on which u and v lie. If u is an ancestor of v in T, then we have that $\pi(u, b) = i < j = \pi(v, b)$, and if v is an ancestor of u in T, we have that $\pi(u, b) = i \leq j = \pi(v, b)$. In either case, we have that u_{out} can supply 1 flow via $x'_{u,v,i,j}$ and $x_{u,v,i,j}$ to v_{in}. It is therefore the case that every $v \in V \setminus S_0$ is able to receive its necessary $\tau(v)$ flow to v_{in} from other vertices that have already been activated.

Now we shall prove the lemma in the opposite direction. To that end, assume that a valid flow f exists on G' such that $f(a) = 0$ or $f(a) = c(a)$ for every arc in G', and which has a flow value of $|V|(D + F) + kT + P \sum_{u \in V} |\sigma_T(u)|$.

We let $S_0 = \{u_1, ..., u_k\}$ be the k number of vertices for which s_1 receives D flow from s_1.

It follows from the flow value of v that for every $v \in V$ exactly one vertex $\phi_{v,i}$ is selected by receiving flow from ϕ_v. Following the outflow of $\phi_{v,i}$ in Figure 5.14, we can see how for every pair $u, v \in V$ where $u = v$ or u is an ancestor of v in T, we have that exactly one vertex of the type $\pi_{u,v,j}$ is selected, which is analogous to creating a branch ordering $\pi : V \to \{1, ..., td\}$ as described in Lemma 5.10.1 and imposing it on G', such that a vertex u_{out} can only transmit flow to v_{in} if u and v are adjacent in G and u precedes v in π. This is analogues to turning G into a directed graph H where every $\{u, v\} \in E$ is directed towards v if and only if u precedes v in π; it follows then from Lemma 5.10.6 that a topological ordering s' exists of H, so that we can sort the remaining vertices in $V \setminus S_0$ as $v_1, ..., v_{|V|-k}$ as they appear in s'. It then follows that, in the ordered sequence $s = u_1, ..., u_k, v_1, ..., v_{|V|-k}$, we have that every v_i with $i \in [1, |V| - k]$ must have at least $\tau(v_i)$ neighbours in the vertices that precede it in s, since every vertex v_i receives its requisite $\tau(v_i)$ flow from neighbouring vertices in G', and no flow can be directed from any of the vertices v_j with $j \in [i, |V| - k]$. We conclude that S_0 is a valid target set for G given threshold function τ. □

Having proven Lemma 5.10.8, all that remains to do is to show that G' has bounded tree-depth relative to G'. To that end, we create a tree-depth spanning tree T' of G' as follows: for each sink and source node, we simply place them at the top of T' in a single path. Then, for each $v \in V$, we place the vertices v_1, v_2, v_{in}, v_{out}, and ϕ_V on a single path in arbitrary order. Additionally, we extend the path with every vertex of the type $\phi_{v,i}$, as well as every vertex of the type $\pi_{u,v,i}$, $\pi'_{u,v,i}$, and $\pi''_{u,v,i}$, where $u = v$ or u is an ancestor of v in T, and $i \in [1, td_v]$. Since v has at most a td number of ancestors and $td_v \leq t$, it follows that the chain of vertices for every v is at most a constant factor of t^2 long. This means that if we copy the structure from T onto T' by connecting the paths v and w if we have that v and w are adjacent in T, we end up in with a tree T' that has at most $\mathcal{O}(td^3)$ depth.

Suppose now that $\{v, w\} \in E$. To include the corresponding edge vertices in G' into T', we can observe in Figure 5.13 that, for every pair $i, j \in [1, td_w]$, we have that the vertices $x'_{v,w,i,j}$, $x_{v,w,i,j}$, $x'_{w,v,j,i}$, and $x_{w,v,j,i}$ are only adjacent to vertices corresponding to v, w, and b, where b is the leaf of the branch of T on which v and w lie., We can incorporate these edge vertices into T' by suspending them below the chain of vertices corresponding to b, which at most increases the height of T' by 4 across all edges $\{v, w\} \in E$.

By studying the Figures 5.11, 5.12, 5.13, and 5.14, we can see that every vertex in G' is accounted for in in T', and that every arc in G' is also accounted for with an ancestor-descendant relationship in T'. Since T' has a height that is at most $\mathcal{O}(td^3)$, we conclude that G' has bounded tree-depth relative to G. □

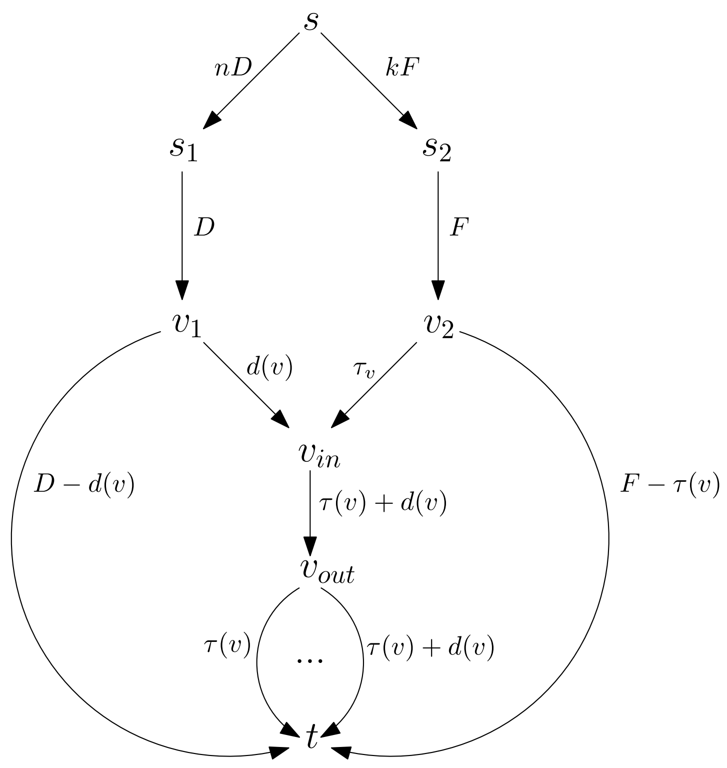

Figure 5.11: Global reduction graph from Target Set Selection to All-or-Nothing Flow

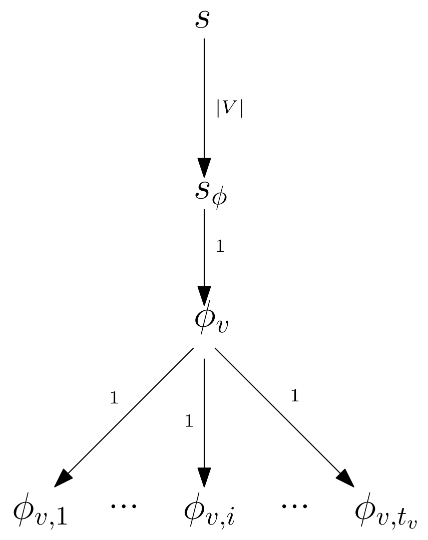

Figure 5.12: Part of the reduction graph from Target Set Selection to All-or-Nothing Flow that imposes an arbitrary reduced branch ordering on G via flow distribution.

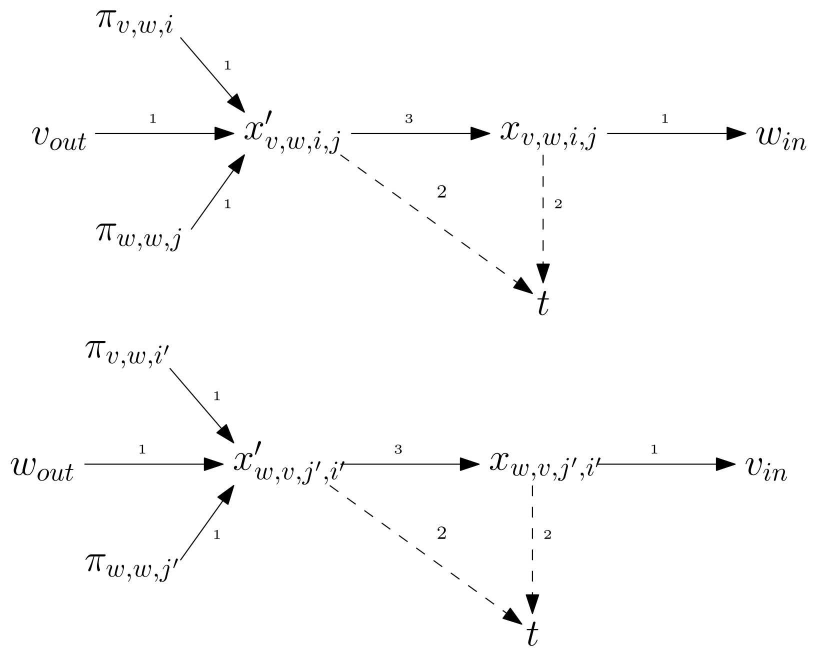

Figure 5.13: Edge gadgets in the reduction graph from Target Set Selection to All-or-Nothing Flow for $\{v, w\} \in E$, where v is an ancestor of w in the tree-depth spanning tree T. Here $i < j$ and $j' \leq i'$, and b is the leaf of the branch that v and w lie on in T.

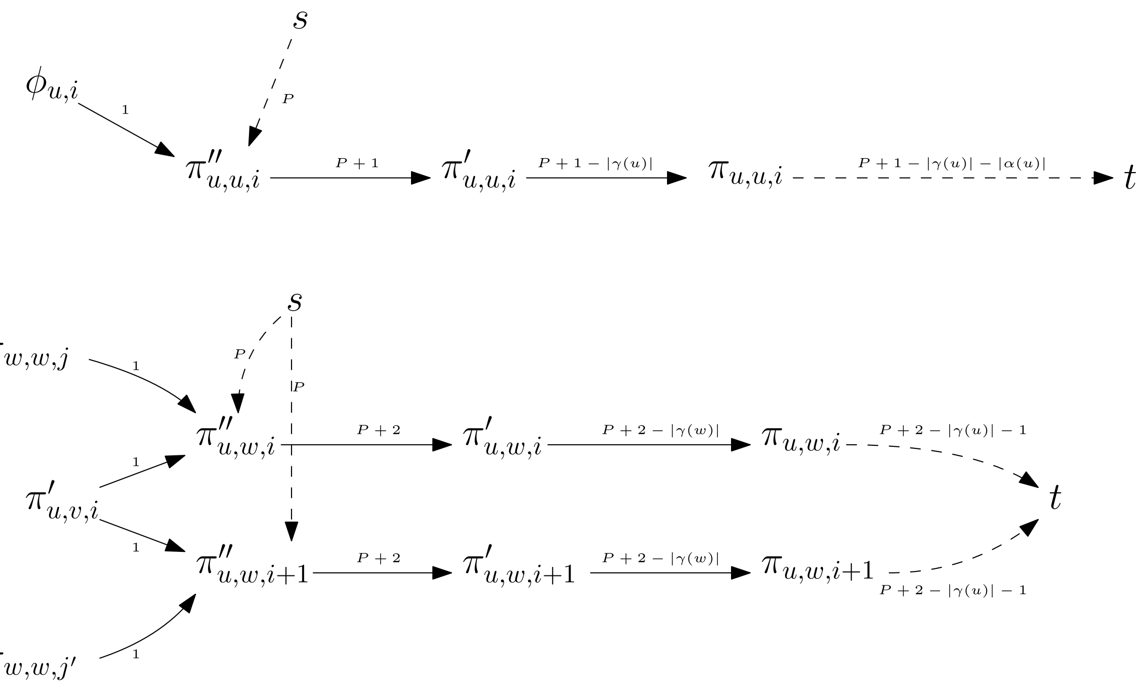

Figure 5.14: Part of the reduction graph from Target Set Selection to All-or-Nothing Flow, in which a reduced branch ordering ϕ is translated into a complete branch ordering π.

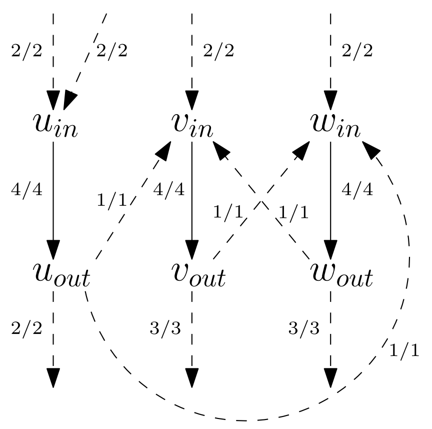

Figure 5.15: Counter example of a permissible flow on the reduction graph of an instance of TSS that has no solution when reduced to AoNF, assuming that a naive approach is used (source and sink vertices have been left implicit, as well as arcs that have no flow sent across them).

Chapter 6

Complexity of flow-like problems

6.1 W[1]-hardness for flow-like problems parameterised by tree-depth

In [11], Michael Dom et al. show that Capacitated Vertex Cover is W[1]-hard for the treewidth parameter. Subsequently, Tatsuya Gima et al. noted in [12] that the proof of W[1]-hardness of Capacitated Vertex Cover for treewidth could easily be shown to work for tree-depth as well. By combining this with our results, we can conclude that all the flow-like problems we have discussed in this paper are W[1]-hard as well when parameterised by tree-depth.

Theorem 6.1.1. *All-or-Nothing Flow, Target Outdegree Orientation, Chosen Maximum Outdegree, Minimum Maximum Outdegree, Outdegree Restricted Orientation, Circulating Orientation, Red-Blue Capacitated Dominating Set, Capacitated Dominating Set, Capacitated Vertex Cover, Undirected Flow with Lower Bounds, and Target Set Selection are W[1]-hard when parameterised by tree-depth.*

Note that we included Capacitated Vertex Cover in the result for completeness. Similarly, we included Capacitated Dominating Set, although its W[1]-hardness was also shown in [12] for the $td + k$ parameter, where td is the graph's tree-depth and k is the solution size (hence we could've also concluded W[1]-hardness for all the flow-like problems from this result).

6.2 Flow-like problems and XSLP

As mentioned in Chapter 5, we were unable to provide either an XSLP membership proof or an XSLP-hardness proof for any of the flow-like problems when parameterised by tree-depth, despite significant effort. An intuitive reason for this is that the flow-like problems defined in Section 2.9 are all, at their essence, solved by making a binary choice for every edge in the graph (with the domination-adjacent problems like Capacitated Dominating Set, it is still a binary state whether a given edge has a dominating vertex on one side). Conversely, the problems related to BinCSP all involve having to choose from an $\mathcal{O}(|V|)$ number of colours for every vertex. Our intuition is that there is a fundamental barrier between these two types of problems (selecting binary values for edges and selecting colours for vertices), such that it is impossible to provide a pl-reduction between BinCSP and one of the flow-like problems when using tree-depth as a paremetr. Since we have shown these problems to be a part of a strongly connected reduction graph in Chapter 5 (and, as such, any reduction to or from BinCSP would immediately give membership or hardness for XSLP for all problems in the graph), this suggests that the flow-like problems are part of a complexity class that is fundamentally different from XSLP. More formally, we state our intuition in the following conjecture:

Conjecture 1. *There exists a complexity class C such that the flow-like problems defined in Section 2.9 are complete for C when parameterised by tree-depth, and for which we have that C and XSLP are not subsets of each other; in other words, they are unequal such that they both contain parameterised problems which the other does not.*

Chapter 7

Conclusion

7.1 Conclusion

The end-of-paper discussion of [3] conjectures that Dominating Set and Independent Set can be proven XSLP-complete for the logarithmic tree-depth parameter. We have proven that these problem are indeed XSLP-hard for that parameter, and we have proven XSLP-hardness for several related problems when parameterised by logarithmic tree-depth, such as Vertex Cover, Roman Dominating Set, and Independent Dominating Set.

Our main results are found in Chapter 5, where we show that the flow-like problems defined in [1], as well as several other existing problems (Target Set Selection, Capacitated Vertex Cover, and f-Dominating Set) are all part of a strongly connected reduction graph when using tree-depth as the parameter for every problem, which suggests a strong cohesion between the problems in the graph in terms of hardness when parameterised by tree-depth. Furthermore, resistance to attempts to show either XSLP-hardness or XSLP membership for any of the flow-like problems suggests that such a proof is not possible, and that therefore the problems in the reduction graph are in fact in their own complexity class when parameterised by tree-depth. We also show W[1]-hardness for the flow-like problems parameterised by tree-depth in Section 6.1, which provides further insight into the position of the flow-like problems parameterised by tree-depth in the hierarchy between FPT and XP.

Our proof of the existence of a parameterised reduction from Target Set Selection to All-or-Nothing Flow we find particularly interesting, since it makes the most use of any of our reduction proofs of the tree-depth spanning tree of the given target graph.

We end our paper by enumerating a number of open questions:

1. We were not able to prove XSLP membership for the Dominating Set, Independent Set, and related problems when parameterised by logarithmic tree-depth, since a reduction from a problem which uses the logarithmic tree-depth parameter to BinCSP parameterised by ordinary tree-depth seems infeasible (one would have to construct a Gaifman graph that had a number of vertices smaller by a factor of $\log(|V|)$ relative to the instance of Dominating Set or Independent Set). However, using the machine definition of the class XSLP, it may still be possible to give a membership proof of Dominating Set and Independent Set when parameterised by tree-depth, as well as their related problems Roman Domination, Independent Dominating Set, and Vertex Cover when parameterised by tree-depth.

2. Using the concept of ordering branches relative to the tree-depth composition T of a given graph G (Section 5.10.1), might there be a way to provide reductions between other interesting problems when parameterised by tree-depth? As an example, we have tried to apply a similar technique to the concept of graph colouring, so that for every branch of T only a $\mathcal{O}(td)$ number of choices are available, where td is the height of T. The purpose of this was to attempt a tree-depth parameterised reduction from BinCSP to AoNF, and, even though we were unsuccessful, we believe it is possible that the technique may yield more success elsewhere.

Acknowledgements

The author of this paper would like to thank Prof. Hans Bodlaender for being the primary supervisor for this thesis project, as well as Dr. Erik Jan van Leeuwen for being the secondary supervisor. Additionally, the proofs of theorems 5.4.1 and 5.9.1, which provide reductions from the Circulating Orientation problem to the Target Set Selection problem and to the f-Dominating Set problem respectively, were initially discovered by Prof. Hans Bodlaender. Additionally, the reduction from Circulating Orientation to Capacitated Vertex Cover (Theorem 5.6.1) draws its inspiration from these proofs.

Bibliography

[1] Hans L. Bodlaender, Gunther Cornelissen, and Marieke van der Wegen. *Problems hard for treewidth but easy for stable gonality.* 2022. arXiv: `2202.06838 [cs.DS]`.

[2] Hans L. Bodlaender, Carla Groenland, Hugo Jacob, Marcin Pilipczuk, and Michał Pilipczuk. "On the complexity of problems on tree-structured graphs". In: *17th International Symposium on Parameterized and Exact Computation (IPEC 2022).* Vol. 249. Leibniz International Proceedings in Informatics (LIPIcs), 6:1–6:17. DOI: `10.4230/LIPIcs.IPEC.2022.6`.

[3] Hans L. Bodlaender, Carla Groenland, and Michał Pilipczuk. *Parameterized complexity of Binary CSP: vertex cover, treedepth, and related parameters.* 2023. arXiv: `2208.12543 [cs.DM]`.

[4] Hans L. Bodlaender, Carla Groenland, Jesper Nederlof, and Céline M. F. Swennenhuis. "Parameterized problems complete for nondeterministic FPT time and logarithmic space". In: *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS).* 2022, pp. 193–204. DOI: `10.1109/FOCS52979.2021.00027`.

[5] Michael Elberfeld, Christoph Stockhusen, and Till Tantau. "On the space and circuit complexity of parameterized problems: classes and completeness". In: *Algorithmica* 71 (2015), pp. 661–701. DOI: `https://doi.org/10.1007/s00453-014-9944-y`.

[6] Hans L. Bodlaender, Carla Groenland, Hugo Jacob, Lars Jaffke, and Paloma T. Lima. "XNLP-completeness for parameterized problems on graphs with a linear structure". In: *17th International Symposium on Parameterized and Exact Computation (IPEC 2022).* Vol. 249. Leibniz International Proceedings in Informatics (LIPIcs). 2022, 8:1–8:18. DOI: `10.4230/LIPIcs.IPEC.2022.8`.

[7] Jaroslav Nešetřil and Patrice Ossona de Mendez. "Tree-depth, subgraph coloring and homomorphism bounds". In: *European Journal of Combinatorics* Volume 27 (6 2006), pp. 1022–1041. DOI: `https://doi.org/10.1016/j.ejc.2005.01.010`.

[8] Ananth V. Iyer, H.Donald Ratliff, and G. Vijayan. "Optimal node ranking of trees". In: *Information Processing Letters* 28 (5 1988), pp. 225–229. DOI: `https://doi.org/10.1016/0020-0190(88)90194-9`.

[9] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* The MIT Press, 2022.

[10] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. "Known algorithms on graphs of bounded treewidth are probably optimal". In: *ACM Transactions on Algorithms* 14 (2 2018), 13:1–13:30. DOI: `https://doi.org/10.1145/3170442`.

[11] Michael Dom, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. "Capacitated domination and covering: a parameterized perspective". In: *Parameterized and and Exact Computation, Third International Workshop, IWPEC 2008.* Vol. 5018. Lecture Notes in Computer Science. 2008, pp. 78–90. DOI: `https://doi.org/10.1007/978-3-540-79723-4_9`.

[12] Tatsuya Gima, Tesshu Hanaka, Masashi Kiyomi, Yasuaki Kobayashi, and Yota Otachi. "Exploring the gap between treedepth and vertex cover through vertex integrity". In: *Theoretical Computer Science* 918 (2022), pp. 60–76. DOI: `https://doi.org/10.1016/j.tcs.2022.03.021`.