Karlsruhe Institute of Technology

On Difference-Labellings for Directed Graphs

Bachelor Thesis of

Matthew Akram

At the Department of Informatics
Institute of Theoretical Informatics

Reviewers:	T.T.-Prof. Dr. Thomas Bläsius
	PD Dr. Torsten Ueckerdt
Advisors:	Marcus Wilhelm

Time Period: 1st December 2021 – 28th February 2022

Statement of Authorship

Ich versichere wahrheitsgemäß, die Arbeit selbstständig verfasst, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde sowie die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet zu haben.

[illegible]

Karlsruhe, February 1, 2022

Abstract

A directed graph $G = (V, E)$ is a difference-digraph if there exists a labelling $\lambda : V \mapsto \mathbb{Z}$ such that the arc (v, w) is in E if and only if there exists a $z \in V$ with $\lambda(v) - \lambda(w) = \lambda(z)$. An undirected graph $G = (V, E)$ is an (integral) sum-graph if there exists a labelling $\lambda : V \mapsto \mathbb{Z}$ such that the edge $\{v, w\}$ is in E if and only if there exists a $z \in V$ with $\lambda(v) + \lambda(w) = \lambda(z)$. In this thesis, we investigate some combinatoric as well as some informatical properties of difference-digraphs. We present a complete dichotomy of which rooted forests are difference-digraphs, as well as some interesting properties that aid in the classification of other graph classes. For some graph classes, we look into the minimum number of vertices one needs to add, in order to obtain a difference-digraph with the original graph as an induced sub-graph. We also investigate the decision problem of deciding if a graph is a difference-digraph, and prove that it is in NP. Lastly, we consider the complexity of storing directed graphs as (induced sub-graphs of) difference-digraphs. We then extend our findings, and apply them to the storage of undirected graphs as induced sub-graphs of sum-graphs, by adding isolated vertices.

Deutsche Zusammenfassung

Ein gerichteter Graph $G = (V, E)$ ist ein Differenzgraph genau dann, wenn es eine Benennung $\lambda : V \mapsto \mathbb{Z}$ der Knoten gibt, sodass die Kante (v, w) in E existiert genau dann, wenn es ein $z \in V$ gibt, sodass $\lambda(v) - \lambda(w) = \lambda(z)$. Ein ungerichter Graph $G = (V, E)$ ist ein integraler Summengraph genau dann, wenn es eine Benennung $\lambda : V \mapsto \mathbb{Z}$ der Knoten gibt, sodass die Kante $\{v, w\}$ in E existiert genau dann, wenn es ein $z \in V$ gibt, sodass $\lambda(v) + \lambda(w) = \lambda(z)$. In dieser Arbeit untersuchen wir einige kombinatorische, sowie komplexitätstheoretische Eigenschaften von Differenzgraphen. Wir präsentieren eine vollständige Unterteilung, die bestimmt wann ein gewurzelter Wald ein Differenzgraph ist und weitere Eigenschaften, die es erleichtern können andere Graphklassen zu klassifizieren. Für bestimmte Graphklassen bestimmen wir die geringste Anzahl an Knoten, die man zu einem beliebigen Graphen hinzufügen muss, um einen Differenzgraphen zu erhalten, mit dem originalen Graphen als induzierten Teilgraphen. Wir untersuchen auch das Entscheidungsproblem, welches entscheidet ob ein Graph ein Differenzgraph ist und beweisen, dass es in NP liegt. Schließlich betrachten wir die Platzkomplexität der Aufgabe, Graphen als induzierte Teilgraphen von Differenzgraphen zu speichern. Wir erweitern daraufhin diese Ergebnisse, um ungerichtete Graphen als induzierte Teilgraphen von Summengraphen platzeffizient speichern zu können, indem wir isolierte Knoten hinzufügen.

Contents

1 Introduction **1**
1.0.1 Thesis Outline 2
1.1 Related Work 3

2 Preliminaries **7**
2.1 Notation 7
2.2 Difference-Digraphs 8
2.3 Some Basic Features 8
2.4 Examples 10

3 Classification **13**
3.1 A Bound on the Number of Edges 13
3.2 Lakes 14
3.3 Disjoint Union 16
3.4 Forbidden Substructures 16
3.5 Rooted Trees 18
3.5.1 Out-Spiders 23
3.5.2 Out-Trees 26
3.5.3 A Complete Dichotomy 30

4 Difference-Number **33**
4.1 Basic Difference-Numbers 34
4.2 The Difference-Number of Rooted Trees 34
4.2.1 Out-Trees 34
4.2.2 In-trees 38

5 Complexity **41**
5.1 Time Complexity 41
5.2 Space Complexity 43
5.2.1 Directed Acyclic Graphs 45
5.2.2 General Digraphs 48
5.2.3 A Remark on the Space Complexity of Sum-Labelling 50

6 Conclusion **53**

Bibliography **55**

1. Introduction

A *difference-digraph* is a directed simple graph such that there exists a labelling λ (referred to as a *difference-labelling*) from the vertex set to the integers, such that the arc (u, v) is present if and only if $\lambda(u) - \lambda(v)$ is a label assigned to at least one vertex in the graph. A *sum-graph* is defined by Frank Harary [Har90] as an undirected simple graph such that there exists an injective labelling λ (a *sum-labelling*) from the vertex set to the natural numbers, such that the edge $\{u, v\}$ is present if and only if $\lambda(u) + \lambda(v)$ is a label assigned to at least one vertex in the graph. Working with these graph classes is of particular interest, due to their combinatoric as well as number-theoretical nature. In comparison to sum-graphs, difference-digraphs have not been studied as intensively, having only been investigated in a handful of papers since their inception. We, therefore, build on the ideas of previous papers, to gain a deeper understanding of difference-digraphs.

Much like their undirected counterpart, difference-labellings pose a strong condition on the structure of the graph. The first and most commonly asked question is therefore one of classification. Aside from some simple examples (paths, transitive tournaments, etc.) there have been only two ([EG84],[Son04]) families of graphs that have been identified as difference-digraphs. There has been, however, more work invested into finding certain properties that aid in the classification of difference-digraphs. We identify a third family, by presenting a complete dichotomy of which rooted-forests are difference-digraphs, as well some structural properties and forbidden substructures that further aid in the classification of difference-digraphs. Our work, together with the work of M. Sonntag [Son04], puts us very close to a complete dichotomy of directed forests.

A large portion of the research on sum-graphs deals with finding the minimum number of isolated vertices to add to an undirected graph for it to become a sum-graph; this is referred to as the *sum-number*. Similarly, we investigate the minimum number of vertices to add to a graph to obtain a difference-digraph with the original graph as an induced sub-graph; we refer to this as the *difference-number* of a graph. It took about three years since the inception of sum-graphs for the sum-number of trees to be calculated (it is equal to one, except for the trivial tree [Ell93]). We present a tight upper bound on the difference-number of in-trees, as well as a constant upper bound on the difference-number of out-trees.

In the field of computer science, NP-complete problems are of particular interest[1]. It is conjectured, that deciding if a given graph is a sum-graph is NP-complete. Not in part due to lack of effort, an NP-completeness proof is still outstanding, although the problem is

[1]At the time of writing, the PNP-problem has not yet been solved.

known to be in NP. On the other hand, as we see throughout this thesis, difference-digraphs have more structure than sum-graphs, but do not differ by enough to make one believe that they are much easier to detect. Their more specific structure could however make a (non)-NP-completeness proof a bit more attainable. With the belief that deciding if a directed graph is a difference-digraph is an NP-complete decision problem, we show that it does lie in NP.

Continuing on the topic of complexity, a very recently proposed question is the space complexity of storing undirected graphs as induced sub-graphs of sum-graphs by adding isolated vertices. While not quite an adjacency scheme, this could still prove to be a valid method of storing certain graph classes. For example, the currently most space-efficient manner of storing graphs as induced sub-graphs of sum-graphs [FG21] is the second most space-efficient manner to implicitly (i.e. without using pointers) represent planar graphs. Similarly to the aforementioned paper, we investigate the space complexity of storing directed graphs as induced sub-graphs of difference-digraphs. Due to the similarity of the problems, we also present an algorithm that asymptotically beats the bound presented by H. Fernau and K. Gajjar [FG21] (in general graphs, and matches it for graph classes of constant degeneracy).

1.0.1 Thesis Outline

In our introduction, we give a brief outline of the historical context of our work, as well as a general overview of our results. In Section 1.1 we give a brief history on sum-graphs and a complete survey on difference-digraphs. This survey also includes our own results in more detail. In Chapter 2 we state any non-standard notation that we use throughout this thesis and then we begin formally investigating some simple properties of difference-digraphs. In Chapter 3 we present some properties of difference-digraphs that help us classify which graphs are difference-digraphs. We then conclude that chapter with a complete dichotomy of rooted forests. In Chapter 4 we present some results on representing directed graphs as induced sub-graphs of difference-digraphs and then we calculate upper bounds on the difference-number of rooted trees. In Chapter 5 we look at the time complexity of identifying difference-digraphs, as well as the space complexity of storing directed graphs as induced sub-graphs of difference-digraphs. In Chapter 6 we discuss some conjectures and unanswered questions that pique our curiosity.

1.1 Related Work

In this section, we give a short overview of related work. We briefly discuss some major results on sum and difference-labellings.

Sum-Graphs

In 1990, F. Harary introduced sum-graphs [Har90]. A clear structural property of sum-graphs is that they must be disconnected, since the vertex that receives the largest label cannot be connected to any other vertices. He also shows that the sum-number of a graph is always defined.

The definition of sum-graphs was later generalized by G. J. Chang [Cha93]. In a *strong* sum-graph the labelling need not be injective. In 1994, F. Harary then expanded the definition of sum-graphs [Har94] to include negative labels (and zero). Sum-graphs of this type are referred to as *integral*. Unlike sum-graphs, integral sum-graphs need not be disconnected.

Boland, Laskar, Turner, and Domke created a modular version of sum-graphs [BLTD90]. A graph is a *mod-sum-graph* if there exists an $x \in \mathbb{N}$ and a labelling $\lambda : V \mapsto \mathbb{Z}/x\mathbb{Z}$ such that $\{u, v\} \in E$ if and only if there exists a $w \in V : \lambda(u) + \lambda(v) = \lambda(w) \mod x$. It is clear that all mod-sum-graphs are also sum-graphs. The opposite is however untrue. For instance, they have shown that all trees on 3 or more vertices as well as all cycles on 4 or more vertices are mod-sum-graphs, but not sum-graphs.

Noga Alon and Scheinerman generalized sum-graphs [AS92], by replacing the condition $\lambda(v) + \lambda(u) = \lambda(w)$ with $f(\lambda(v), \lambda(u)) = \lambda(w)$ for some symmetric polynomial f. For a given polynomial f, they refer to graphs with an f-labelling as *f-graphs*. For any polynomial f they give an asymptotic bound on the number of f-graphs on n vertices. They also show, that for any graph, there exists a polynomial, such that the given graph is an f-graph (this polynomial can however be pretty large). Notably, they show that every graph $G = (V, E)$ can be transformed into an f-graph by adding at most $|E|$ isolated nodes. This also means, that for any graph, the sum-number is bound by the number of edges.

An incomplete list of results on sum-labellings can be found in [Gal13].

Difference-Digraphs

This work is mostly inspired by F. Harary's sum-graphs and *difference-graphs* [Har90]. He defines a difference-graph as an undirected graph $G = (V, E)$, such that there exists an injective labelling λ from the vertex set to the natural numbers, such that for any two distinct vertices v, u, $\{v, u\} \in E$ if and only if $|\lambda(v) - \lambda(u)| \in \lambda(V)$. The term difference-digraph was defined almost a decade prior by Gervacio [Ger82]. Gervacio defines a difference-digraph, as a simple directed graph $G = (V, E)$ with a surjective labelling λ to a finite set S of real numbers such that (x, y) is an arc of G if and only if x and y are distinct vertices in G and $\lambda(x) - \lambda(y) \in S$. Gervacio also introduces different classes of difference-digraphs. A difference-digraph is called *proper* if it can be labelled using strictly positive integers. A difference-digraph is called *monographic* if its labelling is bijective, meaning that no two vertices receive the same label. A difference-digraph is called *natural* if it has a labelling that uses distinct strictly positive integers. Results on difference-digraphs are scattered around the mathematical community. A lot of results are however quite difficult to locate, and a thorough summary of all results does not exist. We, therefore, give a thorough survey on difference-digraphs.

Most work that has been published on difference-digraphs works towards their classification. In trying to achieve this goal, quite a few structural properties of difference-digraphs have been discovered. Gervacio [Ger82] defines an in-pair, to be a pair of edges $(u, v), (w, v)$, such that $u \neq w$. He defines an out-pair, to be a pair of vertices $(v, u), (v, w)$ such that $u \neq w$. He shows that in a monographic difference-digraph, every in-pair contains at least one edge that is also be part of an out-pair. This condition is referred to as the *in-out-pair-condition* (IOC). It was later proven by Gervacio and Eggleton [EG84] that any difference-digraph can be labelled using integer labels. This is the reason that our definition of difference-labellings can, without loss of generality, use integer labels instead of real labels. They show that the disjoint union of graphs is a proper difference-digraph if and only if every connected component is a proper difference-digraph. We show, that the disjoint union of digraphs is a difference-digraph if and only if each digraph is a difference-digraph that can be labelled without using the label zero. They also prove that natural difference-digraphs contain no directed cycles. They show, that a digraph with a universal sink is a proper difference-digraph if and only if for any valid labelling λ the vertex set can be partitioned into subsets $V_1, \ldots, V_k$ such that $|V_1| = 1,\ \lambda(u) + \lambda(v) \in \lambda(V)$ if and only if $u \in V_i, v \in V_j$ for some $i > j$. They define a *source-join*, of two graphs G_1, G_2 as a graph G obtained by the disjoint union of G_1 and G_2, with one extra vertex s, that has exactly one arc going to G_1 and one arc going to G_2. They prove that the source-join of two proper/natural difference-digraphs is also a proper/natural difference-digraph. M. Sonntag [Son04] generalises the source-join result for proper/natural difference-digraphs. Instead of only two graphs, he generalises the results for any even number of graphs. For an induced sub-graph, we say it is *almost* a *lake*, if the only arcs that exit the sub-graph are from vertices that are sinks in the sub-graph. We show that a valid difference-labelling on a graph is also a valid difference-labelling when restricted to any sub-graph that is almost a lake. We also present two substructures that are forbidden in difference-digraphs.

There has also been a number of developments in discovering which graphs are difference-digraphs. Gervacio and Eggleton [EG84] show that not all graphs are difference-digraphs, not all difference-digraphs are proper, not all difference-digraphs are monographic, not every proper difference-digraph is monographic, and not every monographic difference-digraph is proper. More interestingly, they show that not every difference-digraph that is proper and monographic is natural. An *orientation* is a digraph obtained by orienting each edge of a complete graph in a direction. They show that the only orientations that are proper difference-digraphs are transitive tournaments. They also show that an oriented cycle is a difference-digraph if and only if it fulfills the IOC and is not a directed cycle, unless it is isomorphic to two given orientations of C_4 and C_5. The exact statement is restated by M. Sonntag in his paper titled 'Difference labelling of digraphs' [Son04]. In the same paper, Gervacio and Eggleton define an *end-source* as a source with out-degree one, an *end-sink* as a sink with out-degree one, and all other sources/sinks as *internal*. A source is *odd* if it has odd out-degree. A sink is referred to as *special* if it is adjacent to distinct sources u, v such that u is an odd source, and v is an end-source. All other sinks are referred to as *ordinary*. An alternating tree is a digraph, with a tree as its underlying undirected graph, where every vertex is either a source or a sink. They show that an alternating tree is a natural difference-digraph if and only if every odd-source is adjacent to an ordinary sink. M. Sonntag [Son04] then defines a *d-tree* as a tree $T = (V, E)$ that fulfills the IOC, and for every $v \in V : |N_{\text{out}}(v)|$ is either even or one, and if there exists a $u \in N_{\text{in}}(v)$ with $|N_{\text{out}}(u)| = 1$ then in $N_{\text{in}}(N_{\text{out}}(v))$ there are at most $N_{\text{out}}(v)/2$ vertices with out-degree one. He proves that all d-trees are proper difference-digraphs. In this thesis, we classify which rooted forests are difference-digraphs, as well as provide some lemmas and algorithms that help in the labelling of denser graphs.

There has not yet been any work done towards investigating the difference-number of digraphs. However, Eggleton and Gervacio [EG84] do prove that any digraph is an induced sub-graph of a monographic difference-digraph. We show that the difference-number of directed cycles and the graph K_n with one additional isolated vertex is bound tightly by one. We also show that the difference-number of out-trees is bound from above by two, and that the difference-number of in-trees on n vertices is tightly bound from above by $\lfloor n/2 \rfloor - 1$.

Hegde and Vasudeva [HV09a] define a *mod-difference-digraph* as a graph $G = (V, E)$ such that there exists an $m \in \mathbb{N}^+$ and an injective labelling $\lambda : V \mapsto \mathbb{Z}/m\mathbb{Z}$ such that $(u, v) \in E$ if and only if $\lambda(v) - \lambda(u) \mod m \in \lambda(V)$[2]. They also define the mod-difference-number of a graph as the minimum number of isolated vertices one must add to convert it to a mod-difference-digraph; this is an invalid definition and should be ignored. The reason for this is that if a vertex u receives a label that induces the arc (x, y), then $\lambda(y) - \lambda(x) \equiv \lambda(u)$ must hold, meaning $\lambda(y) - \lambda(u) \equiv \lambda(x)$ must hold, so u cannot be isolated. They show that all bi-directed complete digraphs and all directed cycles are mod-difference-digraphs. In a different paper [HV09b], the same authors also show that a directed acyclic graph is a proper difference-digraph if and only if the graph obtained by inverting the direction of its edges (due to the difference in definitions) is a mod-difference-digraph in an odd modulo. Sooryanarayana and Sunita [SS20] show that bi-directed complete bipartite graphs as well as in-trees where every non-leaf has exactly two children, ladder graphs and fan graphs are all mod-difference-digraphs.

[2]That is not a typing error. Their definition is opposite to the definition we have seen so far. Their results can still be converted to our definition by inverting the direction of all arcs.

2. Preliminaries

This chapter serves as a basic introduction to difference-digraphs, we present some basic examples of difference-digraphs, as well as some of their basic reoccurring features.

2.1 Notation

In this section, we state any specific graph theoretical, or number theoretical notation that we use throughout this thesis.

We denote a directed graph (*digraph*) G as a tuple (V, E) of vertices and edges (a.k.a. arcs). Throughout this thesis, we only consider simple directed graphs, meaning there are no loops and E is a set. An edge from the vertex u to the vertex v is denoted by the tuple (u, v), or simply uv for short. The *out-neighbourhood* of a vertex v (denoted $N_{\text{out}}(v)$ or $N^G_{\text{out}}(v)$ if the graph is ambiguous) is the set $\{u \mid vu \in E\}$. In contrast, the *in-neighbourhood* of a vertex v (denoted $N_{\text{in}}(v)$ or $N^G_{\text{in}}(v)$ if the graph is ambiguous) is the set $\{u \mid uv \in E\}$. The *open-neighbourhood* of v refers the set $N_{\text{out}}(v) \cup N_{\text{in}}(v)$, while the *closed-neighbourhood* of v refers to the set $N_{\text{out}}(v) \cup N_{\text{in}}(v) \cup \{v\}$.

A vertex with an empty open-neighbourhood is reffered to as *isolated.* A vertex whose out-neighbourhood is empty is reffered to as a *sink*. A sink $v \in V$, whose in-neighbourhood is $V \backslash \{v\}$ is called a *universal* sink. A vertex $v \in V$, whose in-neighbourhood is $V \backslash \{v\}$ is called a *total sink* (even though it is not necessarily a sink).

We denote that a graph S is a sub-graph of G using $S \subseteq G$. For a graph $G = (V, E)$, and a set $A \subseteq V$, we denote the sub-graph induced by A as $G[A]$. We also use $G = (v_1, \ldots, v_n)$, to denote the directed path from v_1 to v_n, and $G = (v_1, \ldots, v_n, v_1)$ to denote the directed cycle on $v_1, \ldots, v_n$. For a vertex v, we use $G - \{v\}$ to denote the graph obtained by removing v from G, and $G + \{v\}$ to denote the graph obtained by adding v to G as an isolated vertex (assuming it is not already present in G). Similarly for $v, w \in V$ we use $G + \{vw\}$ to denote the graph obtained by adding the edge vw to G (assuming it is not in G already), and $G - \{vw\}$ to denote the graph obtained by removing the edge vw from G. A *topological ordering* is an ordering σ on the vertex set of a graph, such that $uv \in E$ only if $u <_\sigma v$.

For a function $f : A \mapsto B$, and a $C \subseteq A$, we use $f(C)$ to denote $\{f(c) \mid c \in C\}$. For any numbers x, y, z we denote $x = y \mod z$ with $x \equiv_z y$. For any set of numbers S, we use $\max S$ to denote the largest element of the set S. For two sets A, B of numbers, we use $A + B$ and $A - B$ to denote $\{a + b \mid a \in A, b \in B\}$ and $\{a - b \mid a \in A, b \in B\}$ respectively.

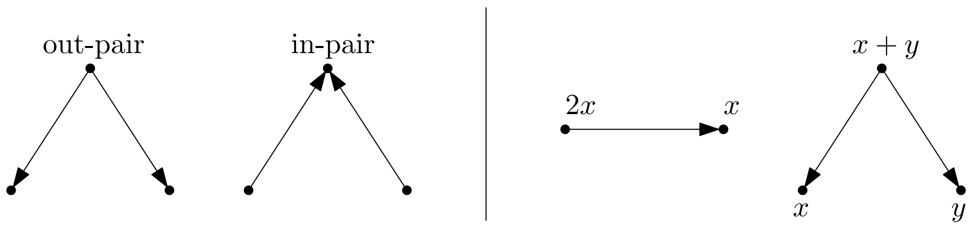

Figure 2.1: A drawing of an in/out-pair (left), and the two ways labels can induce edges (right)

2.2 Difference-Digraphs

We define a *difference-digraph* (DG), as a simple directed graph $G = (V, E)$ with a labelling λ from V to $\mathbb{Z}$, such that for any two distinct vertices $u, v \in V$ the vertex uv is in G if and only if $\lambda(u) - \lambda(v) \in \lambda(V)$. Such a labelling is reffered to as a *difference-labelling*. Over the course of this thesis we refer to a graph G as being *induced* by a difference-labelling λ, if λ is a valid difference-labelling on G. Given a vertex set V, and a difference-labelling λ on V, the distinct graph induced by λ on V is defined as

$$G := (V, \{uv \mid \exists z \in V : \lambda(u) - \lambda(v) = \lambda(z)\}).$$

There are two ways to induce an edge in a DG, represented in Figure 2.1. A pair of distinct arcs ab, cd is reffered to as an *in-pair* if $b = d$, and an *out-pair* if $a = c$, represented in Figure 2.1. For a difference-labelling λ and the vertices x, y, z, we say that x *is connected to* y *through* z if $\lambda(x) - \lambda(y) = \lambda(z)$. Given a labelling λ, we say a vertex x is *independent*, if there exist no vertices a, b such that $\lambda(a) - \lambda(b) = \lambda(x)$.

There are different classes of DGs, that propose stronger conditions on their labellings. These are presented in Table 2.1.

Table 2.1: The different classes of DGs

DGs	There exists a valid difference labelling $\lambda : V \mapsto \mathbb{Z}$.
proper DGs	There exists a valid difference labelling $\lambda : V \mapsto \mathbb{N}^+$.
monographic DGs	There exists a valid injective difference labelling $\lambda : V \mapsto \mathbb{Z}$.
natural DGs	There exists a valid injective difference labelling $\lambda : V \mapsto \mathbb{N}^+$.

2.3 Some Basic Features

One can deduce some basic but useful features of DGs by considering the features of subtraction. Firstly, we know that $\forall x, y, z \in \mathbb{Z} : x - y = z$ if and only if $x - z = y$. More specifically this means that if the vertex x is connected to y through z, then x must also be connected to z through y. This means that isolated nodes are also independent, since if the vertex with the label z is isolated, there cannot exist two assigned labels x, y such that $x - y = z$. This means that unlike sum-graphs, any non-DG cannot be converted into a DG by adding isolated vertices. As we see in Section 4.1, isolated nodes can however turn a DG into a non-DG. Moreover, if any node gets the label 0, then every node in the graph must be connected to it, since $x - 0 = x$. This also means, that two nodes with the same label are connected if and only if some node in the graph has received the label zero. Due to the nature of multiplication, we know that given any difference-labelling on a DG, we can multiply each assigned label by any non-zero factor, and our new labelling remains a valid difference-labelling. This is because for any $x, y, z \in \mathbb{Z}$, $a \neq 0$, $x - y = z$ if and only if $ax - ay = az$. The next question, would be to investigate, how changes in our definition of DGs affect the structure of the resulting graphs.

Lemma 2.1. *Two nodes u, v with different open-neighbourhoods must have different labels, unless at least one vertex receives the label zero. If that is the case, then all u, v with different closed-neighbourhoods must have different labels.*

Proof. Let V be a vertex set, and λ a difference-labelling on V. If there are two vertices $u, v \in V$ with the same label, then for all $x, y \in V$, $\lambda(u) - \lambda(x) = \lambda(y)$ if and only if $\lambda(v) - \lambda(x) = \lambda(y)$. It follows that $N_{\text{out}}(u) \backslash \{v\} = N_{\text{out}}(v) \backslash \{u\}$, regardless of any other labels. Hence, if λ assigns the label zero at least once, then $v \in N_{\text{out}}(u)$, and $u \in N_{\text{out}}(v)$, meaning that v and u have the same closed-neighbourhoods. On the other hand, if λ does not assign the label zero, then $v \notin N_{\text{out}}(u)$, and $u \notin N_{\text{out}}(v)$, meaning that v and u have the same open-neighbourhoods. □

We refer to nodes that must have different labels due to Lemma 2.1 as having *distinct relevant neighbourhoods*. Note that the relevant neighbourhoods of vertices are labelling specific. In order to show that two vertices have distinct relevant neighbourhoods, one must therefore explore the case that the labelling assigns the label zero, and the case that it does not, independently. There are however, plenty of graphs where every potentially valid difference-labelling cannot use the label zero (any graph without a total sink). There are other cases, where every potentially valid labelling must use the label zero, examples of such graphs are however not as simple[1]. In the case that we know that every valid difference-labelling must (not) use the label zero, then we can very easily distinguish what the relevant neighbourhood of any vertex is, since it remains unchanged for every valid labelling.

Lemma 2.2. *For any DG $G = (V, E)$, the graph $G' = (V', E')$ obtained from G by duplicating any node is also a DG as long as there exists a labelling λ, where every node in G receives a non-zero label. The graph G' has a difference-labelling λ' with $\lambda'(V') = \lambda(V)$.*

Proof. Let us assume that we have a DG $G := (V, E)$ with a difference-labelling λ that assigns only non-zero labels. Let us select any arbitrary node $v \in V$, and define $G' := (V', E')$ with $V' = V \cup \{v'\}$ and $E' := E \cup \{av' | av \in E\} \cup \{v'a | va \in E\}$ for a $v' \notin V$. For our newly added v' we set $\lambda'(v') := \lambda(v)$, and all other labels stay the same.

We know that for all nodes except v', λ' is still a valid difference-labelling, since no new labels were used. Since v' receives the same label as v, we know that λ' induces the same in and out-neighbourhoods for both nodes, and $vv', v'v$ are not induced, since no node receives the label zero. That means that exactly the edges of E' are induced by λ', and therefore G' is also a DG. □

It is important to keep in mind, that the labelling we choose for our new graph uses the same labels as in our original graph. This means, that if we can label a DG with exclusively non-zero labels, then we can duplicate any node an arbitrary number of times. As we see

[1]Take the graph represented in Figure 2.4, and extend an edge from w to y; all valid labellings of this graph must assign y the label zero.

in Section 3.5.1, monographic-DGs do not have this property, of being closed under vertex duplication, meaning allowing vertices to have the same labels does in-fact change the structure of the resulting graph class.

A directed cyclic graph, is a directed graph, with a directed cycle as a (not necessarily induced) graph.

Theorem 2.3 ([Ger82]). *In every cyclic DG, with a labelling λ, there exists two nodes v, w, such that* $\operatorname{sgn}(\lambda(v)) = -\operatorname{sgn}(\lambda(w))$

Proof. Given a cyclic DG $G = (V, E)$, let us assume, for the sake of contradiction, there exists a labelling λ on G, such that λ assigns labels with the same sign to all nodes in G. We know that if λ is a valid difference-labelling for G, then $\lambda' := -\lambda$ is also a valid difference-labelling for G, so we can assume w.l.o.g. that λ only assigns positive labels.

If λ assigns the label 0 to any node, then the theorem clearly holds, since $0 = -0$. We therefore assume, that the label 0 is never assigned. Let us consider the labels of a cycle $(v_1, \ldots, v_c, v_1) \subseteq G$. Letting v_i be an arbitrary node on this cycle and v_j be the next node in the cycle, we know that $v_i v_j \in E$, so that must mean that $\lambda(v_i) - \lambda(v_j) = \lambda(w)$ for some $w \in V$. However, we know that $\lambda(w)$ is strictly positive, so $\lambda(v_i)$ must be strictly greater that $\lambda(v_j)$. Inductively, we can see that $\lambda(v_1) > \lambda(v_2) > \ldots > \lambda(v_c) > \lambda(v_1)$, a contradiction. □

One could reach the same result in a more intuitive way, by noticing that any valid proper difference-labelling can be used to construct a topological ordering of the graph, simply by ordering the vertices, according to the size of their labels. This works, since a vertex with a smaller label cannot be connected to a vertex with a larger label, due to the absence of negative labels. Since the graphs that have topological orderings are exactly the DAGs, we know that no cyclic graph can have a proper difference-labelling.

It follows, that proper-DGs do not have any directed cycles. We see in Theorem 4.1, that any graph is an induced sub-graph of a (monographic) DG, meaning there exist DGs with directed cycles. The same cannot be said for proper DGs. Allowing negative labels therefore also has an effect on the structure of DGs.

2.4 Examples

A good start when working with DGs, is to take some integer sets, and find the DG that they induce. For example, a set of consecutive powers of two, induces a directed path, while a set of the powers of three induces an independent set. We now present some simple examples of DGs, after which we present some simple examples of non-DGs.

Difference-Digraphs

Given a path $G_n := (v_n, ..., v_1)$ we provide a difference-labelling for each node and prove that our labelling is valid difference-labelling. Notice here that the nodes are labelled backwards, with the root of the path receiving the label v_n. This is done purely to make our construction a bit simpler. Along with the proof, refer to Figure 2.2 for a visual representation of a valid labelling scheme for directed paths.

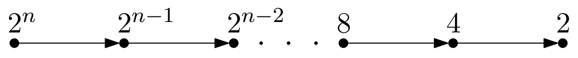

Figure 2.2: The only valid way to label a path (except for multiplication).

Theorem 2.4. *For any $n \in \mathbb{N}$ the labelling $\lambda(v_i) = 2^i$ is a valid difference-labelling on $G_n := (v_n, ..., v_1)$.*

Proof. We prove this inductively over n. The base case is clear, the set $\{4, 2\}$ induces a directed path on two vertices (the case $n = 1$ is trivial).

We know that λ is a valid difference-labelling for $v_n, \ldots, v_1$, we must now only prove that λ induces only one outgoing edge from v_{n+1} going only to v_n, and no incoming edges to v_{n+1}. We know that $\lambda(v_{n+1}) - \lambda(v_n) = 2^{n+1} - 2^n = 2^n = \lambda(v_n)$, so v_{n+1} and v_n are connected. Since v_{n+1} has the largest label, and all labels are positive, the labelling induces no edges towards v_{n+1}. For any $i \in \{n-1, ..., 1\}, 2^{n+1} > 2^{n+1} - 2^i > 2^n$, so the labelling does not induce any other outgoing edges from v_{n+1} either.

The labelling λ is thereby a valid difference-labelling for G_{n+1}, making all paths DGs. □

Moreover, one can verify, that except for multiplication, this is the only valid way to label paths.

A transitive tournament is a graph $T_n := (V_n, E_n)$, with $V_n := \{v_1, \ldots, v_n\}$ and $E_n := \{v_i v_j \mid i > j\}$. Along with the proof, refer to Figure 2.3 for a visual representation of a valid labelling scheme for transitive tournaments.

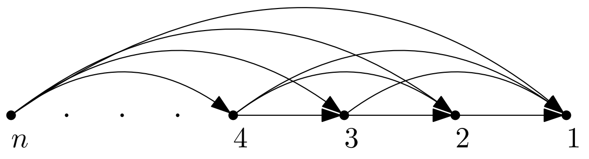

Figure 2.3: A valid labelling scheme for transitive tournaments

Theorem 2.5. *For any $n \in \mathbb{N}$ the labelling $\lambda(v_i) = i$ is a valid difference-labelling for T_n.*

Proof. We prove this inductively over n. The case $n = 1$ is not interesting, so we therefore ignore it. The base case is clear; for $n = 2$ this is a valid labelling.

We know that λ induces T_n on V_n, so λ must still be a valid difference-labelling on $T_{n+1} - \{v_{n+1}\}$. We know that for every node v_j with $j \neq n+1$, $\lambda(v_{n+1}) - \lambda(v_j) = n + 1 - j \in \lambda(V_{n+1})$, so λ induces an edge from v_{n+1} to every other node in the graph. The node with the largest label, v_{n+1}, cannot have any induced incoming edges, since all assigned labels are strictly positive. This labelling scheme therefore provides a valid difference-labelling for T_{n+1}.

Inductively, λ is a valid difference-labelling for any transitive tournament. □

Let $\overrightarrow{K}_{i,j} = (A \cup B, E)$ denote a directed complete bipartite graph, where $|A| = i, |B| = j$ and edges only go from A to B.

Theorem 2.6. *For all $i, j \in \mathbb{N}^+, \overrightarrow{K}_{i,j}$ is a DG.*

Proof. With Lemma 2.2, we know that as long as a DG can be labelled using exclusively non-zero labels, then we can duplicate nodes, and our graph remains a DG that can be labelled with exclusively non-zero labels. That means, that if $\overrightarrow{K}_{i,j}$ is a DG that can be labelled with exclusively non-zero labels, then $\overrightarrow{K}_{a,b}$, is also a DG for all $a \geq i, b \geq j$, since we can create it by duplicating nodes from $\overrightarrow{K}_{i,j}$. We must therefore only prove the $\overrightarrow{K}_{1,1}$ is a proper DG, and our theorem follows inductively.

The graph $\overrightarrow{K}_{1,1}$ is a path, so it is a proper DG by Theorem 2.4. It follows that all complete directed bipartite graphs are (proper) DGs. □

Non-Difference-Digraphs

It might seem intuitively clear, that not all graphs are DGs, it is however still useful to look at some examples. Consider the graph represented in Figure 2.4

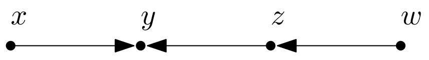

Figure 2.4: A graph that is not a difference graph.

In a DG, nodes with the same label must have the same outgoing and incoming neighbourhoods as each other. We show that the labels of x and z must be the same. However, w is in the incoming neighbourhood of z, but not in that of x, meaning that this graph cannot be a DG. Let λ be a difference-labelling on the given graph. We know that no node can have the label zero, since no node has in-degree three. In general, if a labelling λ does not assign the label zero, then for any vertex a with $N_{\text{out}}(a) = \{b\}$, we know that $\lambda(a) - \lambda(b)$ must equal $\lambda(b)$. It follows that $\lambda(x) = 2\lambda(y) = \lambda(z)$. This graph is therefore not a DG.

A *directed cycle* is a directed graph defined as $C_n = (v_1, \dots, v_n, v_1)$. Let us now consider the family of directed cycles. It is clear, that C_1 and C_2 are DGs. For all other n, this is not the case.

Theorem 2.7. *All directed cycles except for C_1 and C_2 are not DGs.*

Proof. We know for any two consecutive vertices v, u, that for any valid labelling $\lambda(v) - \lambda(u) = \lambda(u)$, since v only has out-degree one. This implies that for any difference-labelling on a C_n, $|\lambda(v_1)| > \dots > |\lambda(v_n)| > |\lambda(v_1)|$ $(n > 2)$, a contradiction. □

Throughout this thesis, we see some more complicated examples of graphs that are not DGs. In the meantime, these simple examples are sufficient to gain an intuitive understanding of some more complex properties of DGs.

3. Classification

One of the most commonly asked questions, when studying a graph class, is identifying which graphs fall into this graph class. In this chapter, we identify characteristics of DGs that allow us to more easily detect which graphs are (not) DGs. Subsequently, we also give a complete dichotomy of which rooted trees are DGs.

3.1 A Bound on the Number of Edges

In this section, we present a bound on the density of DGs. Since complete graphs are DGs (assign the label zero to all vertices), one would assume that DGs can be arbitrarily dense. While there is no bound on the number of edges of DGs in terms of the number of vertices, we can give a bound on the number of edges of a DG in relation to the maximum number of distinct labels that a valid difference-labelling can assign.

Lemma 3.1. *For any monographic DG with n nodes and m edges, that can be labelled without using two labels a, b such that $a = 2b$,*

$$\frac{m}{2} \leq \left\lceil 3(n-1)^2/8 \right\rceil + \lfloor (n-1)/2 \rfloor .$$

Proof. For any $n \in \mathbb{N}$, Tiwari and Tripathi [TT13] show that there exist integral sum-graphs with m edges if and only if $m \leq \lceil 3(n-1)^2/8 \rceil + \lfloor (n-1)/2 \rfloor$. We assume that monographic DGs can be arbitrarily dense and show that from arbitrarily dense monographic DGs one could construct arbitrarily dense integral sum-graphs, in order to reach a contradiction.

Let us assume we have a monographic DG $G := (V, E)$ with n nodes and m edges, with a difference-labelling λ that assigns each vertex a distinct label and no labels a, b, such that $a = 2b$. We define a sum-graph $G' := (V, E')$ implicitly by labelling the nodes. Notice that the vertex set of G' is V. Let us interpret λ as a a sum-labelling on G'. Remember, that G is a directed graph, while G' is an undirected graph. We must prove that G' has at least $\frac{m}{2}$ induced edges.

For every node $v \in V$ and every arc $xv \in E$, we then know that there must exist exactly one other node u with $\lambda(u) + \lambda(v) = \lambda(x)$. We know that u is not v, since there exist no labels a, b with $a = 2b$. This means that the presence of the arc xv in G, forces the presence of the edge $\{u, v\}$ in G'. This means that $|\mathrm{N}^{G'}(v)| \geq |N_{\text{in}}^{G}(v)|$ for all nodes v in G. With the degree sum formulas for directed and undirected graphs, it follows directly that

$$m = \sum_{v \in V} |N_{\text{in}}^{G}(v)| \leq \sum_{v \in V} |\mathrm{N}^{G'}(v)| = 2|E'|.$$

Let us now assume of the sake of contraposition, that $\frac{m}{2} > \lceil 3(n-1)^2/8 \rceil + \lfloor (n-1)/2 \rfloor$. It would follow, that

$$|E'| \geq \frac{m}{2} > \left\lceil 3(n-1)^2/8 \right\rceil + \lfloor (n-1)/2 \rfloor,$$

a contradiction, since G' is a sum-graph due to its construction. The lemma therefore holds for monographic DGs. ☐

Corollary 3.2. *For any monographic DG with n nodes and m edges*

$$\frac{m}{2} \leq \left\lceil 3(n-1)^2/8 \right\rceil + \lfloor (n-1)/2 \rfloor + n.$$

Proof. We now consider monographic DGs where any labelling forces the existence of two labels a, b such that $a = 2b$ (such as paths), so we must modify our algorithm for generating G' slightly. Assume we have a DG $G = (V, E)$ with m edges and n nodes and a difference-labelling λ. We then create a $G' := (V, E')$ by removing all edges from G that are induced by exactly two labels (edges induced by labels of the form $a = 2b$). The graph G' is not necessarily a DG, however generating a $G'' := (V, E'')$ from G' as in our proof of Lemma 3.1 does give us a sum-graph with $|E''| \geq |E'|/2$. Since there can only be at most n edges induced by exactly two labels, the corollary follows directly from Lemma 3.1 ☐

This bound is represented in Figure 3.1. For non-monographic DGs, this bound does not hold. However, by contracting vertices with the same label, we can create a monographic DG from any DG. This results in the following theorem

Theorem 3.3. *For a DG $G = (V, E)$ on n vertices, let λ be a valid difference-labelling on G that uses the maximal number of labels, and let m' denote the number of edges in the graph obtained by contracting vertices with the same label, then*

$$\frac{m'}{2} \leq \left\lceil 3(|\lambda(V)|-1)^2/8 \right\rceil + \lfloor (|\lambda(V)|-1)/2 \rfloor + |\lambda(V)|.$$

3.2 Lakes

We define a *lake* as an induced sub-graph, such that no edge exits the sub-graph. I.e. for a graph $G := (V, E)$ an induced sub-graph $G[A] \subseteq G$ is a lake if and only if for all $a \in A, av \in E$ implies that $v \in A$.

Theorem 3.4 (The lakes condition)**.** *If a given graph G contains a lake, that is not a DG, then G cannot be a DG.*

Proof. Let us assume that we have a valid difference-labelling λ for $G = (V, E)$, and that there exists an $A \subseteq V$, such that $G[A]$ is a lake, and is not a DG. Our goal is to show that λ is also a valid difference-labelling for $G[A] = (A, E')$. One thing is clear, since $A \subseteq V$, then $\lambda(A) \subseteq \lambda(V)$. This means that λ cannot induce any edges on A, that do not exist in G.

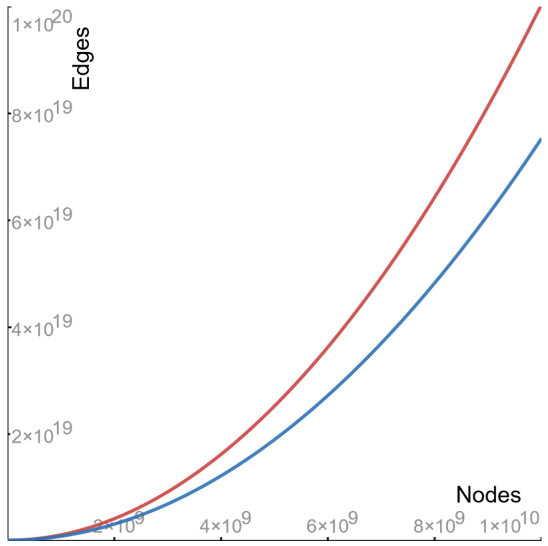

Figure 3.1: Bound on the number of edges in a monographic DG (blue) and the maximum number of edges in a directed graph (red).

Let us assume, for the sake of contradiction, that there exist an $a, b \in A, c \in V \backslash A$ such that $\lambda(a) - \lambda(b) = \lambda(c)$, meaning there exists an edge (ab) that is induced by λ on V, but not on A. The labelling λ induces the edges ab, ac, which are edges that must be in E, since G is a DG. Since $a \in A$ and $c \notin A$ then that would mean that $G[A]$ is not a lake, a contradiction.

In layman's terms we have just shown, that if λ induces an edge in E' on V, then it must also induce that edge on A. It follows, that $G[A]$ is a sub-graph of the graph induced by λ on A. Since λ cannot induce any edges on A that do not exist in G, then $G[A]$ is exactly the graph induced by λ on A. □

This theorem is a lot stronger than it might initially seem. For example, it follows directly from Theorem 3.4 that the disjoint union of graphs is not a DG, if at least one of the graphs is not a DG, since a connected component is also a lake. Nevertheless, large lakes are rare structures in directed graphs, so we also present an even stronger condition.

We refer to an induced sub-graph $G[A]$ of a graph $G = (V, E)$ as being *almost* a lake if and only if it is a lake when all sinks in $G[A]$ are removed. In other words,

$$\text{for all } a \in A : av \in E \text{ only if } v \in A \vee a \text{ is a sink in } G[A].$$

Theorem 3.5 (The strong lakes condition)**.** *If a graph $G = (V, E)$ is a DG, then every induced sub-graph $G[A]$ that is almost a lake in G is also a DG.*

Proof. Our proof is analogous to the proof of the lakes condition. Assume there is a valid difference-labelling λ for G. Given an $A \subseteq V$ such that $G[A]$ is almost a lake in G, we show that λ is a valid difference-labelling on $G[A]$.

We know that λ cannot induce any edges on A that do not exist in G, since $\lambda(A) \subseteq \lambda(V)$. Let us assume for the sake of contraposition that there exist an $a, b \in A, c \in V \backslash A$, such that $\lambda(a) - \lambda(b) = \lambda(c)$. Since G is a DG, the edges ab, ac must be in E. It follows that a must be a sink in $G[A]$, meaning that $b \notin A$, a contradiction. It follows that every edge in $G[A]$ that is induced by λ on V, is also induced by λ on A. □

3.3 Disjoint Union

In this section we examine, when the disjoint union of two graphs is a DG. By Theorem 3.4, we know that if at least one of the two graphs is not a DG, then their disjoint union cannot be a DG. We must therefore only focus on the case that the two graphs are DGs.

Theorem 3.6. *The disjoint union of two non-empty DGs, is a DG if and only if both DGs have a labelling that does not use the label zero.*

Proof. By Theorem 3.4 we know that if there exists a difference-labelling for a given graph, then the given labelling is a valid difference-labelling of all lakes in the graph. If we assume that there exist graphs G_1, G_2 and that w.l.o.g. G_1 cannot be labelled without using non-zero labels, then $G_1 \dot\cup G_2$ can also not be labelled without using non-zero labels. The disjoint union of two non empty graphs cannot have a vertex that receives the label zero, meaning that $G_1 \dot\cup G_2$ cannot be a DG.

Let us assume we are given graphs $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ and labellings λ_1, λ_2 for G_1, G_2 respectively, that do not use the label zero. We can find the label in G_2 with the largest absolute value, *max*, and create the labelling

$$\lambda(v) = \begin{cases} 2(|max| + 1)\lambda_1(v), & \text{if } v \in V_1 \\ \lambda_2(v), & \text{if } v \in V_2. \end{cases}$$

This labelling induces G_1 on V_1 and G_2 on V_2, and cannot induce any edges between G_1 and G_2. That is because the sum of any two labels in G_1 has an absolute value too large to be in G_2, the sum of two labels assigned in G_2 has an absolute value too small to be in G_1, and the sum of any label in G_1 with a label in G_2 has an absolute value too large to be in G_2 or is strictly between two labels assigned in G_1 (or has an absolute value too large to have been assigned in G_1).

The given labelling is therefore a valid difference-labelling for $G_1 \dot\cup G_2$. ☐

From this we can directly imply the following corollary.

Corollary 3.7. *The disjoint union of graphs is a DG if and only if all the graphs are DGs that can be labelled without using the label zero.*

3.4 Forbidden Substructures

Our goal in this section is to find forbidden substructures in DGs. There are a few major reasons we could come up with, as to why a graph would not be a DG (this is by no means a comprehensive list).

1. A valid labelling must assign the same label to two vertices with different relevant neighbourhoods.
2. A valid labelling on the graph must have two or three nodes v_1, v_2, v_3 with $\lambda(v_1) - \lambda(v_2) = \lambda(v_3)$ with one or both of v_1v_2, v_1v_3 not being in our edge set.

We present two substructures that fulfill the first and second conditions respectively. Our substructures are by no means the only substructures that fulfill these conditions, but in order to avoid listing a possibly infinite list of forbidden substructures, we only present the substructures, that we later require.

Odd Matching Theorem

Here we present a substructure that fulfills the first condition. We refer to a vertex as *odd*, if it has odd out-degree, and all the vertices in its out-neighbourhood have pairwise distinct relevant neighbourhoods.

Lemma 3.8. *For any valid labelling λ, and any odd vertex v, there exists exactly one vertex u with $\lambda(v) = 2\lambda(u)$.*

Proof. Let us consider an arbitrary vertex u. If $\lambda(v) = 2\lambda(u)$, then we know, that $\lambda(v) - \lambda(u) = \lambda(u)$, meaning that $u \in N_{\text{out}}(v)$. Since labels in $N_{\text{out}}(v)$ are distinct with Lemma 2.1, u is the only vertex with that label. It follows that there is at most one vertex u with $\lambda(v) = 2\lambda(u)$.

If $u \in N_{\text{out}}(v)$, and $\lambda(v) \neq 2\lambda(u)$, then we know that there exists (due to the labels in $N_{\text{out}}(u)$ being distinct) exactly one $w \in N_{\text{out}}(v)$, with $\lambda(u) + \lambda(w) = \lambda(v)$. If there were another u', with $\lambda(u') + \lambda(w) = \lambda(v)$, then u' and u, must have the same labels, which cannot be, since u' would also have to be in $N_{\text{out}}(v)$. It follows, that each vertex $u \in N_{\text{out}}(v)$ with $\lambda(v) \neq 2\lambda(u)$ must 'pair up' with an other vertex $w \in N_{\text{out}}(v)$ with $\lambda(v) \neq 2\lambda(w)$. There is therefore one vertex $x \in N_{\text{out}}(v)$, with no other vertex in $N_{\text{out}}(v)$, with which it can be paired up. It follows that $\lambda(v) = 2\lambda(x)$. □

Theorem 3.9. *For a graph $G = (V, E)$, without a total sink, let $A \subset V$ be an arbitrary set of odd vertices, with pairwise distinct relevant neighbourhoods, such that A forms an independent set in G. If the underlying undirected bipartite graph obtained by considering $G[A \cup N_{\text{out}}(A)]$ and disregarding all edges starting in $N_{\text{out}}(A)$ does not have a matching that matches all of A, then G is not a DG.*

Proof. For a graph $G = (V, E)$, let $A \subset V$ be an arbitrary set of odd vertices, with pairwise distinct relevant neighbourhoods, such that A forms an independent set in G. If the underlying undirected bipartite graph obtained by considering $G[A \cup N_{\text{out}}(A)]$ and disregarding all edges starting in $N_{\text{out}}(A)$ does not have a matching that matches all of A, then by Hall's marriage theorem, we know that there exists a subset $A' \subseteq A$ with $|N_{\text{out}}(A')| < |A'|$. Since no valid labelling can use the label zero, we know by Lemma 3.8, that every vertex $a \in A$, requires a vertex $v \in N_{\text{out}}(a)$, with $\lambda(a) = 2\lambda(v)$. There must therefore be $a, a' \in A, v \in N_{\text{out}}(A)$ with $\lambda(a) = 2\lambda(v)$, and $\lambda(a') = 2\lambda(v)$. It follows, that $\lambda(a) = \lambda(a')$, a contradiction, since labels in A need to be distinct. □

This is a generalisation of why the graph in Figure 2.4 is not a DG. We refer to a graph that has such a substructure as a sub-graph that is almost a lake as an *Odd Matching-less Graph* (OMG).

Odd-Out-Problematic Structures

Our goal is to construct a graph, that fulfills the second condition. We create a very specific graph, where some nodes are odd. We then prove, that any labelling on this graph induces an unwanted edge. For a vertex v we call a vertex u *problematic with respect to* v if all the following conditions are held.

- The vertex v is odd.
- The vertices $N_{\text{out}}(v)$ form an independent set.
- $v \notin N_{\text{out}}(N_{\text{out}}(v))$.
- The vertex u is odd.
- $N_{\text{in}}(N_{\text{out}}(u)) \cap N_{\text{out}}(v) = \{u\}$.

Theorem 3.10. *For any graph G without a total sink, if there exists a vertex v with at least*

$$\frac{|N_{\text{out}}(v)| + 1}{2} + 1$$

nodes from $N_{\text{out}}(v)$ being problematic with respect to v, then G is not a DG.

Proof. Let λ be a labelling for G. By Lemma 3.8 and Lemma 2.1 we know that the labels of vertices in $N_{\text{out}}(v)$ must 'pair up'; with the sums of their labels being equal to the label of v. Since v is odd, one of its children is left without a partner and is forced to take the label $\frac{\lambda(v)}{2}$ (It is completely irrelevant which node in $N_{\text{out}}(v)$ takes this label). Let us refer to the node with this label as z. Now, with the pigeon hole principle, one can see that there must exist two vertices x, y in $N_{\text{out}}(v)$ that are problematic with respect to v, with $\lambda(x) + \lambda(y) = \lambda(v)$. This is because, there are at least two more problematic vertices than non-problematic vertices. With Lemma 2.1, we know that x has a child x' with $\lambda(x') = \frac{\lambda(x)}{2}$. Similarly for y, we know that y has a child y' with $\lambda(y') = \frac{\lambda(y)}{2}$. However, this must mean that

$$\lambda(x') + \lambda(y') = \frac{\lambda(x) + \lambda(y)}{2} = \frac{\lambda(v)}{2} = \lambda(z).$$

The edges zx', zy' are therefore induced by λ, but are not in the edge set of G since x and y were chosen to be different problematic nodes, a contradiction.[1] $\square$

We refer to a graph that has such a substructure as a sub-graph that is almost a lake as an *odd-out-problematic-structure* (OOPS).

3.5 Rooted Trees

A *directed tree* is a directed acyclic graph, such that the underlying undirected graph is a tree. An *out-tree*, is a directed tree with a unique source, called the *root*. An *in-tree*, is a directed tree with a unique sink, called the *root*. A *rooted* tree, is a directed tree that is either an out-tree or an in-tree. In this section, we provide conditions that are necessary and sufficient for a rooted tree to be a DG, as well as algorithms for labelling such trees. To start off, we present algorithms for the labelling of specific sub-classes of rooted trees. These algorithms display the basic concepts of how we aim to label rooted trees that are DGs, and also provide much better bounds on label sizes.

[1] Notice now how our construction from Theorem 4.1 ensures that every node has even out-degree, in order to avoid odd vertices.

A Labelling Scheme for Even-Degree Out-Trees

Let $G := (V, E)$ be an out-tree such that for all $v \in V, |N_{\text{out}}(v)|$ is even. Our goal is to prove that such a graph is a DG. We do this by providing an algorithm for generating a valid labelling λ for G. We label the graph from the top down, starting with the root. Our algorithm goes as follows.

Algorithm 1: Labelling Even-Degree Out-Trees

```
1. Assign an arbitrary positive label to the root (e.g. 1).
2. Let $max := \max\{|\lambda(v)| \mid v \in V, v \text{ is labelled}\}$.
3. Select two arbitrary unlabelled nodes $v_i, v_j$, with a common labelled parent $p$.
4. Set $\lambda(v_i) := 4max$, and $\lambda(v_j) := \lambda(p) - \lambda(v_i)$.
5. Return to step 2, until all nodes are labelled.
```

Claim. Algorithm 1 provides a valid difference-labelling for even degree out-trees.

Proof. Labelling the root does not cause any unwanted edges. For all other vertices, we are going to prove that their labels are valid inductively over the number of completed iterations. Keep in mind that this labelling scheme never uses the label 0. Labelling two children in step 3, creates the edges pv_i, pv_j. All that is left to show is that labelling two nodes v_i, v_j does not create any other edges.

It is clear that setting $\lambda(v_i) := 4max$ means that for any other label $x, \lambda(v_i) - x > max$. This cannot cause any collisions since the only label larger than max is $\lambda(v_i)$, and $x \neq 0$. We also know that $x - \lambda(v_i) < -max$. The only label that is that small is $\lambda(v_j)$ and $x - \lambda(v_i) = \lambda(v_j)$ would imply that $x = \lambda(p)$. There are therefore no collisions involving the labelling of v_i, since assigned labels are clearly distinct.

It still possible that there is a label $x \neq \lambda(v_i)$ that creates an undesired edge with $\lambda(v_j)$. We know that $\lambda(v_j) = \lambda(p) - 4max \leq max - 4max = -3max$. This means that $|x - \lambda(v_j)| \geq |-max + 3max| = 2max$. There are no labels that large other than $\lambda(v_i)$, and we have already proven that the labelling of v_i does not take part in any collisions. This means that neither the labelling of v_i or that of v_j, induce any edges in the graph, other than pv_i, pv_j. It follows inductively, that the given algorithm provides a valid difference-labelling of G. ☐

Algorithm 1 provides a valid labelling of out-trees, where every vertex has even out-degree, and uses labels in $O(2^n)$. This type of graph is very easy to label, since we never have to use two labels a, b, such that $a = 2b$. We see, that when such labels are forced, the difficulty of labelling trees can increase greatly.

A Labelling Scheme for Binary Out-Trees

A *binary* out-tree is an out-tree such that for all vertices v, $|N_{\text{out}}(v)| \in \{0, 1, 2\}$. In this section, we show that all binary out-trees are (natural) DGs. This is meant to demonstrate, how we can label a graph where every valid difference-labelling must have two labels a, b with $a = 2b$, as well as providing a much better bound on label sizes in comparison to general out-trees. Throughout this section we use, that any out-tree has a straight-line planar drawing, with the root at the top and the leaves lying on the same horizontal line, as in Figure 3.2.

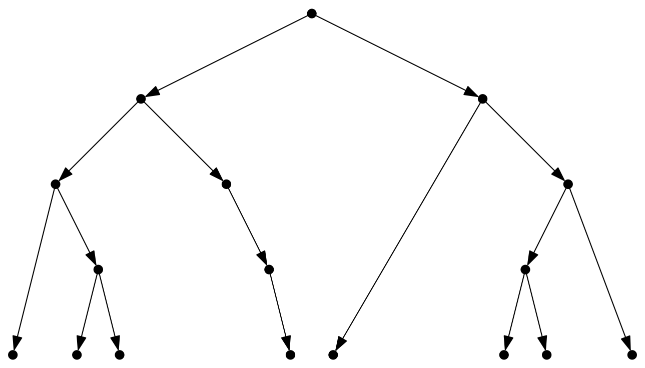

Figure 3.2: A straight-line planar drawing of a binary tree, with all the leaves on the same horizontal line.

If binary out-trees are in fact DGs, then for every binary out-tree, there exists a valid difference-labelling that does not use the label zero, since the only binary out-tree with a total sink is also a path (the binary tree on two vertices). We therefore only focus on labellings that do not use the label zero. Consider a vertex v in a binary out-tree with at least one child. If v has exactly one child a, then we know that $\lambda(v) = 2\lambda(a)$ for every valid labelling λ. If v has two children a, b, then we know that $\lambda(v) = \lambda(a) + \lambda(b)$ for every valid labelling λ. We therefore only explicitly label the leaves of the tree, and this forces the labels of all other vertices in the tree.

Algorithm 2: Labelling Binary Out-Trees

Given a binary out-tree T

1. Find a planar drawing of T, such that all leaves lie on the same horizontal line. This defines an ordering on the leaves $\ell_1, \dots, \ell_k$ (from left to right).
2. Start by labelling ℓ_1 as an arbitrary strictly positive integer (e.g. 1).
3. This new label, forces the labels of some unlabelled nodes above it. To each unlabelled non-leaf, whose label is now uniquely defined, assign it its label.
4. Select the next leaf if it exists.
5. Let *max* denote the value of the largest already assigned label in the graph.
6. Set the label of the selected leaf as $2max + 1$.
7. Return to step 3, until all nodes are labelled.

Claim. Algorithm 2 provides a valid natural difference-labelling for binary out-trees.

Proof. This algorithm only uses strictly positive labels. For every vertex v, let $t(v)$ denote the vertex set of the largest induced sub-tree in T with v as the root. By looking closely at our algorithm, one can see that the vertex v gets labelled only when all other vertices in $t(v)$ are labelled. We prove inductively that v receives the largest label in $t(v)$. If v, is a leaf, then this must be true. If v has two children a, b, then its label is the sum of both its children's labels, which are inductively the largest labels in $t(a)$ and $t(b)$ respectively. Since the label assigned to v is then larger than those assigned to a and b, and $t(v) = \{v\} \cup t(a) \cup t(b)$, we know that v receives the largest label in $t(v)$. The case that v has one child can be dealt with analogously. We can imply two things from this. If there exist two vertices v, w where

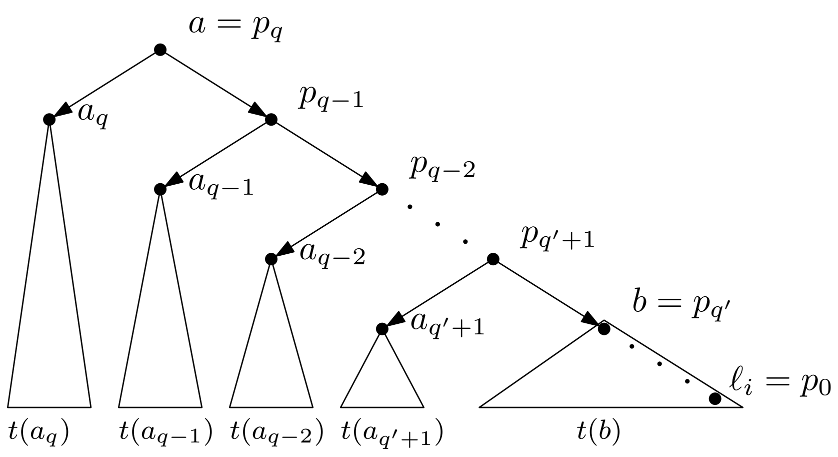

Figure 3.3: Visual aid for the proof of the correctness of Algorithm 2.

the rightmost leaf in $t(v)$ lies to the left of the rightmost leaf of $t(w)$, then we know, since v gets labelled as soon as $t(v)$'s rightmost leaf is labelled, that $t(w)$'s rightmost label is labelled after v. This means, due to steps four and five, that w's label must be strictly larger than v's label. It also follows that labels assigned by our algorithm are strictly increasing, and therefore distinct.

We must now prove that Algorithm 2 produces a valid difference-labelling. We prove this inductively over the number of completed iterations (refer to Figure 3.3 for visual aid). Before any iteration is completed, all assigned labels are valid, since no labels are assigned.

Assume that the labelling assigned after the $i-1$-th iteration is correct. In the i-th iteration, we label ℓ_i, and potentially some vertices above it. If we do not label any vertices other than ℓ_i, then it is clear that our labelling remains valid after the i-th iteration, and we are done. If we do label vertices above ℓ_i in step 3, then we refer to them as $P := \{p_0, p_1, \ldots, p_c\}$, where $p_0 = \ell_i$. For any vertex v with two children a, b, where a is the left child, and b is the right child, we know that the leaves in $t(a)$ must lie to left of the leaves in $t(b)$. The left child, a, must therefore be labelled before any vertex in $t(b)$. It follows, that $2 \max \lambda(t(a)) < \min \lambda(t(b))$. It also follows that P induces a path, in T. Without loss of generality let p_a be p_{a+1}'s child for all a. We therefore know that p_a is p_{a+1}'s right child. Due to step 3, all required edges of labelled vertices are induced by our labelling. We therefore only have to prove, that there are no edges induced by our labelling that are not in T.

Let us assume that the labelling does induce an edge that is not in T, i.e., there exist vertices a, b, c, such that $\lambda(a) - \lambda(b) = \lambda(c)$, even though $ab \notin E(T)$. Step 3 of our algorithm ensures that if this is the case, then $ac \notin E(T)$, since each edge in $E(T)$ is induced by two labels, or induced with another edge in $E(T)$. We can therefore continue the proof without loss of generality.

The vertices a, b, c are not all labelled in previous iterations, since we assume that the labelling provided by the $i-1$-th iteration is correct. Note that labels are assigned in ascending order on P, meaning $\lambda(p_0) < \lambda(p_1) < \ldots < \lambda(p_c)$. It is therefore not possible for exactly two of a, b, c to not be in P, since for all vertices $x, y \notin P, \lambda(x) + \lambda(y) < \lambda(p_0) = \min \lambda(P)$. It follows, that at least one of b, c are in P.

Without loss of generality, let b be in P, and $\lambda(b) \geq \lambda(c)$. It follows, since all assigned labels are strictly positive, that $\lambda(a) > \lambda(b)$, so $a \in P$, a is higher up in P in our drawing of T, and a is not b's parent, since the edge ab is not in E. Let $a = p_q, b = p_{q'}$. For all $v \in \{p_q, p_{q-1}, \ldots, p_{q'+1}\}$, we know that $|N_{\text{out}}(v)| = 2$. This is because, if $|N_{\text{out}}(a)| = 1$, then step 3 assigns $\lambda(a) = 2\lambda(p_{q-1})$. It would follow, since b and c receive labels smaller than that of a, and p_{q-1} has the largest label smaller than that of a, that $\lambda(b) + \lambda(c) < \lambda(p_{q-1}) + \lambda(p_{q-1}) = \lambda(a)$, a contradiction. Similarly, if there were any $q_r \in \{p_{q-1}, \dots, p_{q'+1}\}$ with out-degree one, then since c's label is at most as large as that of b

$$\lambda(a) > \lambda(q_{r-1}) \geq 2\lambda(b) \geq \lambda(b) + \lambda(c),$$

a contradiction.

With that we know that for all $q_r \in \{p_q, \dots, p_{q'+1}\}$ there exists an a_r with $\lambda(q_r) = \lambda(a_r) + \lambda(q_{r-1})$, where in our planar drawing of T, a_r lies left of q_{r-1}. Here we use the fact that a vertex v gets labelled by our algorithm only if all other vertices in $t(v)$ are already labelled. We also know for all c that $\max \lambda(t(a_{c+1})) < 2 \min \lambda(t(a_c))$, since the leaves in $t(a_{c+1})$ all lie to the left of all the leaves in $t(a_c)$. There are two cases.

Case 1. $c \notin t(b) \cup P$

In this case, $\lambda(b) + \lambda(c) \leq \lambda(b) + \lambda(a_{q'+1}) = \lambda(p_{q'+1}) < \lambda(a)$, a contradiction.

Case 2. $c \in t(b) \cup P$

In this case

$$\begin{aligned}
\lambda(b) + \lambda(c) =& \lambda(p_{q'}) + \lambda(c) \\
>& \lambda(p_{q'}) + 2\lambda(a_{q'+1}) \\
>& \lambda(p_{q'}) + \lambda(a_{q'+1}) + 2\lambda(a_{q'+2}) \\
>& \dots \\
>& \lambda(p_{q'}) + \lambda(a_{q'+1}) + \lambda(a_{q'+2}) + \ldots + \lambda(a_{q-1}) + 2\lambda(a_q) \\
>& \lambda(p_{q'}) + \lambda(a_{q'+1}) + \lambda(a_{q'+2}) + \ldots + \lambda(a_{q-1}) + \lambda(a_q) \\
=& \lambda(p_{q'+1}) + \lambda(a_{q'+2}) + \ldots + \lambda(a_{q-1}) + \lambda(a_q) \\
=& \lambda(p_{q'+2}) + \ldots + \lambda(a_{q-1}) + \lambda(a_q) \\
=& \dots \\
=& \lambda(p_q) = \lambda(a),
\end{aligned}$$

a contradiction.

This concludes our proof; we first show that a and b must be in P and are not consecutive, and when that is the case, that there is no choice of c whose label could induce the edge ab. It follows, that the calculated labelling remains valid after the i-th iteration. Inductively, Algorithm 2 provides a valid difference-labelling for any binary out-tree, while assigning a distinct strictly positive label to every vertex. □

The label assigned to each vertex is at most asymptotically twice the size of the largest already assigned label, thus the following theorem is a consequence of the correctness of Algorithm 2.

Theorem 3.11. *For all $n \in \mathbb{N}$, all binary out-trees on n vertices are DGs that can be labelled with distinct strictly positive labels in $O(2^n)$.*

Moreover, this bound on label sizes is asymptotically tight, since paths are binary out-trees that require labels in $\Omega(2^n)$.

A Remark on Other Out-Trees That Are Natural DGs

Here we use the generalised source-join result by M. Sonntag [Son04]. Let $G_1 = (V_1, E_1)$, $\dots, G_n = (V_n, E_n)$ be natural DGs, and $v_1 \in V_1, \dots, v_n \in V_n, s \notin V_1 \cup \dots \cup V_n$. The generalised source-join result says that the graph $G_1 \dot\cup \dots, \dot\cup G_n + \{s\} + \{sv_1, \dots, sv_n\}$ is also a natural DG. We know, due to the fact that out-trees are DAGS with an obvious topological ordering, that in any natural difference-labelling of a tree, the root has the largest label. For any out-tree, we can add a new vertex, and set the root of the tree to be its only child. The new vertex is the new root of the tree, which we can assign a label that is twice the label of the old root of the tree, to create a valid labelling for the new tree. With that, and with the generalised source join result, it is clear to see that any out-tree, where every vertex has out-degree that is either even or one, is a natural DG. This does, however, provide a sub-optimal asymptotic bound on label sizes for binary out-trees, hence the inclusion of our algorithm.

3.5.1 Out-Spiders

An *out-spider*, is an out-tree, where every node except the root has an out-degree of one. An out-spider can be visualised as an out-star where some edges are subdivided. A *leg* is a path starting at the root and ending at a leaf, excluding the root. An out-spider with k leaves is then said to have k legs. We refer to a leg as *trivial* if it consists of one vertex. In this section, we present a complete dichotomy of which out-spiders are DGs, as well as an algorithm to label any spider that is a DG. In Section 3.5.2, we extend this algorithm in order to label any out-tree that is a DG. The following construction is the main building block for both algorithms, since it allows us to repeatedly label induced paths in out-trees.

Path Extension

Given, is a graph G', a valid labelling λ' on G', the paths $P = (p_1, \dots, p_k)$, $P' = (p'_1, \dots, p'_{k'})$ $(0 < k \leq k')$, and a vertex v in G'. We need to label the graph $G = G' \dot\cup P \dot\cup P' + \{vp_1, vp'_1\}$ obtained by extending the paths P and P' from v. We provide a construction for a labelling λ on G. We refer to this construction as *path extension.*

Provided that the labelling λ' does not use the label 0, and that v is the only vertex with a label in

$$\{\lambda'(v)/2^i \mid i \in \{0, \dots, k-1\}\},$$

then we construct a labelling λ on G in the following manner. Let

$$\lambda(u) = 2^{k'}\lambda'(u) \quad (u \in V(G'))$$

and ℓ be an arbitrary odd number such that $\ell > 4\max\{|\lambda(u)| \mid u \in V(G')\}$,

$$\lambda(p'_i) = 2^{k'-i}\ell \qquad (i \in \{1, \dots, k'\}) \text{ and}$$

$$\lambda(p_i) = \lambda(v)/2^{i-1} - \lambda(p'_i) \quad (i \in \{1, \dots, k\}).$$

Lemma 3.12. *Given a DG $G = (V, E)$, with*

$$G = (G' \dot\cup P \dot\cup P') + \{vp_1, vp'_1\},$$

for a graph G', a vertex $v \in V(G')$, and paths $P = (p_1, \dots, p_k)$ $P' = (p'_1, \dots, p'_{k'})$ with $0 < k \leq k'$. If there exists a valid difference-labelling λ' for G' fulfilling the requirements for using path extension, then for any integer $\gamma \geq 4$ path extension can be used to construct a valid difference-labelling λ on G with

$$-\lambda(p_k), \lambda(p'_{k'}) \geq \gamma \max\{|\lambda(u)| \mid u \in V(G')\}.$$

Proof. Let λ' be a difference-labelling on G' fulfilling the requirements to extend P and P' from v. Let us construct the difference-labelling λ on G using path extension. By stating that the chosen ℓ in our path extension is arbitrary, this allows us allows us to vary our choice of ℓ in order to adjust the size of the assigned labels in accordance with any given γ.

We need to prove that λ is a valid difference-labelling for G. Notice how the labels assigned on P' are strictly positive, while the labels assigned on P are strictly negative. First we prove that G is a sub-graph of the graph induced by this labelling. For all the vertices in G', we just multiply their labels by a factor, so all edges from G' are induced by λ. For any edge $p'_ip'_{i+1}$ we know that $\lambda(p'_i) - \lambda(p'_{i+1}) = 2^{k'-i}\ell - 2^{k'-i-1}\ell = 2^{k'-i-1}\ell = \lambda(p'_{i+1})$, so it is induced by λ. For any edge p_ip_{i+1} we know that

$$\lambda(p_i) - \lambda(p_{i+1}) = \frac{\lambda(v)}{2^{i-1}} - \lambda(p'_i) - \frac{\lambda(v)}{2^i} + \lambda(p'_{i+1}) = \frac{\lambda(v)}{2^i} - \lambda(p'_{i+1}) = \lambda(p_{i+1}).$$

The edges vp_1, vp'_1 are induced by $\lambda(v) - \lambda(p_1) = \lambda(v) - \lambda(v) + \lambda(p'_1) = \lambda(p'_1)$. It follows that G is a sub-graph of the graph induced by λ. What we now need to prove is that the graph induced by λ is a sub-graph of G. Our argumentation goes as follows. The labels assigned on the paths are so much larger than previously assigned labels that they do not interfere with G', and they are spaced out in a way, such that the sum/difference of labels assigned to non-adjacent vertices on the paths results in a label that we know is not assigned. This is mostly because p_i is roughly equal to $-p'_i$ and the difference between p'_i and p'_{i+1} decreases exponentially.

Let us assume for the sake of contraposition, that the graph induced by λ is not a sub-graph of G. There must therefore exist vertices a, b, c with $\lambda(a) - \lambda(b) = \lambda(c)$, whose labels induce an edge not in G. It is not possible for a, b, c to be in $V(G')$, since λ induces the same graph on $V(G')$ as λ'. Furthermore, it is not possible for exactly two of a, b, c to be in $V(G')$ since the sum of any two labels assigned by λ in $V(G')$ has an absolute value of at most $\ell/2$, which is smaller than the absolute value of any label assigned on the two paths. We can therefore focus on the case that at least two of a, b, c lie on the paths.

We know with Theorem 2.4, that if there is an induced edge going between two vertices of the same path that should not be there, then there must be another induced edge leaving that path, since the labels assigned to the path cannot induce such an edge on their own. All that is then left to prove, is that there are no edges entering or leaving either path, except of those from v, which only has edges going to the first node of each path. In Case 1 we show that there cannot be any edges entering the paths, except for those coming from v and going to p_1 and p'_1. In Case 2 we show that there cannot be any edges leaving either path.

Case 1. $a \in V(G')$

In this case, b and c must lie on the paths. We know that b and c must lie on different paths. Otherwise, $\lambda(b)$ and $\lambda(c)$ would have the same sign, and the absolute value of $\lambda(b) + \lambda(c)$ would be much larger than any label assigned to nodes in $V(G')$. This means that $\lambda(a) = \lambda(p_x) + \lambda(p'_y)$ for some $x, y \in \mathbb{N}_0$. If $x = y$, then $\lambda(a) = \lambda(p_x) + \lambda(p'_x) = \lambda(v)/2^{x-1} - 2^{k'-x}\ell + 2^{k'-x}\ell = \lambda(v)/2^{x-1}$, which means $a = v$, and $x = y = 1$ since as per our assumption, v is the only vertex with a label of this form.

For all other possible combinations of x and y, the absolute value of the sum of $2^{k'-y}\ell$ and $\lambda(v)/2^{x-1} - 2^{k'-x}\ell$ is simply too large for it to be equal to $\lambda(a)$. If $x > y$, then

$$\begin{aligned}\lambda(a) &= \lambda(v)/2^{x-1} - 2^{k'-x}\ell + 2^{k'-y}\ell \\ &\geq \lambda(v)/2^{x-1} - 2^{k'-y-1}\ell + 2^{k'-y}\ell \\ &= \lambda(v)/2^{x-1} + 2^{k'-y-1}\ell \\ &\geq -|\lambda(v)| + \ell \\ &> -\ell/4 + \ell \\ &> \max\{|\lambda(u)| \mid u \in V(G')\}.\end{aligned}$$

The case that $x < y$ results analogously (since $\lambda(p'_i) = \lambda(v)/2^{i-1} - \lambda(p_i)$) in $\lambda(a) < -\max\{|\lambda(u)| \mid u \in V(G')\}$. The vertex a can therefore not lie outside the paths, unless $a = v$, in which case it only has edges going to the first vertex of each path.

Case 2. $a \notin V(G')$

If a lies on one of the paths then we know that either b or c (or both) also lie on the paths. If b lies on the other path as a, then $\lambda(a) - \lambda(b) = 2^{\xi}\lambda(a)$ for some $\xi \in \mathbb{N}$. This is because the nodes that are above a on its leg are the only nodes with labels that have such a large absolute value, and the same sign as $\lambda(a) - \lambda(b)$. Note that we know due to our choice of $\lambda(v)$ and ℓ, that $\lambda(p'_{k'})$ and $\lambda(v)/2^{k'-1} - \lambda(p'_{k'})$ are odd and both much larger than $\lambda(v)$. Hence, $\lambda(a) - \lambda(b) = 2^{\xi}\lambda(a)$ would mean that for some odd $\psi \in \{\lambda(p'_{k'}), \lambda(v)/2^{k'-1} - \lambda(p'_{k'})\}$ and $x, y, z \in \mathbb{N}, 2^x\psi = 2^y\psi + 2^z(\lambda(v)/2^{k'-1} - \psi)$. Since $\lambda(v)$ is smaller than ψ, there are some prime factors that make up ψ that are either not present in $\lambda(v)$ or do not divide $\lambda(v)$ enough times. By multiplying or dividing $\lambda(v)$ with any power of two, we do not add those missing prime powers back. It follows that $(2^z\lambda(v))/2^{k'-1} \not\equiv_{\psi} 0$. This leads us to the following congruence obstruction

$$0 \equiv_{\psi} 2^x\psi = 2^y\psi + 2^z\left(\frac{\lambda(v)}{2^{k'-1}} - \psi\right) \equiv_{\psi} \frac{2^z\lambda(v)}{2^{k'-1}} \not\equiv_{\psi} 0.$$

With the same argument, we can show that c can also not lie on the other path as a. We can therefore assume that b lies on the same path as a, which means c cannot lie on either path. If b lies beneath a on its path, then $\lambda(a) - \lambda(b)$ has an absolute value at least as large as the label on a's leg with the smallest absolute value, which still has an absolute value much larger than that of c. If b lies above a on its leg, then $|\lambda(a) - \lambda(b)| \geq |\lambda(a) - 2\lambda(a)| \geq |\lambda(a)|$, which is again much larger than $|\lambda(c)|$.

We already know that there cannot be any edges induced by three labels in $V(G')$. Case 1 excludes the possibility of any incoming edges to the newly labelled paths, unless they are coming from v, and going to the first node of each path. Case 2 excludes the possibility of any edges leaving the newly labelled paths. It follows that λ is a valid labelling of G. Since ℓ is selected arbitrarily, and the lower bound for the minimum label size is below that stated by the lemma, the lemma holds true. □

Labelling Out-Spiders

Notice how if we can use path extension to extend paths from vertices v, u, then one could extend paths from v, and in the resulting graph, one can still extend paths from both vertices. This leads us to the following result.

Corollary 3.13. *All out-spiders with an even number of legs are DGs.*

Proof. Given an out-spider with an even number of legs we can assign an arbitrary label to the root and use path extension repeatedly to label two legs at a time until all legs are labelled. □

Corollary 3.14. *Out-spiders with an odd number of legs, with at most one trivial leg and at least five legs, are not DGs.*

Proof. Let us consider an out-spider with at least five legs, an odd number of legs, and at most one trivial leg. All relevant neighbourhoods in such a spider are distinct. The root r of the spider has odd out-degree, and in-degree zero. All but at most one of the nodes in the neighbourhood of r have odd out-degree (because they must have out-degree one). Furthermore, $N_{\text{in}}(N_{\text{out}}(r)) = \{r\}$, and all nodes in $N_{\text{out}}(N_{\text{out}}(r))$ have in-degree one. The out-spider is therefore an OOPS, meaning it cannot be a DG. □

Corollary 3.15. *Out-spiders with more than one trivial leg are DGs.*

Proof. Let us assume we have an out-spider with more that one trivial leg. If the spider has an even number of legs, then it is a DG according to Corollary 3.13. If the spider has an odd number of legs, then we can remove one trivial leg. The new spider has an even number of legs, and we can therefore find a valid difference-labelling for it. We can then put that leg back in, and assign its vertex the same label as one of the labelled vertices from one of the other trivial legs. Since both vertices have the same neighbourhoods, this does not cause any problems.

In both cases, the out-spider with more than one trivial legs has a valid difference-labelling. □

One can verify, that out-spiders with three legs, and one trivial leg have a valid difference-labelling. This can be obtained by labelling the root with a sufficiently large power of two, and setting the label of the first node one of the non-trivial legs to be half of that. That implicitly labels the rest of that leg. The other two legs are labelled using path extension. Such spiders therefore have a valid difference-labelling.

Out-spiders with three legs, and no trivial legs are OOPSs, meaning they cannot be DGs. It is interesting to note at this point that the only spiders that are not DGs are OOPSs. This leads us nicely into the next section.

3.5.2 Out-Trees

In this section we prove that an out-tree is a DG if and only if it is not an OOPS (refer to Section 3.4). We do this by providing an algorithm that labels such trees. Our algorithm assumes that the given graph is not an OOPS, and finds very specific induced out-spiders in it. By using an altered version of our labelling algorithm from Section 3.5.1 we are able to label these induced spiders without inducing unwanted edges with preexisting labels.

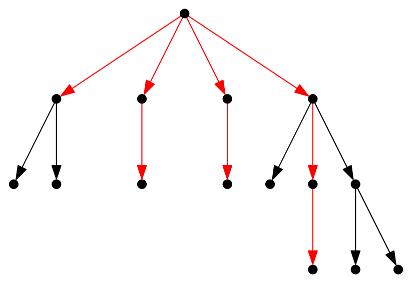

Figure 3.4: A maximal OIS belonging to the root of this tree (red).

We work our way from the root to the leaves, by labelling induced out-spiders until all nodes are labelled.

In Section 3.4, we introduce the term 'problematic with respect to a vertex'. Since for u to be problematic with respect to v, u must be in $N_{\text{out}}(v)$, we simply refer to a vertex of an out-tree as *problematic* if it is problematic with respect to its parent. By closely inspecting the conditions for a vertex to be problematic one can determine, that a vertex w in an out-tree is non-problematic, if and only if it has even out-degree, or if the relevant neighbourhoods of $N_{\text{out}}(w)$ are not pairwise distinct. From a given out-tree T we can therefore construct an out-tree T', where a vertex is non-problematic if and only if it has even out-degree, such that T can be obtained from T' by vertex duplication. Every vertex in T that has even out-degree can be kept. If a vertex w has odd out-degree in T but is not problematic, then it must have two children with non-distinct relevant neighbourhoods. The only way for the relevant-neighbourhoods of two vertices in a tree to be the same, is if both vertices are leaves that share a parent. We can therefore delete one of the leaves of w, and its degree in T' becomes even. Due to our construction of T', we know that T can be obtained from T using node duplication. All we must therefore prove, is that T' is a DG, and it follows by Lemma 2.2 that T is a DG. We therefore assume for the rest of this section that a vertex in T is problematic if and only if it has odd out-degree.

We must now discuss how the induced spiders are found.

Odd-Induced Out-Spiders

We define an *odd-induced-out-spider* (OIS) of an out-tree T as a sub-out-tree S, such that S is an induced out-spider in T, the leaves of S have even out-degree in T, and all other vertices except the root of S have odd out-degree in T. For a vertex v in a tree T, let

$$S(v) = \{S \mid S \text{ is an OIS with the maximal number of vertices and } v \text{ as the root.}\}$$

Figure 3.4 shows a possible maximal OIS for the root of an out-tree. We denote the length of the longest leg (in number of vertices) of any element in $S(v)$ with $l(v)$, note that this is a well defined function, meaning any element in $S(v)$ has at least one leg of length $l(v)$.

Lemma 3.16. *If T is not an OOPS, then for every odd number k, every maximal OIS in T, with k legs, has at least $\left\lfloor \frac{k}{2} \right\rfloor$ trivial legs.*

Proof. If T is not an OOPS, then every vertex in T with odd out-degree k has at least $\left\lfloor \frac{k}{2} \right\rfloor$ non-problematic children. We assumed that in T, all non-problematic children have even out-degree. A maximal OIS includes at least all the children of its root. If a non-problematic child is in an OIS, then it must be a leaf; hence, any maximal OIS with an odd number, k, of legs must therefore have at least $\left\lfloor \frac{k}{2} \right\rfloor$ trivial legs. $\square$

Altered Labelling Algorithm for OISs

In the previous section, we provided an algorithm for labelling out-spiders. However, in that section, we did not need to work around labels that exist outside of the spider. With this algorithm, the root of the spider is already labelled, and we have to label the rest of the spider, while ensuring that there are no collisions between new labels, and labels that already exist in the graph. For this reason, our algorithm depends on some very specific invariants in order to label an OIS in a given out-tree T.

Invariants

1. For every labelled vertex v with odd out-degree and unlabelled children, exactly one of its children is labelled, and that child is the first vertex of a longest leg of every spider in $S(v)$.
2. If a vertex has even out-degree, then either none or all of its children are labelled.
3. For all vertices v with even out-degree, v is the only vertex that is assigned a label in
$$\{\lambda(v)/2^i \mid i \in \{0, \dots, l(v)-1\}\}.$$
4. No two vertices have the same label.
5. The sub-graph of labelled vertices is a sub-tree of T.

When our algorithm is called, we can assume that all invariants hold true, and we are given an out-tree T on n vertices and a vertex v, as well as a maximal OIS $S \in S(v)$, and the label of v, $\lambda(v)$.

Algorithm 3: Labelling an OIS

For a vertex v and a maximal OIS $S \in S(v)$

- If S has an even number of legs, then with Lemma 3.12, the legs of S can be labelled two at a time, while ensuring in each step, that the leaves of the spider receive labels with an absolute value at least 2^{n+1} times larger than the labels in the rest of the graph.
- If S has an odd number of legs, a longest leg has at least one labelled vertex. The remaining legs can be paired up so that in each path-extension a trivial leg is labelled with a non-trivial leg. Label all unlabelled legs in accordance to Lemma 3.12, while ensuring in each step, that the leaves of the spider receive labels with an absolute value at least 2^{n+1} times larger than the labels in the rest of the graph.

Claim. Algorithm 3 provides a valid difference-labelling for the sub-tree of labelled vertices, and does not break the invariants.

Proof. In order to show that our labelling is correct, all we need to show is that our use of path extension is correct. The correctness of the labelling would then follow directly from the correctness of path extension.

If S has an even number of legs, then we know with our third invariant, that the requirements to use path extension are held. If S has an odd number of legs, then due to our first invariant, the first vertex of a longest leg is already labelled. We therefore know with Lemma 3.16 that at least half of the remaining legs are trivial. Due to $\{\lambda(v)/2^i \mid i \in \{0, \dots, 0\}\} = \{\lambda(v)\}$, and our fourth invariant, we know that as long as one of the two paths we would like to extend from v is trivial, we can always use path extension to extend the two legs. In either case, our algorithm provides a correct labelling. We now need to prove that our invariants remain true after our algorithm is called.

If a vertex w with odd out-degree is labelled by Algorithm 3, then due to the fact that in every maximal OIS in which w is a part it is not a leaf, exactly one of w's children must also be labelled. We specifically picked all our OIS to be maximal. It is clear that a maximal OIS that includes the vertex w must also include a longest leg from an $S \in S(w)$. If a child of w is the first vertex on a longest leg of S, then it is the first vertex of a longest leg in every element of $S(w)$ (due to their maximality). It follows, that exactly one child of w is labelled, and that this child is the first vertex on a longest leg of every element of $S(w)$. Thus our first variant remains true.

If a vertex w has even out-degree and unlabelled children, then in every OIS of which w is a part up to this point, it is a leaf, meaning that none of the vertices below it in the tree could be labelled. Thus our second variant remains true.

By Lemma 3.12, we can ensure whenever we label two paths, that we find a valid labelling such that the leaves receive labels that are 2^{n+1} times larger than any label. It also ensures, that if the invariants held for any already labelled vertex, then they still hold. Furthermore, the only vertices that our algorithm labels, that have even degree, are the leaves. Due to the relative size and the sign of the labels assigned to the leaves, the third invariant also holds true.

Path extension ensures that labels remain distinct after our algorithm is called, since labels on the legs must be distinct, and already assigned labels are all multiplied by a factor. Thus the fourth invariant remains true.

Due to the fact that we label all the vertices of paths at the same time, and that we know that the root of the OIS is already labelled, it is not possible for us to label a vertex, without labelling its parent. We therefore know that the sub-graph of labelled vertices is in fact a sub-tree of T. Thus the fifth invariant remains true.

All invariants then hold for the newly labelled vertices, and all previously labelled vertices of the tree. □

Labelling Algorithm

By using Algorithm 3 multiple times, we can create a valid difference-labelling for any out-tree that is not an OOPS.

Algorithm 4: Labelling Out-Trees

1. Start by labelling our root r with the label 2^n.
2. If the root has odd out-degree, then pick an arbitrary $S \in S(r)$. Label a longest leg in S as in Theorem 2.4.
3. Pick an arbitrary labelled vertex v with unlabelled children.
4. Pick an arbitrary $S \in S(v)$ and label it with Algorithm 3.
5. Return to Step 3.

Claim. Algorithm 4 provides a valid difference-labelling.

Proof. The correctness of Algorithm 4 follows from the correctness of Algorithm 3. All we need to show is that the invariants and requirements to call Algorithm 3 always hold, and that all vertices in our tree are assigned a label. After the first 2 steps it is clear that all invariants hold true. Since the invariants still hold true after each call to Algorithm 3, we know that the invariants always hold true.

Since there is a directed path from the root of T to every vertex in T, our algorithm assigns a label to every vertex in T. It follows then from the correctness of Algorithm 3, that Algorithm 4 is also correct. □

In conclusion, the correctness of Algorithm 4 implies the following theorem.

Theorem 3.17. *An out-tree T is a DG if and only if, it is not an OOPS.*

Proof. If an out-tree is not an OOPS, then Algorithm 4 and Lemma 2.2 provide a valid difference-labelling for it. If an out-tree is an OOPS, then by Theorem 3.10 it cannot be a DG. It follows that an out-tree is a DG if and only if it is not an OOPS. □

3.5.3 A Complete Dichotomy

With the work of the previous section, we already almost have a complete dichotomy of rooted trees.

Out-trees

By Theorem 3.17, we know that an out-tree is a DG if and only if it is not an OOPS.

In-Trees

Let us consider a path $P := (v_1, \ldots, v_n)$. By Theorem 2.4 we know that P is a DG, that can be labelled with distinct non-zero labels. By Lemma 2.2, we know that if we duplicate any vertex of P, the new graph is also a DG. Furthermore, we know that P is an in-tree, and no matter how many times we duplicate v_1, it remains a DG. We define an *in-broom* as an in-tree that can be obtained from a directed path on at least two vertices, by duplicating the first vertex an arbitrary number of times[2].

Corollary 3.18. *An in-tree $T = (V, E)$ is a DG if and only if T is an in-broom.*

Proof. If T is an in-broom, then we already know that it is a DG.

In-brooms are exactly the in-trees in which a vertex has more than one child if an only if all of its children are leaves. Hence, if T is not an in-broom, then there are three vertices u, v, w with $uw, vw \in E$, and at least one of u, v are not leaves. It also means, that there is no total sink in T, since the only in-trees with total sinks are in-stars, and those are also in-brooms. The vertices u, v must then also have different neighbourhoods, and $uv, vu \notin E$, since that would make T not a tree. Since T is an in-tree, we know that $N_{\text{out}}(\{u, v\}) = \{w\}$. We know that the bipartite graph $B = (\{u, v, w\}, \{\{u, w\}, \{v, w\}\})$ does not have a matching that matches $\{u, w\}$, so we know that T cannot be a DG since it is an OMG. □

A great thing to note, is that this allows us to classify rooted trees in linear time. This is because it is very easy to check if an in-tree is an in-broom, and for any out-tree it is possible to check in linear time, if it is an OOPS.

[2]This definition of in-brooms therefore also includes all paths

A Short Dichotomy of Rooted Forests

A rooted forest, is the disjoint union of rooted trees. By Corollary 3.7, we know that a disjoint union of graphs is a DG if and only if all connected components are DGs that can be labelled without using the label 0. We know that all rooted trees that are DGs can be labelled without using the label 0. We can then conclude this section with the following corollary.

Corollary 3.19. *A rooted forest, is a DG if and only if all rooted trees in the forest are DGs.*

4. Difference-Number

As previously discussed, you cannot turn any non-DG into a DG by adding isolated nodes, like you can with sum-graphs. We define the *difference-number* of a graph $G = (V, E)$ (denoted $m(G)$), as the minimum number of extra nodes one needs to add, in order to create a DG $G' = (V', E')$ with $G = G'[V]$.

Theorem 4.1 ([EG84])**.** *For any graph $G = (V, E)$, $m(G) \leq |E|$.*

Proof. Given a directed graph $G := (V, E)$, with $V := \{v_1, \ldots, v_n\}$ and $E = \{e_1, ..., e_m\}$, our goal is to add m new nodes, and label each node in such a way as to induce a DG with G as an induced sub-graph. Since all directed graphs with one or two nodes are DGs, we can assume that $n > 2$.

Let us set $G' := (V', E')$ with $V = \{v_1, \ldots, v_n, \ell_1, \ldots \ell_m\}$. We provide an implicit definition of E' by providing a labelling λ' on G'. This ensures that G' is a DG. Let $\lambda'(v_i) = 3^i$ for $i \in \{1, \ldots, n\}$. Currently, no two nodes are connected by an edge. Then let, for all $e_i = v_j v_k, \lambda'(\ell_i) = \lambda'(v_j) - \lambda'(v_k)$. Clearly then, v_j is connected to v_k through ℓ_i, so $E \subseteq E'$. We must only prove that ℓ_i does not connect any other two nodes from V.

Let $\ell \in V' \backslash V$, with $\lambda'(\ell) = 3^k - 3^t$ and $\lambda'(\ell) = 3^i - 3^j$. That means that $3^k - 3^t = 3^i - 3^j$ meaning $3^k + 3^j = 3^i + 3^t$. Without loss of generality let $k > j, i > t$. If without loss of generality $k > i$ then $3^k + 3^j > 3^k > 2 \cdot 3^i > 3^i + 3^t$, a contradiction. It follows, that $k = i$, thus $j = t$.

So, all added ℓ_i's connect exactly two nodes from V to each other, meaning that our construction is correct. The graph G' is a DG and G is an induced sub-graph of G'. The theorem holds true, since G' has exactly $n + m$ vertices. $\square$

This also shows, that the difference-number of a graph is always defined. Throughout this thesis, we provide multiple constructions, of varying complexity, that prove Theorem 4.1. Another very similar construction is presented by Eggleton and Gervacio ([EG84]). For now, this simple construction is sufficient.

4.1 Basic Difference-Numbers

In this section, we present bounds on the difference-numbers of some graph classes.

Theorem 4.2. *The difference-number of any directed cycle on n nodes is one for $n > 2$ and zero otherwise.*

Proof. By Theorem 2.7 we know that C_1 and C_2 are DGs; their difference-numbers are therefore zero. All other directed cycles are not DGs, their difference-number must be strictly larger than zero. By adding one extra node however, one can create a DG with the cycle as an induced sub-graph. Given a cycle $G = (v_n, \ldots, v_1, v_n)$ we label it with $\lambda(v_i) = 2^i$. We add a node a to G, and give it the label $2 - 2^n$. This is a negative number and is therefore not an already existing label. This also connects v_1 to v_n. It also does not connect any other two nodes since for any $i \leq n, 2 - 2^n - 2^i < 0$ and for any $1 < i \leq n, 2^i + 2^n - 2 > 2^n$. □

We know that directed complete graphs are DGs, since we can assign the label zero to each node. However, if we add one isolated vertex to K_n, then no vertex can receive the label zero, since the new node has no incoming edges. Moreover, it forces all labels in the graph to be distinct by Lemma 2.1, since the graph no longer contains a total sink. Let $K_n^* := K_n \cup \{v\}$ where v is an isolated vertex.

Theorem 4.3. *For all $n \in \mathbb{N}$ $m(K_n^*) \leq 1$.*

Proof. Given a K_n^* with isolated vertex v, we assign the label one to every vertex, and the label -2, to the isolated vertex. We add one extra vertex to the graph, make it a universal sink, and assign it the label zero. Since at least one vertex receives the label zero, our labelling induces edges between all nodes with the same label. No edges are induced between nodes with the label 1, and the v. This new graph is therefore a DG that has K_n^* as an induced sub-graph. The difference-number of K_n^* is therefore at most one. □

Furthermore, for any valid difference-labelling on K_n^*, every vertex must receive a different label by Lemma 2.1. For sufficiently large n, K_n^* cannot be a DG, since it would violate the bound from Corollary 3.2. It follows that our bound is also tight.

4.2 The Difference-Number of Rooted Trees

After showing which rooted trees are DGs in Section 3.5, we now present bounds on their difference-number.

4.2.1 Out-Trees

We know, that the only out-trees that are not DGs are OOPSs. Here we bound the difference-number of out-trees from above by two. We do this by presenting an algorithm that labels any out-tree, while using exactly two extra vertices. For the most part, this algorithm works similarly to Algorithm 4 from Section 3.5.2. We therefore assume that an intuitive understanding of that algorithm is already present. Just like in that algorithm, we label maximal OISs in the tree, until all vertices are labelled. In Theorem 3.10, we exploit the fact that in some trees, one must assign labels a, b with $a = 2b$, to find a forbidden substructure. Our goal is therefore to mostly avoid assigning such labels, while avoiding other problems that arise along the way.

Weak Path Extension

For a directed graph G and a set of vertices Z with $Z \cap V(G) = \emptyset$, we say that a labelling λ is a difference-labelling *on G with additional vertices Z* if for the DG G' induced by λ on $V(G) \cup Z$, $G'[V(G)] = G$. Given is a graph G', a valid labelling λ' on G' with additional vertices Z, the paths $P = (p_1, \dots, p_k), P' = (p'_1, \dots, p'_{k'})$ $(0 < k \leq k')$, and a vertex v in G'. We need to label the graph $G = G' \dot\cup P \dot\cup P' + \{vp_1, vp'_1\}$ obtained by extending the paths P and P' from v. We provide a construction for a labelling λ on G with additional vertices Z. We refer to this construction as *weak path extension*.

Provided that the labelling λ' does not use the label zero, and we can find a $z \in Z$ such that z is the only vertex with a label in

$$\{\pm i\lambda'(z) \mid i \in \{1, \dots, k'-1\}\},$$

and that v is the only vertex with a label in

$$\{\lambda'(v) - i\lambda'(z) \mid i \in \{0, \dots, k+k'-2\}\},$$

then we construct a labelling λ for G with additional vertices Z in the following manner. Let ℓ denote an arbitrary number larger than $\max\{|\lambda'(u)| \mid u \in V(G') \cup Z\}$, then set

$$\lambda(p'_1) = 4k'\ell + k'|\lambda'(z)|,$$

$$\lambda(p_1) = \lambda'(v) - \lambda(p'_1),$$

$$\lambda(u) = \lambda'(u) \quad (u \in V(G') \cup Z) \text{ and}$$

$$\lambda(p'_i) - \lambda(z) = \lambda(p'_{i+1}), \lambda(p_i) - \lambda(z) = \lambda(p_{i+1}) \text{ (for all } i).$$

Lemma 4.4. *Given a graph $G = (V, E)$, with*

$$G = (G' \dot\cup P \dot\cup P') + \{vp_1, vp'_1\},$$

for a vertex $v \in V(G')$ and the paths $P = (p_1, \dots, p_k)$, $P' = (p'_1, \dots, p'_{k'})$ with $0 < k \leq k'$. If there exists a difference-labelling λ' on G' with additional vertices Z fulfilling the requirements for using weak path extension, then for any $\gamma \geq 4$, weak path extension can be used to provide a valid difference-labelling λ on G with additional vertices Z, such that

$$-\lambda(p_k), \lambda(p'_{k'}) > \gamma \max\{|\lambda(u)| \mid u \in V(G') \cup Z\}.$$

Proof. Let λ' be a valid labelling on G' with additional vertices Z that fulfills the requirements for using weak path extension. Let λ be a difference-labelling on G with additional vertices Z, obtained using weak path extension. The labelling λ induces an edge from every p_i and p'_i to p_{i+1} and p'_{i+1} respectively, and from v to p_1 and p'_1. Since ℓ was selected to be arbitrary, we can vary our choice of ℓ to obtain labels with larger absolute values, in accordance to any selected γ. We therefore know, that G is a sub-graph of the graph induced by λ. All we now need to prove, is that G is in fact an induced sub-graph of the graph induced by λ. Our argumentation goes as follows. The labels assigned on the paths are so much larger than previously assigned labels that they do not interfere with G', and they are spaced out in a way, such that the sum/difference of labels assigned to non-adjacent vertices on the paths results in a label that we know is not assigned.

Let us assume for the sake of contradiction, that λ induces an edge between two vertices in V that is not present in G. There must therefore exist three vertices $a, b, c \in V \cup Z$ with $\lambda(a) - \lambda(b) = \lambda(c)$ with $a, b \in V$ and $ab \notin E$. We know that not all of a, b, c can be in $V(G') \cup Z$ since the labels in G' and Z remain unchanged. The label with the smallest absolute value that is assigned on either path has an absolute value of at least $3k'\ell$. It is therefore not possible for exactly two of a, b, c to be in $V(G') \cup Z$, since the absolute value of the sum of any two labels assigned in $V(G') \cup Z$ is at most 2ℓ. At least one of b, c must therefore lie on one of the extended paths. We know that

$$\begin{aligned}(\lambda(P) \cup \lambda(P')) + (\lambda(P) \cup \lambda(P')) = &\{\lambda(v) - i\lambda(z) \mid i \in \{0, \dots, k' + k - 2\}\}\\ &\cup \{2\lambda(p_1) - i\lambda(z) \mid i \in \{0, \dots, 2k - 2\}\}\\ &\cup \{2\lambda(p'_1) - i\lambda(z) \mid i \in \{0, \dots, 2k' - 2\}\}.\end{aligned}$$

Elements of the first set are obtained by adding one label assigned on P with one label assigned on P', while elements of the second and third sets are obtained by adding up two labels assigned on the same path. Elements of the first set have absolute values much too small to have been assigned on either path. On the other hand, due to the large size of $\lambda(p_1)$ and $\lambda(p'_1)$, the elements of the second and third sets have absolute values much too large to have been assigned on either path. It is therefore easy to verify that $((\lambda(P) \cup \lambda(P')) + (\lambda(P) \cup \lambda(P'))) \cap (\lambda(P) \cup \lambda(P')) = \emptyset$, meaning it is not possible for all three of a, b, c to lie on the paths. We can therefore focus on the case that exactly two of a, b, c lie on the paths.

If exactly b and c lie on the paths, then either they lie on the same path, or they lie on different paths. We now show that in either case, $\lambda(b) + \lambda(c)$ is a label that we assume is not assigned by λ'. By looking closely at the labels assigned on the paths, one can see that $\lambda(P) + \lambda(P') = \{\lambda(v) - i\lambda(z) \mid i \in \{0, \dots, k' + k - 2\}\}$. It follows, due to our assumption on λ', that if b and c lie on different paths, then (w.l.o.g) $b = p_1, c = p'_1, a = v$, a contradiction, since the edge ab is then in G. The vertices b and c must therefore lie on the same path. However, since that implies that the labels of b and c have the same sign, we know that their sum has an absolute value much larger than any label assigned in V, a contradiction.

If exactly one of b, c lie on one of the paths, then we know that a must also lie on one of the paths. Now we show, that for whichever $x \in \{b, c\}$ lies on the paths, $\lambda(a) - \lambda(x)$ is a label that we assume is not assigned by λ'. If b lies on one of the paths, and a lies on the same path as b then $\lambda(a) - \lambda(b) = i\lambda(z)$ ($i \in \{-k' + 1, \dots, k' - 1\}$). Due to our assumption we know that there exists only one vertex in $V(G') \cup Z$ with a label of this form, namely z, meaning that a, b are two consecutive vertices of the same path, so $ab \in E$, a contradiction. If a lies on the other path, then $\lambda(a) - \lambda(b)$ has an absolute value much larger than any assigned label, meaning it cannot equal $\lambda(c)$, a contradiction. Similarly, if c lies on one of the paths, we can show that $\lambda(a) - \lambda(c) \neq \lambda(b)$.

It follows, that λ is a difference-labelling on G with additional vertices Z. Since we choose ℓ arbitrarily, we can vary our choice of ℓ according to our choice of γ. The lemma therefore holds true. □

Labelling Out-Trees

By varying our choice of γ we can ensure that weak path extension can be used repeatedly to extend multiple paths from a given vertex. We use this to add two vertices to a given tree, assign them some very specific labels, and then use them repeatedly to extend paths from vertices to label OISs in the tree, until all vertices are labelled. For a vertex z, if an out-spider S, with an even number of legs is extended from a vertex v by using weak path extension repeatedly to extend pairs of legs, using z's label to label the vertices on the legs, then we say that S is extended (from v) *using* z. Since all graphs on two vertices are DGs, we assume for the rest of this section, that the input graph always has at least three vertices.

Algorithm 5: Labelling Out-Trees Using Two Extra Vertices

Given an out-tree T on n vertices, add two new vertices to the tree (u and w)

1. Set $\lambda(u) = 1$.
2. Set $\lambda(w) = 2n$.
3. Assign an arbitrary label larger than $2^{n+1}n^2$ that is divisible by 2^n to the root of the tree.
4. If the root has even out-degree then skip this step. If the root has odd out-degree, then find a maximal OIS starting at the root of the tree, and label its maximal leg using the trivial path labelling.
5. Find a labelled vertex v with unlabelled children, and find a maximal OIS S with root v.
6. Find a vertex $z \in \{u, w\}$ such that v is the only vertex with a label in $\{\lambda(v) - i\lambda(z) \mid i \in \{0, \dots, 2(n-1)\}\}$.
7. If v has odd out-degree, then a maximal leg of S has at least one labelled vertex; extend the rest of the legs two at a time using z with $\gamma = 4$ (refer to Lemma 4.4)[a].
8. If v has even out-degree, then none of the vertices in S are labelled; extend the legs of the spider two at a time using z, with $\gamma = 4$.
9. Return to step 5

[a]To build an intuitive understanding of the proof, it could help to take γ to be a very large number or variable.

Claim. Algorithm 5 gives a valid labelling for any out-tree, using two extra vertices.

Proof. All we need to prove, is that our use of weak path extension is a correct one, and the correctness of Algorithm 5 follows directly. Since the label of the root is strictly larger that $2^{n+1}n^2$, we know that the smallest label assigned to a vertex in the tree after the first four steps is larger than $4n^2$. Furthermore, after the first four steps, the assigned labels induce a path, and u, w are both isolated. It follows, that after the first four steps, we can use weak path extension to extend paths from any labelled vertex in the tree using u or w.

We know that after the first four steps, the smallest label is larger than $4n^2$. When using weak path extension, we only ever assign labels with absolute values that are larger than previously assigned labels, meaning that for $z \in \{u, w\}$ the condition that z is the only vertex with a label in $\{\pm i\lambda(z) \mid i \in \{1, \dots, n-1\}\}$ always holds. Since the longest leg has length at most $n - 1$, all that is then left to prove is that the following invariant holds. For any labelled vertex a, their exists a $z \in \{u, w\}$ such that a is the only vertex with a label in $\{\lambda(a) - i\lambda(z) \mid i \in \{0, \dots, 2(n-1)\}\}$. We prove this invariant inductively. Due to the size of the labels assigned in the first four steps, the invariant holds after the first four steps for all labelled vertices and all $z \in \{u, w\}$. Now let us assume that the invariant held for the previous iteration. When we select a vertex v in step five, we know that there exists a $z \in \{u, w\}$ such that v is the only vertex with a label in $\{\lambda(v) - i\lambda(z) \mid i \in \{0, \dots, 2(n-1)\}\}$. We then extend an even number of paths from v using z. Since the labels of the newly labelled vertices have absolute values that are so much larger than previously labelled vertices, we know that they do not break the invariant for previously labelled vertices. We therefore only need to prove the invariant for the newly labelled vertices. More specifically, when a pair of legs are labelled, their labels are so large, that they cannot break the invariant for any previously labelled vertex. We therefore only focus on extending two paths from v using z. If $z = u$ then for the root of a leg r, we know that the rest of the leg receives labels within $\{\lambda(r) - i \mid i \in \{0, \ldots, 2(n-1)\}\}$. For any vertex a on r's path, $\lambda(a) - 2n$ is smaller than any label assigned on a's path, and has a different sign than labels assigned on the other path. The number $\lambda(a) - 2(n-1)2n$ has an absolute value that is larger than the largest previously assigned label. With that we get that a is the only vertex within $\{\lambda(a) - i(2n) \mid i \in \{0, \ldots, 2(n-1)\}\}$. We now deal with the case that $z = w$ very similarly. For any vertex a that lies on one of the two paths that are extended using w, we know that $\{\lambda(a) - i \mid i \in \{1, \ldots, 2(n-1)\}\}$ lies strictly between $\lambda(a)$ and $\lambda(a) - 2n$. There are however no labels that are assigned within this range. It follows that our invariant holds for both cases.

Inductively, we know that the conditions for using weak path extension always hold. The correctness of our algorithm follows directly from the correctness of weak path extension and the fact that it clearly assigns a distinct label to every vertex in the tree. □

With that, we obtain the following theorem.

Theorem 4.5. *For any out-tree T, $m(T) \leq 2$.*

4.2.2 In-trees

We know that an in-tree is a DG if and only if, it is an in-broom. We present a tight bound on the difference-number of all other in-trees. In this section, we show that for any in-tree T, $m(T) \leq \lfloor \frac{n}{2} \rfloor - 1$, and then show that this bound is tight.

Lemma 4.6. *For any in-tree T on n vertices, $m(T) \leq \lfloor \frac{n}{2} \rfloor - 1$*

We provide a very simple algorithm that adds at most $\lfloor \frac{n}{2} \rfloor - 1$ vertices to a given tree, and turns it into a DG, with the original tree as an induced sub-graph.

Algorithm 6: Representing In-Trees as DGs

We start by giving the root of the tree an arbitrary strictly positive label (e.g. 1)[a].

1. Find the label *max*, with the largest absolute value.
2. Find the vertex p with the largest label and at least one unlabelled child v (if p has unlabelled children that are leaves, we select v to be one of those leaves[b]).
3. If v is the first child of p to be labelled, then assign it the label $2\lambda(p)$.
4. If there exists a labelled vertex u with the same relevant neighbourhood as v in T (i.e. v, u are leaves with the same parent), then assign v the same label as u.
5. Otherwise, add a new vertex v', and set $\lambda(v') := 2max+1$ and $\lambda(v) := \lambda(p)+\lambda(v')$.
6. Return to step 1.

[a]This algorithm also works if the root is assigned any non-zero label. Assigning a strictly positive label only simplifies the proof.

[b]It is not necessary to select the leaves first, but it also serves to simplify the proof.

Note here, that all vertices are assigned labels with the same sign, and the labels of non-leaves are distinct. We prove the correctness of this algorithm by induction over the number of completed iterations. The graph induced by the labels of all labelled vertices (including newly added vertices) after the i-th iteration is named T'_i. The sub-tree of vertices of T that are labelled after the i-th iteration is referred to as T_i. Our goal is only to prove that T is an induced sub-graph of T'_n, (T'_n is a DG due to its construction).

Lemma 4.7. *For all $i \in \{1, \ldots, n\}$, after the i-th iteration of Algorithm 6, there exists at most one vertex from T_i with no labelled children, the vertex with the largest label in T'_i, and T_i is an induced sub-graph of T'_i.*

Proof. We prove this inductively over the number of completed iterations. The base case is clear, when the root is labelled it is the only labelled vertex in the graph, and thereby also the vertex with the largest label and the only labelled vertex that may have no labelled children. Furthermore T_1 is T'_1 (so also an induced sub-graph).

Let us assume that the lemma holds for the iterations $1, \ldots, i$. Let us refer to the vertex that is to be labelled as v, and its parent as p.

If v is to be labelled in step 3 of our iteration, then p is the only labelled vertex with no labelled children and thereby also has the largest label. The vertex v receives the label $2\lambda(p)$, which induces the edge vp in T'_{i+1}. Since the label of v is so large, and labels of non-leaves are distinct, we know that it cannot be represented as the sum of any other two labels in T'_{i+1}. This means that T_{i+1} is an induced sub-graph of T'_{i+1}. Since we labelled one of p's children, we know that v is the only labelled vertex that can have no labelled children. The lemma therefore holds for the $i+1$-th iteration.

If v is labelled in step 4 of our iteration, then the fact that T_{i+1} is an induced sub-graph of T'_{i+1} follows directly from the fact that T_i is an induced sub-graph of T'_i. Since v is a leaf, the lemma clearly holds for the $i+1$-th iteration.

If v is labelled in step 5 of our iteration, then we add a vertex v' to T'_{i+1}. Due to the size of the label belonging to v', we know that it must have out-degree 0, since for any other vertex $u \neq v$, $\lambda(v') > \lambda(v') - \lambda(u) > max$. The in-degree of v' is 1. This is because v is the only vertex, with a label large enough so that $\lambda(v) - \lambda(v') > 0$. That means that the addition of the vertex v' induces only the edges vv' and vp in T'_{i+1}. Due to the fact that the label of v is more than twice as large as any of the labels not belonging to v', we know that no other edges are induced between v and the rest of the tree. The tree T_{i+1} is therefore an induced sub-graph of T'_{i+1}. The vertex v now has the largest label in T'_{i+1}. Due to the assumption that the lemma held for the i-th iteration, and that we know that p is the vertex with the largest label that has unlabelled children, we know that no vertex in T_i has no labelled children. Thus v is the only labelled vertex in T_{i+1} that may have no labelled children. The lemma therefore holds for the $i+1$-th iteration. □

By Lemma 4.7, Algorithm 6 correctly represents T as an induced sub-graph of a DG. All that is left to prove, is that Algorithm 6 adds no more than $\lfloor \frac{n}{2} \rfloor - 1$ vertices.

Proof of Lemma 4.6. In order to prove Lemma 4.6, all we need to do is show that Algorithm 6 adds no more than $\lfloor \frac{n}{2} \rfloor - 1$ vertices to any given in-tree. Note that due to step 2, no leaf is ever labelled in step 5. That means every vertex labelled in step 5 of our iteration has at least one child.

Let us assume for the sake of contraposition that Algorithm 6 adds at least $\lfloor \frac{n}{2} \rfloor$ vertices to a given input in-tree $T = (V, E)$, ($n = |V| > 2$). That means that at least $\lfloor \frac{n}{2} \rfloor$ vertices are labelled in step 5 of our iteration. Let $S_i := \{v \in V \mid v \text{ is labelled in step } i\}$, for $i \in \{3,4,5\}$. We know that $S_i \cap S_j = \emptyset$ for $i \neq j$. Due to our assumption, we know that $|S_5| \geq \lfloor \frac{n}{2} \rfloor$. Every element of S_5 has at least one child, and vertices in $|S_5|$ cannot share a child. Since the first child always gets labelled in step three, we know that $|S_3| \geq |S_5|$. We can also see, that the root of the tree is not in S_5, but it also has a child in S_3, meaning $|S_3| > |S_5|$. The root is clearly in neither of these sets. It follows that:

$$\begin{aligned} n &= |V| \\ &\geq |S_3| + |S_5| + 1 \\ &> \left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor + 1 \\ &\geq \frac{n-1}{2} + \frac{n-1}{2} + 1 \\ &= n, \end{aligned}$$

a contradiction. □

We now have a bound for the difference-number of in-trees. We know that the difference-number of any graph is at most m; at first glance it might therefore seem like our bound for in-trees is not tight. This is however not the case.

Lemma 4.8. *There exists a family of in-trees, $\mathcal{F}$ such that for all $T \in \mathcal{F}$*

$$m(T) = \left\lfloor \frac{n}{2} \right\rfloor - 1.$$

Proof. Let us define the family of trees $\mathcal{F}$ as the family of in-spiders, where every leg has length two, except for one leg which is of length one. Every spider in this family with n vertices, has exactly $\frac{n}{2}$ legs. Algorithm 6 adds exactly $\frac{n}{2} - 1$ vertices to represent such an in-spider. Our goal is to show that this is optimal.

Given, is an in-spider $S \in \mathcal{F}$, with $k = \frac{n}{2}$ legs. Let us refer to the children of the root r of S as $c_1, \dots, c_k$. Let S' be a DG with S as an induced sub-graph. The graph S' consists of the vertices of S, and some extra added vertices; we refer to the set of these extra vertices as M. The out-neighbourhood of any c_i in S' is r, as well as some vertices from M. We therefore know that for all c_i, $\lambda(c_i) - \lambda(r) \in \{r\} \cup M$. If there exist indices i, j such that $\lambda(c_i) - \lambda(r) = \lambda(c_j) - \lambda(r)$, then due to the fact that the closed neighbourhoods of $c_1, \dots, c_k$ are distinct, we can deduce that $i = j$. That means that the cardinality of $\{\lambda(c_i) - \lambda(r) \mid i \in \{1, \dots, k\}\}$ is at least k. It follows, since $\{\lambda(c_i) - \lambda(r) \mid i \in \{1, \dots, k\}\} \subseteq \lambda(\{r\} \cup M)$, that $|M| \geq k - 1 = \frac{n}{2} - 1$. □

It follows, that the number of vertices added by Algorithm 6 is therefore minimal.

5. Complexity

In this chapter, we deal with the complexity of difference-labellings. We discuss both the time complexity of identifying DGs, as well as the space complexity of storing graphs as induced sub-graphs of DGs.

5.1 Time Complexity

We formulate the existence of a difference-labelling for a given graph, as a decision problem. For a given graph, we would like to be able to decide, if it is a DG or not. Before looking into an algorithm that would be able to classify DGs, it would make sense to look into the complexity of the problem. Our main goal in this section is to prove that this decision problem is solvable in non-deterministic polynomial time (*NP*). This means that if a graph G is a DG, then there must exist a valid difference-labelling that can be encoded in polynomial space, whose validity can be checked deterministically in polynomial time. It is clear that the validity of a labelling can be tested in polynomial time. Hence, in this section we only show that for any DG there exists a valid difference-labelling with label sizes bound exponentially by the number of vertices. We also discuss how one could calculate exponentially bound labellings for arbitrary graphs.

Limit on Label Size

By showing, that there is an exponential bound, on the size of the labels required to label any given DG, we know that we can encode each of these labels in linear space. The entire labelling can therefore be stored in polynomial space. Thus, our goal in this section is to prove the following theorem.

Theorem 5.1. *For every DG $G = (V, E)$ on n vertices, there exists a difference-labelling λ such that for all $v \in V$, $|\lambda(v)| \leq 4^n$.*

Proof. We assume that G has a difference-labelling λ, and we use a linear programming approach, to construct a valid labelling λ' that fulfills the given bound. This proof follows the proof by J. Kratochvíl, M Miller and H. M. Nguyen, of the same bound for sum-graphs [KMN01]. For the sake of completeness, since this paper is unfortunately very difficult to find, the proof is given in full detail. In general we follow the terminology from 'Linear Programming' by Murty K. G. [Mur83].

For the given DG, on n vertices, we introduce n variables $x_1, \dots, x_n$. We know, that for every edge $v_iv_j \in E$ there exists a vertex $v_k \in V$ such that $\lambda(v_i) - \lambda(v_j) = \lambda(v_k)$. For each edge in G we can add the constraints

$$\begin{aligned} \operatorname{sgn}(\lambda(v_i))x_i - \operatorname{sgn}(\lambda(v_j))x_j - \operatorname{sgn}(\lambda(v_k))x_k \leq 0, \\ -\operatorname{sgn}(\lambda(v_i))x_i + \operatorname{sgn}(\lambda(v_j))x_j + \operatorname{sgn}(\lambda(v_k))x_k \leq 0. \end{aligned}$$

For every non-edge $v_iv_j \notin E$, we know for all $k \in \{1, \dots, n\}$ that $\lambda(v_i) - \lambda(v_j) \neq \lambda(v_k)$. For every non-edge $v_iv_j \notin E$ and every $k \in \{1, \dots, n\}$ we can therefore add the constraint

$$\begin{cases} \operatorname{sgn}(\lambda(v_j))x_i - \operatorname{sgn}(\lambda(v_j))x_j - \operatorname{sgn}(\lambda(v_k))x_k \leq -1, & \text{if } \lambda(v_i) - \lambda(v_j) < \lambda(v_k) \\ -\operatorname{sgn}(\lambda(v_i))x_i + \operatorname{sgn}(\lambda(v_j))x_j + \operatorname{sgn}(\lambda(v_k))x_k \leq -1, & \text{if } \lambda(v_i) - \lambda(v_j) > \lambda(v_k). \end{cases}$$

We also add the constraints [1]

$$x_i \geq 0.$$

We know that setting $x_i = |\lambda(v_i)|$, is a solution to all of our constraints, this linear program (LP) is therefore feasible. We also know, that a solution to our LP, provides a valid difference-labelling, that uses real numbers as labels instead of integers (after returning the signs of the labels). Since this LP is feasible, the solution space of an LP of this form forms a non-empty convex polyhedron in $\mathbb{R}^n$, with each vertex of the polyhedron being a basic feasible solution (BFS). By representing the LP in matrix form, $Ax \leq b$ and letting x be a vertex of our polyhedron (a BFS), we know that there are n linearly independent constraints that hold with equality, and that there is a regular $n \times n$ sub-matrix A' that consists of the rows of A with indices $c_1, \dots, c_n$ and a $b' = (b_{c_1}, \dots, b_{c_n})^{\mathrm{T}}$, such that

$$A'x = b'.$$

We can set A'_i to be A', where the i-th column is replaced by b'. By Cramer's Rule [Rob70], we know that

$$x_i = \frac{\det(A'_i)}{\det(\mathrm{A}')}.$$

It follows that if we set

$$x'_i := x_i \det(A') = \det(A'_i),$$

then x' can also be used to obtain a valid difference-labelling for our graph with rational labels (since we would only multiply the old solution with a factor). With the Leibniz Formula, we also know, since A'_i is an integer matrix, that its determinant is also an integer. The solution x' uses therefore only integer labels. Furthermore we know, that the sum of each row of A'_i has an absolute value of at most 4. For any vector v and any matrix M where the absolute value of any row is at most four, let k be the index of the entry of v with the largest absolute value. We know that

$$|(Mv)_k| = |\sum_{j=0}^{n} M_{kj}v_j| \leq |\sum_{j=0}^{n} |M_{kj}v_j|| \leq |\sum_{j=0}^{n} |M_{kj}v_k|| = |v_k \sum_{j=0}^{n} |M_{kj}| \leq |4v_k|.$$

The absolute value of the largest eigenvalue is therefore at most 4. The absolute value of the determinant of any A'_i is therefore at most 4^n, since the determinant is the product of the eigenvalues.

By setting $\lambda'(v_i) := \operatorname{sgn}(\lambda(v_i))x_i$, we have thus created a valid difference-labelling for our graph, that uses labels with absolute values of at most 4^n.

□

[1]This is to represent our linear program in canonical form (not notation from 'Linear Programming')

Time Complexity

We know that the decision problem of whether a given graph is a DG or not, is in NP. The proof of this bound labelling, is however purely existential. Here we discuss the time complexity of actually calculating a valid labelling that is bound by 4^n.

In our proof of Theorem 5.1, we use an existing labelling, and linear programming, to construct a labelling with exponentially bound labels. More importantly we did not specifically use the labels assigned by that labelling, but only their signs and certain relationships between each triplet of labels. For any graph class, if we have an algorithm that can find these relations in polynomial time, then we can use the algorithm from Theorem 5.1 in order to find an exponentially bound labelling for our graph in polynomial time, since one can find BFSs in polynomial time.

As a small example, Algorithm 4 for the labelling of out-trees could be slightly altered to not explicitly find labels for the vertices, but only calculate the sign of each label, and the required relations between all the vertices. This could then be used to find a labelling for any out-tree, so that no label is larger than 4^n.

5.2 Space Complexity

In the context of computer science, one might want to represent certain graphs as induced sub-graphs of DGs. In order to do this, one would have to store the labels. It would therefore be beneficial to minimize the size of the labels used to label the graph, as well as the number of added nodes used to represent a non-DG as a DG. In this section, we present a polynomial-time algorithm that stores any directed graph on n vertices, with m arcs as an induced sub-graph of a DG on $n+m$ vertices, and provides a difference-labelling for the given graph with label sizes in $O(n^2)$. Due to the number-theoretical nature of our algorithm, we first present a construction that requires very little prior knowledge, in order to explain the intuition behind our algorithm. We then adapt our algorithm for the storage of undirected graphs as sum-graphs, beating the previously best known bound on label sizes of $O(n^3)$ (while adding the same number of vertices).

Cycles

As we have seen in Theorem 4.2, the difference-number of cycles is one. In the proof of Theorem 4.2, exponential labels are used. It is however possible to use linear labels to represent a cycle as a DG.

Let G be a graph on $n+1$ vertices $\{v_0, \dots, v_{n-1}, w\}$. We assign labels which implicitly define the edge set E of G. We then show that G has an induced cycle of length n. More importantly, we only use labels of linear size.

Define a labelling λ on G as $\lambda(v_i) = n-1+i$ and $\lambda(w) = -1$. Since $\lambda(v_i) - \lambda(v_{i+1}) = \lambda(w)$ for $i \in \{0, \dots, n-2\}$, and $\lambda(v_{n-1}) = n-1+n-1 = 2\lambda(v_0)$, our labelling induces all edges required for a cycle on $v_0, \dots, v_{n-1}$. All that is left, is to check that this cycle is induced. We know that the presence of the label -1 in the graph only induces an edge from each node on the cycle to the one that follows it (and to w). Any other edge between two vertices in our cycle must therefore be induced by three nodes in the cycle. However the sum of any two labels in the cycle (other than $\lambda(v_n)$), is larger than the largest label in the graph, meaning that there exist no other edges of the form v_iv_j other than those induced by $\lambda(w)$ and $v_{n-1}v_0$. The nodes $v_0, \dots, v_{n-1}$ therefore induce a cycle of length n in G.

Paths

In Theorem 2.4, we discuss the fact that paths are DGs. Due to the structure of paths, the only way to label $(v_1, \dots, v_n)$ is to set the label of v_i to be double the label of v_{i+1}. This always produces exponentially large labels. However, if we add an extra vertex to a path, it is possible to use only linear sized labels to label the new graph.

Given a graph G with vertex set $V = \{v_0, \dots, v_{n-1}, w\}$, we define the edge set of G implicitly by labelling the vertices of V, and then show that G has an induced path of length $n-1$. Let us define the labelling $\lambda(v_i) = n+i, \lambda(w) = -1$. This labelling induces the edges v_iv_{i+1} for $i \in \{0, n-2\}$. All that is left, is to show is that the path $(v_1, \dots, v_{n-1})$ is an induced sub-graph of G. Let $v_i, v_j, u \in V$ be vertices with $\lambda(v_i) + \lambda(u) = \lambda(v_j)$. If $u = w$, then $j = i-1$, so this edge should be part of the induced path. If $u \neq w$, then $\lambda(u) \geq n$, so $\lambda(v_i) + \lambda(u) \geq 2n$, there is however no vertex with a label that large. This means that $G[\{v_0, \dots, v_{n-1}\}]$ is a path of length n.

Target-Set-Difference-Digraphs

We define a Target-Set-DG (*TSDG*) $S = (L, T)$ as a set $L \subset \mathbb{Z}$ of labels and a target-set $T \subset \mathbb{Z}$. Each element in L denotes a vertex in the graph S. There exists an arc from u to v in S if and only if $u - v \in T$.

Theorem 5.2. *For a given TSDG $S = (L = \{\ell_1, \dots, \ell_{|L|}\}, T = \{t_1, \dots, t_{|T|}\})$, S can be represented as an induced sub-graph of a DG using at most $|T|$ extra vertices, with the largest label having the same asymptotic size as the largest label in $L \cup T$.*

Proof. Let us define a graph G on $|L| + |T|$ vertices. We give a difference-labelling on the vertices $\{v_1, \dots, v_{|L|+|T|}\}$ of G, which implicitly defines the edge set, ensuring that G is a DG. Let the labelling $\lambda(v_i) = \ell_i$ $(i \in \{1, \dots, |L|\})$ and $\lambda(v_{|L|+i}) = t_i$, $(i \in \{1, \dots, |T|\})$. It is clear to see that all edges that are in S are induced by this labelling on G. More specifically, for $i, j \leq |L|, k > |L|, \lambda(v_i) - \lambda(v_j) = \lambda(v_k)$ if and only if $\ell_i - \ell_j = t_{|L|-k}$. It follows, that if we were to ensure that there are no indices $i, j, k \leq |L|$ with $\lambda(v_i) - \lambda(v_j) = \lambda(v_k)$, then $G[\{v_1, \dots, v_{|L|}\}] = S$.

We set $max = \max\{L \cup T\}$ and $min = \min\{L \cup T\}$. We then define a new labelling λ' on the vertex set, with

$$\lambda'(v_i) := \begin{cases} \lambda(v_i) + max - 2min + 1 & \text{If, } i \leq |L| \\ \lambda(v_i) & \text{If, } i > |L| \end{cases}.$$

With this new labelling, we know for indices $i, j \leq |L|, k > |L|$ that

$$\begin{aligned} & \lambda'(v_i) - \lambda'(v_j) = \lambda'(v_k) \\ \iff & \lambda(v_i) + max - 2min + 1 - (\lambda(v_j) + max - 2min + 1) = \lambda(v_k) \\ \iff & \lambda(v_i) - \lambda(v_j) = \lambda(v_k), \end{aligned}$$

so $S \subseteq G[\{v_1, \dots, v_{|L|}\}]$. We also know, for $i, j, k \leq |L|$, if $\lambda'(v_i) - \lambda'(v_j) = \lambda'(v_k)$, then

$$\begin{aligned} \lambda(v_i) - \lambda(v_j) &= \lambda(v_i) + max - 2min + 1 - (\lambda(v_j) + max - 2min + 1) \\ &= \lambda'(v_i) - \lambda'(v_j) \\ &= \lambda'(v_k) \\ &= \lambda(v_k) + max - 2min + 1 \\ &\geq min + max - 2min + 1 \\ &= max - min + 1 > max - min, \end{aligned}$$

which is not possible. Thus $S = G[\{v_1, \dots, v_{|L|}\}]$, and G is a DG due to its construction. □

This is useful, since we now know that in order to obtain a space efficient difference-labelling of a given graph, all we have to do is find a space efficient representation of the given graph as a TSDG. We can also use such a representation to bound the difference-number of any given graph from above, since clearly the difference-number of a given graph is bound by the minimum size of the target-set of its TSDG representation. This is a great feature, since TSDGs are much easier to work with than DGs.

5.2.1 Directed Acyclic Graphs

We can now deal with the space complexity of storing any given DAG as a sub-graph of a DG. It is beneficial to assign the labels in such a way, that one could retroactively identify which vertices belong to the original graph, and which vertices were added, in order to represent it as a DG. In our construction, we guarantee that vertices belonging to the original graph receive strictly positive labels, while labels that are added receive strictly negative labels. Our algorithm also gives a polynomial bound on label sizes.

Lemma 5.3. *For any $n \in \mathbb{N}$ there exist n elements $e_1, \dots, e_n \in \mathbb{N}$ such that for any four indices i, j, k, l with $i \neq j, k \neq l$*

$$\sum_{c=j}^{i} e_c = \sum_{c=l}^{k} e_c \text{ only if } i = k, j = l,$$

and these elements are in $O(n^2)$

Proof. We start by considering an odd n and then generalise our construction for even n. For a given odd n, we set

$$p = \min\{i \in \mathbb{N} \mid \frac{n+1}{n-1} i > i + n - 1\}.$$

Let

$$e_i = p + i - 1.$$

Let us assume for the sake of contraposition, that there exist indices i, j, k, l with $i \neq j, k \neq l, i \neq k, j \neq l$ and

$$\sum_{c=j}^{i} e_c = \sum_{c=l}^{k} e_c.$$

Without loss of generality, we can assume that $j < l$. If $i \geq k$, then one sum is a strict sub-sum of the other, and since we only sum positive values, the equality cannot hold. We can therefore assume that $i < k$. By removing terms that are present in both sums, we can obtain indices i', l' with $i' < l'$ such that

$$\sum_{c=j}^{i'} e_c = \sum_{c=l'}^{k} e_c,$$

with at least one (so therefore both) of these sums being non-zero. Now, all we show is that the chosen elements are so large, that the longer of the two sums must also be strictly larger. If $k - l' \geq i' - j$ then

$$\sum_{c=j}^{i'} e_c \leq \sum_{c=j}^{i'} e_{i'} = (i' - j) e_{i'} \leq (k - l') e_{i'} < (k - l') e_{l'} = \sum_{c=l'}^{k} e_{l'} \leq \sum_{c=l'}^{k} e_c.$$

We can therefore assume that $k - l' < i' - j$, meaning that $i' - j \geq 2$. With that we know that

$$\sum_{c=j}^{i'} e_c > \sum_{c=j}^{i'} p = (i' - j)p \text{ and}$$
$$\sum_{c=l'}^{k} e_c \leq \sum_{c=l'}^{k} p + n - 1 = (k - l')(p + n - 1).$$

It follows that $(i' - j)p < (k - l')(p + n - 1)$. In the case that $(i' - j) > n/2$ since $(i' - j) + (k - l') \leq n$, we obtain

$$p + n - 1 > \frac{(i' - j)}{(k - l')}p \geq \frac{n + 1}{2(k - l')}p \geq \frac{n + 1}{n - 1}p,$$

which is a contradiction to our choice of p. We can therefore assume that $(i' - j) < n/2$. With that we know that

$$\begin{aligned} p + n - 1 >& \frac{i' - j}{k - l'}p \\ \geq& \frac{i' - j}{i' - j - 1}p, \end{aligned}$$

where we can use the fact that for all $x \in \mathbb{N}$, $\frac{x+1}{x} > \frac{x+2}{x+1}$ to obtain

$$\begin{aligned} \frac{i' - j}{i' - j - 1}p >& \frac{i' - j + 1}{i' - j}p \\ >& \dots \\ >& \frac{\lceil n/2 \rceil}{\lceil n/2 \rceil - 1}p = \frac{n + 1}{n - 1}p, \end{aligned}$$

which is a contradiction to our choice of p. It follows that the first part of the lemma holds for any odd n. For any even n we can select $n + 1$ elements such that the first part of the lemma holds. By removing the largest of these elements, it is clear that the remaining n elements satisfy the first part of the lemma. Furthermore, we know that the largest of the selected elements is

$$e_n = p + n - 1,$$

where (due to the way we handle even numbers)

$$\begin{aligned} p \leq& \min\{i \in \mathbb{N} \mid \frac{n + 1}{n}i > i + n\} = \min\{i \in \mathbb{N} \mid ni + i > ni + n^2\} \\ =& \min\{i \in \mathbb{N} \mid i > n^2\} = n^2 + 1. \end{aligned}$$

It follows that all of the selected elements are in $O(n^2)$. The lemma therefore holds for all n. □

We now use these elements in order to create a set A such that setting A to be the label-set in a TSDG allows us to add a distinct label into our target-set in order to induce each edge.

Theorem 5.4. *A DAG $G = (V, E)$ on n vertices with m edges can be represented as an induced sub-graph of a DG on $n + m$ vertices with label sizes in $O(n^3)$.*

Proof. By Theorem 5.2, we know that if we represent G as a TSDG using a target-set of size at most m, and labels with sizes in $O(n^3)$, then this theorem also holds. Our goal is therefore to construct a TSDG $S = (L, T)$ such that $G = S$. Let $e_1, \ldots, e_{n-1}$ be $n - 1$ numbers as in Lemma 5.3. Any given DAG has a topological ordering of its the vertices. Let $v_1, \ldots, v_n$ be a topological ordering of G. We use this topological ordering to ensure that arcs in S only go from a smaller label, to a larger label. Let

$$\lambda(v_i) := 1 + \sum_{c=1}^{i-1} e_c \text{ and}$$

$$L := \lambda(V).$$

For any edge $v_iv_j \in E$ we know that

$$\lambda(v_i) - \lambda(v_j) = \sum_{c=1}^{i-1} e_c - \sum_{c=1}^{j-1} e_c = \sum_{c=i}^{j-1} e_c.$$

It follows with Lemma 5.3, that for any $v_iv_j, v_kv_l \in E$

$$\lambda(v_i) - \lambda(v_j) = \lambda(v_k) - \lambda(v_l),$$

means that $i = j, k = l$. We can therefore set

$$T := \{\lambda(v_i) - \lambda(v_j) \mid v_iv_j \in E\}.$$

With this choice of T, it is clear that the induced TSDG S is isomorphic to G. As the sum of at most n elements that are all in $O(n^2)$, every label we use (either in L or in T) is in $O(n^3)$. □

This very simple construction is by no means optimal, as we see in Section 5.2.2, however it serves an important demonstrational purpose. In order to represent a DAG on n vertices as a TSDG, what we have essentially done is construct a set of n elements, such that if you subtract any element from any smaller element, you get a distinct negative value. What this means, is that in this specific set we have constructed, if you subtract any element from a larger element, you get a distinct positive value. This means that for a set A that is constructed as above, for all $x \neq y, a \neq b \in A, x - y = a - b$ if and only if $x = a, y = b$. We refer to this set property as having *distinct differences*. If we have any set with distinct differences, then we can use the exact same scheme as above to represent any graph as a TSDG, by labelling the vertices of our graph with elements of the set, and adding a new element in our target-set for each edge we want to induce. Note that any set A with distinct differences, also has *distinct additions*, meaning that for all $x \neq y, a \neq b \in A, x + y = a + b$ if and only if $x = a, y = b$. One could verify this simply by rearranging terms, since for $x \neq y, a \neq b, x + y = a + b$ if and only if $x - b = a - y$. However, not all sets with distinct additions have distinct differences. This stems purely from the fact that, $x + x = a + b$ if and only if $x - b = a - x$, so there exist sets like $\{0, 1, 2\}$ that have distinct additions, but not distinct differences. It might have just become clear, that in this manner, we can not only provide a labelling scheme to represent digraphs as DGs, but also to represent graphs as sum-graphs. We discuss the algorithm for representing digraphs as DGs in the next section, and then adapt that algorithm for sum-graphs.

5.2.2 General Digraphs

Sidon [Sid32] defines a B_2 sequence as a sequence of positive integers $a_1 < a_2 < \dots$ such that for all $1 \leq i \leq j$, the sums $a_i + a_j$ are distinct. If you take any subset of a B_2 sequence, it forms a set with distinct differences. There are a multitude of constructions of B_2 sequences, which vary greatly in space and time complexity. Here we use a B_2 sequence construction from Lindenström [Lin98], but any construction of (in)finite B_2 sequences is sufficient.

Algorithm 7: Generating a B_2 Sequence of Size n in Poly. Time

1. Find the smallest prime p strictly larger than n.
2. Find a primitive root ζ modulo p.
3. Calculate a B_2 sequence of size $p-1$ by ordering the elements of the set

$$\{pi + (p-1)\zeta^i \mod p(p-1) \mid 1 \leq i \leq p-1\}.$$

Claim. Algorithm 7 generates a B_2 sequence of size n in polynomial time, with elements in bound by $2n(2n-1)$ for all n and elements that tend towards being bound by $n(n-1)$ as n goes to infinity

Proof. Most of the complexity is completely in the first two steps. In order to find the next prime number, we can very simply iteratively test numbers larger than n for primality until we find a prime number. It is known, that we can (deterministically) determine, in polynomial time, if a number is prime [AKS04]. It is known that for any $n > 1$, there exists a prime number between n and $2n$ [Tch52]. It follows, that we can find the next prime number in polynomial time.

For the second step, we can simply test all elements in $\mathbb{Z}/p\mathbb{Z}$ until we find a primitive root. There are many algorithms to test if a number is a primitive root in $\mathbb{Z}/p\mathbb{Z}$, but even the naive algorithm of testing for each element j that $j^i \neq 1$ for all $1 \leq i < p-1$ can be implemented in polynomial time. The fact that Algorithm 7 generates a B_2 sequence is proven by Lindenström [Lin98].

For small n, we know that p is bound by $2n$, meaning that $p(p-1)$ is bound by $2n(2n-1)$. Since all elements of the created sequence are in $\mathbb{Z}/p(p-1)\mathbb{Z}$, we know that for small n, the elements of the generated B_2 sequence are bound by $2n(2n-1)$. As n tends to infinity, we know that p tends towards n, since the quotient of two consecutive primes tends to one, thus $p(p-1)$ tends towards $n(n-1)$. Since all elements of the created sequence are in $\mathbb{Z}/p(p-1)\mathbb{Z}$, we know that as n goes to infinity, the elements of the generated B_2 sequence tend towards being bound by $n(n-1)$. □

For any graph on n vertices that we would like to represent as a TSDG, we can use Algorithm 7 to generate a B_2 sequence of size at least n. We can set the label-set of our TSDG to be any subset of our B_2 sequence of size n. For each edge uv we would like to add to our TSDG, there is exactly one integer $(u-v)$ that we can add into our target-set that would induce it, and that integer induces no other edges, due to the label-set having distinct differences. This would allow us to represent any graph as a TSDG in polynomial time, with labels in $O(n^2)$. The following theorem is implied by Theorem 5.2

Theorem 5.5. *Any directed graph $G = (V, E)$ is an induced sub-graph of a DG G' with $|V| + |E|$ vertices, that can be labelled using labels in $O(n^2)$.*

With the goal of improving on this asymptotic bound, a plausible starting point would be to try to construct a set with distinct differences of size n with numbers in $o(n^2)$. Unfortunately, this is in general not possible.

Theorem 5.6. *As n goes to infinity, every set of size n with distinct differences tends to have elements of size $\Omega(n^2)$.*

Proof. Let $\Psi(n)$ denote the size of the largest set with distinct differences and numbers with absolute values smaller or equal to n. Let the function $\Psi^{-1}(x)$ be the smallest n such that there exists a set of size x with distinct differences, and elements with absolute values less than or equal to n.

Define $\Phi(n)$ as the maximal x such that there exists a B_2 sequence $a_1, a_2, \ldots$ with $a_x \leq n < a_{x+1}$. In 1941 Erdős and Túran [ET41] proved that

$$\lim^{\sup} \frac{\Phi(n)}{\sqrt{n}} \leq 1.$$

Let $\Phi^{-1}(x)$ be the smallest number n such that there is a B_2 sequence of size x with elements smaller or equal to n. Since the function $\Phi(n)$ is an increasing function, we know

$$\lim^{\inf} \frac{\Phi^{-1}(x)}{x^2} \geq 1.$$

Let us assume for the sake of contraposition, that $\lim^{\inf} \Psi^{-1}(x)/f(x) \leq 1$ for an $f \in o(n^2)$. We know that every set with distinct differences and only positive elements can be ordered as a B_2 sequence. For every set with distinct differences of size n, we can subtract the smallest element of the set from all elements, to obtain a set of size n with distinct differences, and only positive integers that are in $O(2f) \subseteq o(n^2)$. It would follow, that

$$\lim^{\inf} \frac{\Phi^{-1}(x)}{f(x)} \leq \lim^{\inf} 2\frac{\Psi^{-1}(x)}{f(x)} \leq 2.$$

This implies that $\lim^{\inf} \Phi^{-1}(x)/x^2 = 0$, a contradiction. It follows that in general one requires values in $\Omega(n^2)$ to create a set with distinct differences of size n. □

It follows, that in order to achieve better asymptotic bounds, one would need to take a more graph theoretical approach by using specific features of the graph, although there could be some very simple improvements even to this number theoretical approach that we discuss in Chapter 6.

5.2.3 A Remark on the Space Complexity of Sum-Labelling

In a recent paper titled 'The Space Complexity of Sum-Labelling' [FG21], H. Fernau and K. Gajjar show that every graph on n vertices with m edges of minimum degree of at least one, can be made a sum-graph by adding at most m isolated vertices. This new graph can be labelled using labels bound by $8n^3$. In the same paper they also show, that a graph on n vertices and m edges that is d-degenerate, can be made into a sum-graph by adding at most m isolated vertices. This new graph can be labelled using labels bound by $12dn^2$. In both cases, they present algorithms for constructing such graphs, as well as a labelling that satisfies their respective bound. In this section, we take a number theoretical approach to provide a polynomial time algorithm that adds m isolated vertices to a graph on n vertices, with m edges in order to represent it as a sum-graph. We provide a labelling for this new graph, where for all n the labels are bound by $24n^2$, and as n goes to infinity, the labels tend to be bound by $6n^2$.

Algorithm 8: Representing Graphs as Sum-Graphs

Given an undirected graph $G = (\{v_1, \dots, v_n\}, E)$

1. Use Algorithm 7 to generate a B_2 sequence $a_1, \dots, a_n$.
2. Let $L := \{\ell_{v_iv_j} \mid v_iv_j \in E\}$.
3. Define the graph $G' := (V \cup L, E)$ and the labelling

$$\lambda(v) := \begin{cases} a_i + 2(a_n + 1), & v = v_i \\ \lambda(v_i) + \lambda(v_j), & v = \ell_{v_iv_j} \end{cases}$$

Algorithm 8 adds exactly m extra vertices to the input graph, and uses strictly positive labels to label all vertices. Analogous to the previous section, we know that Algorithm 8 runs in polynomial time. We must therefore only prove the following claim.

Claim. For a given graph $G = (V, E)$, the labelling generated by Algorithm 8 is valid and the labels used are bound by $24n^2$ for any n, and tend towards being bound by $6n^2$ as n goes to infinity.

Proof. In order to show that this labelling is a valid sum-labelling, we must first show that the edges induced by this labelling are exactly the edges in E, and that assigned labels are distinct. For an edge $\{v_i, v_j\} \in E$, we know $\lambda(v_i) + \lambda(v_j) = \lambda(\ell_{v_iv_j})$. Every edge in E is therefore induced by this labelling.

Let us assume for the sake of contradiction that there exists an edge $\{v, w\} \notin E$ that is induced by our labelling. Neither v nor w can be in L, since if w.l.o.g. v is in L, then

$$\begin{aligned} \lambda(v) + \lambda(w) &\geq 2a_1 + 4(a_n + 1) + \lambda(w) \geq 3a_1 + 6(a_n + 1) \\ &> 6(a_n + 1) > 2a_n + 4(a_n + 1) = \lambda(v_n) + \lambda(v_n) > \max \lambda(V \cup L), \end{aligned}$$

a contradiction.

So both v and w are in V. But since we only add a vertex with the label $\lambda(v) + \lambda(w)$ if $\{v, w\} \in E$, it follows that there must exist an $\{x, y\} \in E$, with $x, y \in V$, and $\lambda(x) + \lambda(y) = \lambda(v) + \lambda(w)$. Thus, there exist indices i, j, k, l with $\lambda(v_i) + \lambda(v_j) = \lambda(v_k) + \lambda(v_l)$.

It follows, that

$$a_i + a_j + 4(a_n + 1) = a_k + a_l + 4(a_n + 1), \text{ thus}$$
$$a_i + a_j = a_k + a_l,$$

meaning (due to $a_1, \ldots, a_n$ forming a B_2 sequence), that $\{v, w\} = \{x, y\} \in E$, a contradiction. Our sum-labelling is therefore a valid sum-labelling for G'.

All that is left to show is that the labels are bound by $24n^2$. We know that

$$\max \lambda(V \cup L) < 2\lambda(v_n) + 4(a_n + 1) = 6a_n + 4 < 6(2n(2n-1)) + 4 < 24n^2.$$

As n goes to infinity, $\max \lambda(V \cup L)$ then goes to $6(n(n-1)) + 4 < 6n^2$. □

This beats the best known bound for the storage of general graphs as sum-graphs. With that we can conclude this chapter with the following theorem.

Theorem 5.7. *For every undirected graph G, on n vertices with m edges, there exists a sum-graph that can be obtained by adding m isolated vertices to G, and has a labelling with label sizes in $O(n^2)$.*

In Theorem 5.5 we prove that this construction of sets with distinct differences is optimal, since any set A with distinct differences can be used to create a B_2 sequence with elements of the same asymptotic size as A. Since B_2 sequences of size n require elements in $\Omega(n^2)$ (for large enough n), there cannot be an algorithm that generates sets with distinct differences with elements of asymptotically smaller sizes. As far as we know, this is not necessarily the case for sets with distinct additions. The reason for this is that B_2 sequences also require that for any three elements $x, y, z, x + x = y + z$ if and only if $x = y = z$. For a set to have distinct additions, this is not the case, meaning it is entirely possible that for any n one could construct a set with distinct additions of n elements with sizes in $o(n^2)$, although we conjecture that this is not the case.

6. Conclusion

In this thesis, we begin by studying the structure of DGs and discovering some forbidden substructures in DGs. We then present conditions that are both necessary and sufficient for a rooted tree to be a DG. We move on to show that deciding if a given graph is a DG is in NP, as well as provide space-efficient polynomial-time algorithms for the storage of graphs as induced sub-graphs of DGs. There are still, however, a multitude of open questions.

Classification

In this thesis, we present some forbidden substructures in DGs. We subsequently use these substructures to provide a complete dichotomy of rooted forests. Nevertheless, there is still quite a lot to uncover. In Section 3.2 we present the strong lakes condition, although we never specifically require this stronger statement. To clarify, for our purposes the lakes condition would have completely sufficed. However, we believe the strong lakes condition could prove invaluable in the classification of directed trees, since if a given tree is a DG then the strong lakes condition provides a large list of maximal sub-trees that must also be DGs.

We mention that there may be an infinite number of forbidden substructures in difference graphs. It is, however, also entirely plausible that there be only very few forbidden substructures, making the question of classifying DGs quite a simple one. This direction did not seem advisable for a thesis, but it could be beneficial to work on finding forbidden substructures, in parallel with classifying specific graph classes.

Difference-Number

We present an upper bound on the difference-number of both in and out-trees. We prove, that our bound on the difference-number of in-trees is a tight one. However, we have not managed to present an out-tree whose difference-number matches our upper bound. Since we prove that the upper bound on the difference-number of out-trees is two and that the lower bound is one, one would either need to find an example of an out-tree that requires two extra vertices, or prove that any out-tree can be represented as an induced sub-graph of a difference graph by adding at most one vertex. We have made many attempts at proving that the upper bound is one, unfortunately to no avail. At this point, however, neither an affirmative nor a negative result would be unexpected.

An interesting notion that was proposed to us during the research phase of this thesis, is the existence of a family of graphs with maximal difference-number. We already prove that the difference-number of any graph is at most the number of edges, but have not been able to present a family of graphs with that high a difference-number. It is for this reason that we conjecture that there exists a tighter bound on the difference-number of graphs, although we know with Lemma 4.8 that there cannot be any asymptotic improvements on this bound.

Complexity

The first question that we tried to work on, but unfortunately could not prove, is the time complexity of deciding if a graph is a DG. It is conjectured, that deciding if a graph is a sum-graph is NP-complete. Similarly, we conjecture that deciding if a digraph is a DG is NP-complete. It would however be by no means surprising if this were not the case, since DGs have a much stronger structure than sum-graphs.

On the space complexity of storing graphs as induced sub-graphs of DGs, we provide a number theoretical algorithm for storing any digraph as an induced sub-graph of a DG, that uses no property about the given graph. We show that our approach provides an optimal such construction. However, there are very simple optimisations that one could consider to improve on our bound. For instance, one could even use our construction while ensuring that the vertices in the graph that receive the largest labels do not share an edge. This alone can reduce the sizes of labels used by up to half. Another very simple way to optimise our approach would be to consider that in most graphs, we need not use a set with distinct differences to label the vertex set. The difference between $\lambda(u)$ and $\lambda(v)$ need only be distinct, if the edge uv is present. It follows, that for most graphs it is entirely possible to construct sets with smaller elements, such that a given graph can be represented as a TSDG where every edge is induced by a distinct element in the target-set.

Bibliography

[AKS04] M. Agrawal, N. Kayal, and N. Saxena. PRIMES Is in P. *Annals of Mathematics*, 160(2):781–793, 2004.

[AS92] N. Alon and E. R. Scheinerman. Generalized Sum Graphs. *Graphs and Combinatorics*, 8(1):23–29, March 1992.

[BLTD90] J. Boland, R. Laskar, C. Turner, and G. Domke. On Mod Sum Graphs. *Congr. Numer., 70*, pages 131–135, 1990.

[Cha93] G. J. Chang. Strong Sum Graphs. *Bull. Inst. Combin. Appl., 7*, pages 47–52, 1993.

[EG84] R.B. Eggleton and S.V. Gervacio. Some Properties of Difference Graphs. *Ars Combin*, 1984.

[Ell93] M. N. Ellingham. Sum Graphs From Trees. *Ars Combinatoria*, 35:335–349, 1993.

[ET41] P. Erdös and P. Turán. On a Problem of Sidon in Additive Number Theory, and on some Related Problems. *Journal of the London Mathematical Society*, s1-16(4):212–215, 10 1941.

[FG21] H. Fernau and K. Gajjar. The Space Complexity of Sum Labelling. In Evripidis Bampis and Aris Pagourtzis, editors, *Fundamentals of Computation Theory*, pages 230–244, Cham, 2021. Springer International Publishing.

[Gal13] J. Gallian. A Dynamic Survey of Graph Labeling. *J. Combin*, 17, 01 2013.

[Ger82] S.V. Gervacio. Difference Graphs. *Proceedings of the Second Franco-Southeast Asian Mathematics Conference*, 1982.

[Har90] F. Harary. Sum Graphs and Difference Graphs. *Congr. Numer., 72*, pages 101–108, 1990.

[Har94] F. Harary. Sum Graphs Over All the Integers. *Discrete Mathematics*, 124(1):99–105, 1994.

[HV09a] S.M. Hegde and Vasudeva. On Mod Difference Labeling of Digraphs. *AKCE International Journal of Graphs and Combinatorics*, 6(1):79–84, 2009.

[HV09b] S.M. Hegde and Vasudeva. Some Structural Properties of Mod Difference Digraphs. *AKCE International Journal of Graphs and Combinatorics*, 2009.

[KMN01] J. Kratochvíl, M. Miller, and H. M. Nguyen. Sum Graph Labels - An Upper Bound and Related Problems. *Proceedings AWOCA 2001*, pages 126–131, 2001.

[Lin98] B. Lindström. Finding Finite B2-Sequences Faster. *Mathematics of computation*, 1998.

[Mur83] K.G. Murty. *Linear Programming.* Wiley, 1983.

[Rob70] S. M. Robinson. A Short Proof of Cramer's Rule. *Mathematics Magazine*, 43(2):94–95, 1970.

[Sid32] S. Sidon. Ein Satz Über Trigonometrische Polynome und Seine Anwendung in der Theorie der Fourier-Reihen. *Mathematische Annalen*, 106:536–539, 1932.

[Son04] M. Sonntag. Difference Labelling of Digraphs. *Discussiones Mathematicae Graph Theory*, 24(3):509–527, 2004.

[SS20] B. Sooryanarayana and S. P. D Silva. Mod Difference Labeling of Some Classes of Digraphs. *Malaya Journal of Matematik*, 8(1):32–36, 2020.

[Tch52] P. Tchebichef. Mémoire sur les nombres premiers. *Journal de Mathématiques Pures et Appliquées*, pages 366–390, 1852.

[TT13] A. Tiwari and A. Tripathi. On the Range and Size of Sum Graphs and Integral Sum Graphs of a Given Order. *Discrete Applied Mathematics*, 161(16):2653–2661, 2013.