Maximum Linear Arrangement of Directed Graphs

by

Kathryn Leeann Nurse

B.Sc., University of Waterloo, 2005

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in the School of Computing Science Faculty of Applied Sciences

© Kathryn Leeann Nurse 2019 SIMON FRASER UNIVERSITY Summer 2019

Copyright in this work rests with the author. Please ensure that any reproduction or re-use is done in accordance with the relevant national copyright legislation.
Approval

Name: Kathryn Leeann Nurse
Degree: Master of Science (Computing Science)
Title: Maximum Linear Arrangement of Directed Graphs
Examiner Committee: Chair: Joseph Peters
                    Professor
Pavol Hell
Senior Supervisor
Professor
Matthew DeVos
Supervisor
Associate Professor
Department of Mathematics
Igor Shinkar
Internal Examiner
Assistant Professor

Date Defended: August 02, 2019
Abstract

We study a new problem on digraphs, Maximum Directed Linear Arrangement (MaxDLA). This is a directed and maximization variant of the well-studied Minimum Linear Arrangement (MinLA) problem. We relate MaxDLA to the Maximum Directed Cut (MaxDiCut) problem by bounding each in terms of the other. We prove that both MaxDiCut and MaxDLA are NP-Hard for planar digraphs. By contrast, the undirected Maximum Cut problem is known to be polynomial on planar graphs. We present a polynomial algorithm solving MaxDLA on orientations of bounded-degree trees, and, as a by-product, a polynomial algorithm for MinLA on graphs $G$ when $G$ is a bounded-degree tree. This complements the known fact that MinLA is polynomial-time solvable on trees. Finally, we study maximization analogues of Harper’s celebrated isoperimetric inequality for hypercubes. We study tournaments, orientations of graphs with maximum degree at most two, and transitive acyclic digraphs.

Keywords: graph layout problem; minimum linear arrangement; optimal linear arrangement; maximum directed cut; maximum cut; edge isoperimetric problem
To Mitch, Jack, and Lily, the light and colours of my life.

To Mike Pridham, I wish you could read this. To Paula Pridham, my best friend. To Hannah and Owen, who have their father’s smile.
I would like to thank Dr. Pavol Hell and Dr. Matthew DeVos, my supervisors, for their insightful guidance, and kind, patient encouragement. How lucky I am to have been able to learn from both of you over these past two years. Without both of you, this thesis would not have been possible.

Thank you to Dr. Igor Shinkar for his thoughtful suggestions which helped improve the final version of this work.

Thank you to my friends, labmates, and teachers for sharing inspiring ideas, and for the warm and encouraging atmosphere in which to study.

I acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC), SFU’s Faculty of Applied Sciences, SFU’s School of Computing Science, and research assistantships provided by my supervisors.
# Table of Contents

Approval ii  
Abstract iii  
Dedication iv  
Acknowledgements v  
Table of Contents vi  
List of Figures viii  

1 Introduction 1  
0.1 Why Study Maximum Directed Linear Arrangement: A History 1  
0.2 A New Problem: Maximum Directed Linear Arrangement 4  
0.3 Definitions 6  
0.4 Organization of Following Chapters 13  

2 Complexity of Related Problems 14  
2.1 Maximum Cut 14  
2.2 Minimum Linear Arrangement 16  
2.3 Maximum Directed Cut 18  

3 Maximum Directed Linear Arrangement 25  
3.1 Complexity and Bounds 25  
3.2 The Weighted Maximum Linear Arrangement Problem, and the Signature of an Arrangement 29  
3.3 Bounded-Degree Trees 31  
3.4 Complements of Bounded-Degree Trees 36  

4 Nested-Minimum and Nested-Maximum Graphs 38  
4.1 Harper’s Isoperimetric Inequality for the Hypercube: The Hypercube is Nested-Minimum 38  
4.2 Nested-Maximum Digraphs 41
# List of Figures

| Figure 1.1 | Two solutions to MinLA for $C_4$ | 2 |
| Figure 1.2 | An arrangement of $C_4$ containing nested solutions to the EIP. | 4 |
| Figure 1.3 | Two arrangements of the directed path of length three. | 4 |
| Figure 1.4 | Top: A maximum arrangement of $C_4$ showing the (non-increasing) level of each vertex. Bottom: The corresponding minimum arrangement of $C_4$. | 5 |
| Figure 1.5 | Top: A maximum arrangement of an orientation of $K_4$ showing the (non-increasing) level of each vertex. Bottom: The corresponding minimum arrangement of the complement. | 5 |
| Figure 1.6 | Three hypercubes | 8 |
| Figure 2.1 | Example construction of reduction graph for clauses $(a \lor b), (c \lor \overline{d}), (\overline{e} \lor \overline{f})$. | 15 |
| Figure 2.2 | Example construction of reduction digraph for clauses $(a \lor \overline{b}), (c \lor d), (\overline{e} \lor \overline{f})$. | 19 |
| Figure 2.3 | Planar 3SAT to restricted Planar 3SAT [39] | 21 |
| Figure 2.4 | Example construction of reduction digraph for all clauses containing some variable $a$ in Restricted Planar 3SAT. In this example, the clauses in Restricted Planar 3SAT are $(a \lor b \lor c), (a \lor \overline{x}), (\overline{a} \lor y)$. The associated clauses in restricted Planar 2SAT are $(a), (b), (c), (d), (\overline{a} \lor \overline{b}), (\overline{b} \lor \overline{c}), (\overline{c} \lor \overline{a}), (a \lor \overline{d}), (b \lor \overline{d}), (c \lor \overline{d}); (a \lor \overline{x}); (\overline{a} \lor y)$. | 23 |
| Figure 3.1 | Flow of FindSignatures $(G, f)$ when solving for the above orientation of a path of length three. | 36 |
Chapter 1

Introduction

1.1 Why Study Maximum Directed Linear Arrangement: A History

Maximum Directed Linear Arrangement (MaxDLA) is a directed and maximization variant of the well-studied Minimum Linear Arrangement (MinLA) problem. The original motivation for it’s study came from one of my supervisors, Matt DeVos, when he found an interesting arrangement of vertices while investigating the so-called Caccetta-Häggkvist Conjecture, of importance in structural graph theory. The Caccetta-Häggkvist Conjecture is about finding a long directed cycle in a directed graph (digraph). Often, problems on digraphs are more difficult to analyze than their undirected counterparts. MaxDLA has a nice property that it reduces to a problem on undirected graphs with weighted-vertices (cf. Observation 3.2.1). It’s also an ordering of the vertices of a digraph which tends to separate vertices or subsets of vertices with high out-degree from those with high in-degree, which perhaps could be exploited to find a long directed cycle. Anyway, it soon became apparent that this arrangement problem (MaxDLA) is similar to other well-studied layout problems, but its solution doesn’t seem to follow from any of them.

This thesis is therefore an exploration of MaxDLA, which we approach as an interesting problem on its own. We propose to study MaxDLA because of its relation to well-known and well-studied problems. Hence, we briefly look at the history of the following related problems: Minimum Linear Arrangement (MinLA), Maximum Cut (MaxCut), Maximum Directed Cut (MaxDiCut), and the Edge Isoperimetric Problem (EIP). This introduction is meant to provide a quick overview of the area of combinatorics where MaxDLA seems to lie. One can easily become lost in definitions and so, just for this introduction, we use them sparingly. Instead, we try to refer the reader to definitions later in the thesis. A small note for those familiar with graph theory, we will, unless otherwise stated, restrict ourselves to the simple graph versions of these problems, meaning that the graphs we consider have no loops or multiple edges. However, we do at one point consider weighted graphs.
Minimum Linear Arrangement

The Minimum Linear Arrangement (MinLA) problem was introduced in a 1964 paper by Harper [22] who was then a beginning graduate student working at the Jet Propulsion Laboratory (JPL) of the California Institute of Technology. At the time, the JPL was involved in the transmission of the first ever close-up pictures of the surface of the moon, and Harper introduced the idea of MinLA to aid in proving that the transmission code the engineers were using minimized the average absolute error. But MinLA has since found many and varied applications, both within computer science and beyond, for example to wirelength problems [23], and graph crossing number [37]. Another interesting example is to the study of mammalian brains, where, in 1986 Mitchison and Durbin [32] used a generalization of MinLA to study the cerebral cortex of higher mammals.

MinLA asks for a linear arrangement (ordering) of the vertices of an input graph so that the sum of the forward distances travelled by all edges is minimized. Think of an arrangement $\pi = (v_1, v_2, ..., v_n)$ of graph $G$ as a permutation of the vertices of $G$, and the distance travelled by an edge $v_iv_j$ is just $|i - j|$. We refer the reader to Section 1.3 for the precise definition. As an example, two optimal solutions to MinLA for the 4-cycle $C_4$ are found in Figure 1.1, each having value six.

Maximum Cut

A cut of a graph $G$ is a subset $C$ of its edges, so that for some partition $(A, B)$ of the vertices of $G$, $C$ is exactly the set of edges of $G$ with one end in $A$ and the other in $B$. The MaxCut problem asks for a cut of maximum size. See Section 1.3 for the precise definition.

The Maximum Cut (MaxCut) problem was one of the first known NP-complete problems, it was among Karp’s 1972 list of 21 NP-complete problems [27]. Two applications of

---

1. This can be verified by examining all arrangements.

2. The multigraph version of MaxCut is on Karp’s original list. However, the simple graph version was proved NP-complete shortly thereafter [13]
MaxCut include statistical physics, and circuit layout design, both discussed in a 1988 paper of Barahona and others [4]. An important breakthrough in approximating MaxCut was made by Goemans and Williamson in 1995 [15]. They developed a randomized algorithm using semi-definite programming which finds a solution to MaxCut that is at least 0.87856 times the optimal solution in expectation.

**Maximum Directed Cut**

A *directed cut (dicut)* of a digraph $D$ is a subset $C$ of its edges, so that for some partition $(A, B)$ of the vertices $D$, $C$ is exactly the set of edges of $D$ with tail in $A$ and head in $B$. The Maximum Directed Cut (MaxDiCut) problem asks for a dicut of maximum size. See Section 1.3 for the precise definition.

The MaxDiCut problem is a generalization of the MaxCut problem to digraphs. As such, it is of some practical interest, however it is also NP-complete in general. It is nevertheless interesting to look at special classes of digraphs. In particular, Alon et al. published a study of maximum directed cuts in acyclic digraphs in 2007 [1], and Lehel et al. followed this with a study of maximum directed cuts in digraphs with degree restrictions in 2008 [29]. The approximation algorithm of Goemans and Williamson [15] for MaxCut includes an extension to a 0.79607-approximation of MaxDiCut.

**Edge Isoperimetric Problem**

Isoperimetric Problems have been around since antiquity, and the Edge Isoperimetric Problem (EIP) is simply a reframing in terms of graphs. The classical isoperimetric problem, in two dimensions, asks, among all shapes with area $A$, which has the smallest perimeter. The seemingly obvious (and correct) answer is a circle, but it was not always clear. In ancient times, for example, it was believed that the perimeter of a curve determined its area. This wrong belief had material consequences, in particular, some unequal and unfair divisions of land [35, 5].

The EIP asks, given a graph $G = (V, E)$ and non-negative integer $l \leq |V(G)|$, among all subsets of vertices of $G$ with size exactly $l$, which has the smallest edge boundary. Again, we refer the reader to Section 1.3 for the precise definition. Figure 1.2 shows an arrangement of $C_4$ containing nested optimal solutions to the EIP, indicated by the dashed curves$^3$. Indeed, inside each dashed curve is the set $A_l$ consisting of the first $l$ vertices in the arrangement for $l = 1, 2,$ and $3$ (the curve for $l = 4$ is not shown). Then $A_l$ is a solution to the EIP because it has the smallest edge boundary of any sets of size exactly $l$. That is, for any $S \subseteq V(C_4)$ with $|S| = l$, the edge boundary of $S$ is no bigger than the edge boundary of $A_l$.

$^3$This can be verified by examining all subsets $S \subseteq V(C_4)$
Interestingly, the EIP also first appeared in the paper of Harper’s that introduced the MinLA problem [22]. Harper was looking for a solution to MinLA for the hypercube, but actually found a stronger result, which is an arrangement of the hypercube containing nested solutions to EIP [23, pp. 18-19].

1.2 A New Problem: Maximum Directed Linear Arrangement

The Maximum Directed Linear Arrangement (MaxDLA) problem asks, given a digraph $D$ and integer $k$, is there an arrangement of $D$ so that the total distance travelled by the forward edges is at least $k$? For example, Figure 1.3 shows two arrangement of the directed path of length three. The top arrangement has value three because all three edges travel forward one vertex. However, the bottom arrangement is better. It has value four because one edge travels forward three vertices and another travels forward one vertex. Edges traveling backward are ignored. See Section 1.3 for the precise definition.

The MaxDLA problem involves two simultaneous twists on MinLA: directed vs. undirected, and maximizing vs. minimizing. The first twist is a generalization because the class of all digraphs contains the class of all graphs. To generalize to digraphs is interesting because directed problems often offer a richer environment with more complex phenomena.
The second twist is a different perspective; an arrangement is maximum for a digraph $D$ exactly when it is minimum for the complement $\overline{D}$. To maximize is not new. For example, it was used in a 1975 paper by Even and Shiloach \cite[p. 7]{12} in order to prove a result about MinLA. The maximization twist allows one to more easily take advantage of a non-increasing property of the vertices in an optimum arrangement (Property 3.2.4). This property is the level of a vertex, which can be thought of as what that vertex contributes to the next cut. More specifically, it is the difference between the size of the cut immediately following and immediately preceding that vertex. If $D$ is an $n$-vertex digraph, $\pi = (v_1, v_2, \ldots, v_n)$ a maximum arrangement of $D$, and $c_i$ the size of the cut in $\pi$ after vertex $i$. Then $l(v_i) = c_i - c_{i-1}$ is non-increasing as $i$ increases. See Section 3.2 for the formal definition, and further discussion of vertex level. For now, we will illustrate this property with two examples, one of a graph, the other of a digraph.

The first example, Figure 1.4, shows a maximum arrangement of the undirected four-cycle $C_4$, and the same arrangement of its complement, $\overline{C_4}$, which is minimum. The second example, Figure 1.5, shows a maximum arrangement of an orientation of the complete graph on four vertices $K_4$ and the same arrangement of its complement, which is again minimum. Both examples show the levels of the vertices in the maximum arrangement. These are necessarily non-increasing, as we will see in Section 3.2, Property 3.2.4.
A corresponding property, an immediate consequence of Property 3.2.4, which is non-decreasing for minimum arrangements, is as follows. If $D$ is an $n$-vertex digraph, $\pi = (v_1, v_2, \ldots, v_n)$ a minimum arrangement of $D$, and $c_i$ the size of the cut in $\pi$ after vertex $i$. Then $l^{\min}(v_i) = c_i - c_{i-1} + 2i$ is non-decreasing as $i$ increases. The advantage of maximizing instead of minimizing is that one can remove the term $2i$, and calculate levels using only cuts.

1.3 Definitions

Digraphs and Graphs

A directed graph (digraph) $D$ is a pair $(V, E)$ of sets of vertices and edges respectively, where $V$ is a finite set, and $E \subseteq \{(a, b) : (a, b) \in V \times V$ and $a \neq b\}$ is a binary relation on $V$. If $e = (a, b)$ is an edge, then we say $a$ is the tail and $b$ is the head of $e$. We also call $a$ and $b$ the ends of $e$, and for simplicity may drop the brackets writing $e = ab$. In a drawing, vertices are points, and edges are arrows going from tail to head.

Note that, by the above definition, all digraphs considered are simple. This means that $E$ is both irreflexive (an edge $(a, a)$ is a loop and is not allowed), and not a multiset (edges $e = (a, b)$ and $e' = (a, b)$ are multiple or parallel and are not allowed either), but $E$ may have symmetric or opposite edges (an edge $(a, b)$ is opposite $(b, a)$). Infrequently, we allow $E$ to be a multiset, and call such digraphs multi-digraphs. In that case, we say edge $(a, b)$ has multiplicity $k$ when it appears $k$ times in the multiset $E$.

A digraph $D = (V, E)$ is transitive when the relation $E$ is transitive.

Digraph $D$ is symmetric (or undirected, or just a graph) when the relation $E$ is symmetric. We then refer to pairs of opposite edges $(a, b)$ and $(b, a)$ together as one edge $(a, b)$ (or just $e = ab$), and try to distinguish this special case by using $G$ instead of $D$. In drawings, we represent these symmetric edges as segments, not arrows.

An edge $e$ and vertex $a$ are incident if $a$ is an end of $e$. Vertices (edges) are adjacent if they are incident to the same edge (vertex).

Let $D = (V, E)$ be a digraph. The neighbourhood $N_D(a)$ (or just $N(a)$ when the digraph is understood) of vertex $a \in V(D)$ is the set of vertices $v \in V(D)$ to which $a$ is adjacent. The degree of $a$, denoted $d_G(a)$ or $d(a)$, is the number of edges $e \in E(D)$ to which $a$ is incident. Note that in an undirected graph we count a symmetric pair of edges as one edge. Denote the maximum degree of vertex in $D$ by $\Delta(D)$. The out-degree of $a$, denoted $d_H^+(a)$ or $d^+(a)$, is the number of edges $e \in E(D)$ such that $a$ is the tail of $e$. The in-degree of $a$, denoted $d_H^-(a)$ or $d^-(a)$, is the number of edges $e \in E(D)$ such that $a$ is the head of $e$. The digraph $D$ is regular when $d^+(a) = d^-(b)$ for every $a, b \in V(D)$. An undirected graph $G$ is regular when $d(a) = d(b)$ for every $a, b \in V(G)$. We say $G$ is cubic when $G$ is regular of degree three.
An isolated vertex has degree zero. A vertex that is the head (tail) of every edge to which it is incident is a sink (source).

Two digraphs \(D\) and \(D'\) are edge-disjoint whenever \(E(D) \cap E(D') = \emptyset\). They are vertex-disjoint whenever \(V(D) \cap V(D') = \emptyset\).

If \(D = (V, E)\) is a digraph and \(F \subseteq E(D)\), then \(D - F\) denoted \(D - F\) is a new digraph having vertex set \(V(D)\) and edge set \(E(D) - F\). If \(A \subseteq V(D)\), and \(F \subseteq E(D)\) is the set of edges having an end in \(A\), then \(D - A\) denoted \(D - A\) is a new digraph having vertex set \(V(D) - A\) and edge set \(E(D) - F\). For simplicity, we abbreviate \(D - \{x\}\) as \(D - x\).

A digraph \(H\) is a subgraph of digraph \(D\) if \(H\) may be obtained from \(D\) by a series of edge and/or vertex deletions. \(H\) is an induced subgraph if only vertex deletions are allowed.

Given a digraph \(D = (V, E)\), the complement of a subset of vertices \(S \subseteq V(D)\), denoted \(\overline{S}\), is the set of vertices in \(V(D)\) but not in \(S\). The complement of \(E\), denoted \(\overline{E}\), is the set of edges \(\{(a, b) \in V \times V : (a, b) \notin E, a \neq b\}\). The complement of \(D\), denoted \(\overline{D}\), is the digraph with vertex set \(V\) and edge set \(\overline{E}\).

A clique is a set of vertices in a graph so that every distinct pair of vertices in that set are adjacent. A strong clique is a set of vertices \(S\) in a digraph \(D\) so that for every pair of distinct vertices \(u, v \in S\), it is the case that \((u, v) \in E(D)\). An independent set is a set of vertices in a graph (or digraph) whose complement is a clique (or strong clique).

### Classes of Graphs and Digraphs

We now define some undirected and (often corresponding) directed graph classes.

A path on \(n\) vertices \(P_n\) is a graph with vertex set \(V = \{0, 1, ..., n - 1\}\) and edge set so that two vertices \(a, b\) are adjacent exactly when \(|a - b| = 1\). The ends of a path are the vertices of degree one. The internal vertices of a path are the vertices in the path that are not an end. Two paths \(P\) and \(P'\) are internally-disjoint whenever \(E(P) \cap E(P')\) contains no internal vertices.

A directed path on \(n\) vertices is a digraph with vertex set \(V = \{0, 1, ..., n - 1\}\) and edge set so that two vertices \(a, b\) are joined by a directed edge from \(a\) to \(b\) exactly when \(b - a = 1\). We say a directed path is oriented from source vertex to sink vertex.

A cycle on \(n\) vertices \(C_n\) is a graph with vertex set \(V = \{0, 1, ..., n - 1\}\) and edge set so that two vertices \(a, b\) are joined by an edge exactly when \(|a - b| = 1\) (modulo \(n\)).

A directed cycle on \(n\) vertices is a digraph with vertex set \(V = \{0, 1, ..., n - 1\}\) and edge set so that two vertices \(a, b\) are joined by a directed edge from \(a\) to \(b\) exactly when \(b - a = 1\) (modulo \(n\)). The length of a path or cycle is the size of its edge-set.

We say a path or cycle is in a graph \(G\) when \(G\) has \(P\) or \(C\) respectively as a subgraph.

A graph \(G = (V, E)\) is connected when there is a path in \(G\) between every pair of vertices in \(V(G)\). Graph \(G\) is acyclic if \(G\) contains no cycle. A digraph \(D\) is acyclic if \(D\) contains no directed cycle.
A bipartite graph is a graph containing no cycles of odd length. Equivalently, a bipartite graph $G$ has a bipartition $(A, B)$ so that both $A$ and $B$ are independent sets.

A tree $T$ is a connected graph so that for any edge $e \in E(T)$, it is the case that $T - e$ is not connected.

A complete graph on $n$ vertices, often denoted by $K_n$, has $\binom{n}{2}$ edges, one between each pair of vertices in $V(K_n)$.

A hypercube $Q_d$, where $d \geq 1$, is a graph with vertex set $V = \{0, 1\}^d$ and edge set such that two vertices are joined by an edge exactly when they differ in precisely one coordinate. We say $d$ is the dimension of $Q_d$. Figure 1.6 shows three hypercubes.

An orientation of a graph $G$ is a digraph having the same vertex set as $G$, and for every edge $\{a, b\} \in E(G)$, exactly one of $(a, b)$ or $(b, a)$ is in $E(D)$. If $D$ is an orientation of graph $G$, then we say $G$ is the underlying graph of $D$.

A tournament is an orientation of a complete graph.

A split graph is a graph whose vertex-set can be partitioned into an independent set and a clique.

A strict split digraph, as defined in [24] is a digraph whose vertex set can be partitioned into an independent set and a strong clique.

A split digraph, as defined in [28], is a digraph whose vertex set can be partitioned into four sets, $A, B, C, D$ so that $A$ is a strong clique; $D$ is an independent set; all possible edges go from $A$ to $C$ and from $B$ to $A \cup C$; and no edges go from $D$ to $B$ or from $C$ to $B \cup D$. Additionally, it is forbidden to place all vertices in one of $B$ or $C$.

A chordal graph has no induced cycle of length greater than three. All split graphs are chordal.\(^4\)

A planar graph is a graph that can be drawn in the plane so that its edges intersect only at endpoints.

An apex graph has a vertex whose deletion leaves a planar graph. Any such vertex is called an apex of the graph.

\(^4\)This is easy to see, because any induced cycle of length at least four cannot be partitioned into an independent set and a clique.
Cuts

Given a partition of the vertices of a (di)graph $D$ into two sets $A$ and $B$, the (directed) cut from $A$ to $B$, written $E(A, B)$, is the set of edges with tail in $A$ and head in $B$. Note that for an undirected graph this would simply be the edges with one end in $A$ and the other in $B$. We sometimes refer to a directed cut as a dicut. A maximum (di)cut in $D$ is a (di)cut $E(A, B)$ with largest cardinality. We denote the number of edges in such a cut by $\text{MaxDiCut}(D)$ or $\text{MaxCut}(D)$ as appropriate.

The edge boundary of a subset of vertices $A$ is the (directed) cut $E(A, A)$, and we use $\delta(A)$ to denote $|E(A, A)|$.

Arrangements

Let $D$ be an $n$-vertex digraph. A linear arrangement (henceforth shortened to arrangement) $\pi$ of $D$ is a bijection from $V(D)$ to $\{1, 2, \ldots, n\}$. We write $\pi = (v_1, v_2, \ldots, v_n)$ to indicate that $\pi(v_i) = i$ for every $1 \leq i \leq n$. Then we say $v_i$ precedes $v_j$ if $i < j$, we say $v_i$ follows $v_j$ if $i > j$, and we say $v_i$ is between $v_j$ and $v_k$ if either $j < i < k$ or $k < i < j$. An edge $(v_i, v_j)$ is backward in $\pi$ when $v_j$ precedes $v_i$.

A topological sort is an arrangement $\pi$ of $D$ with no backward edges. Note that $D$ has a topological sort if and only if $D$ is acyclic.

Given an arrangement $\pi$, and edge $e = (u, v)$ of $D$, the value of $e$ in $\pi$ denoted val$_{\pi}(e)$ is the maximum of zero and $\pi(v) - \pi(u)$. The value of $\pi$ is just the sum of the values of all edges. Hence,

$$\text{val}_{D}(\pi) = \sum_{e \in E(D)} \text{val}_{\pi}(e).$$

We omit the subscripts $D$ and $\pi$ when the digraph or arrangement is clear from context.

Importantly, we observe that an arrangement $\pi = (v_1, v_2, \ldots, v_n)$, in a natural way, defines certain cuts of a digraph. Indeed, let

$$S_i = \{v_j \in V(D) : j \leq i\}, \quad \text{and}$$
$$T_i = \{v_j \in V(D) : j > i\}.$$

Then we define the cuts $C_1, \ldots, C_{n-1}$ of $\pi$ by the rule that $C_i$ is the directed cut from $S_i$ to $T_i$. These cuts will play an essential role in later discussions. We say that the arrangement $\pi$ contains each $C_i$ and we let

$$c_i = |C_i|.$$
Observe that there is an alternative method of calculating the value of an arrangement \( \pi \), and that is by summing its cut sizes.

\[
val_D(\pi) = \sum_{i=1}^{n-1} c_i. \tag{1.5}
\]

The *level* of a vertex \( v_i \) in \( \pi = (v_1, v_2, \ldots, v_n) \) is defined as

\[
l(v_i) = c_i - c_{i-1}, \tag{1.6}
\]

where, \( c_0 = c_n = 0 \). Intuitively, the level of vertex \( v_i \) can be thought of as its contribution to \( c_i \).

The value of a cut \( c_i \) is therefore the sum of the levels to its left,

\[
c_i = \sum_{j=1}^{i} l(v_j). \tag{1.7}
\]

Thus, there are three equivalent ways to calculate the value of an arrangement, based on edges, cuts, and levels respectively.

**Property 1.3.1.** The value of an arrangement \( \pi = (v_1, v_2, \ldots, v_n) \) of a digraph \( D = (V, E) \) is

\[
val_D(\pi) = \sum_{e \in E(D)} val(e) = \sum_{i=1}^{n-1} c_i = \sum_{i=1}^{n-1} (n - i)l(v_i).
\]

**Complexity**

A *decision problem* \( \Pi \) is a set \( D_\Pi \) and a subset \( Y_\Pi \subseteq D_\Pi \). We will define \( \Pi \) by giving a set of inputs, and a yes/no question over those inputs. In this way, the set of inputs is exactly the set \( D_\Pi \) and the set of all inputs for which the answer to the question is ‘yes’ is \( Y_\Pi \).

As an example we will define a decision problem called Boolean Satisfiability or SAT. But first, some relevant definitions: A *boolean variable* may take on one of two values, say true or false, and we will say that a *literal* is either a boolean variable or its negation. When evaluated, the negation of truth is false, and vice versa. Given a set \( U \) of boolean variables, a *truth assignment* \( \tau \) is an assignment of true or false to each variable in \( U \). We will think of a *clause* as a set of literals, and a clause is *satisfied* under some truth assignment \( \tau \) if at least one of its literals evaluates to true under \( \tau \). A set of clauses \( C \) is satisfied under \( \tau \) if every \( c \in C \) is satisfied under \( \tau \). Now we define SAT.

\[\text{In the Weighted Maximum Linear Arrangement problem (W-MaxLA) defined in Chapter 4, } c_n \text{ is not always zero. However, every other arrangement problem in this thesis has } c_n = 0, \text{ and it is also intuitive. Thus, in the interest of simplicity, we define } c_n = 0 \text{ for now.}\]
Boolean Satisfiability (SAT)

**Input:** A set $U$ of boolean variables, a set $C$ of clauses over $U$.

**Question:** Is $C$ satisfiable?

SAT is an important decision problem because it was shown to be a hardest problem in the class of decision problems in NP, that is, SAT is NP-complete.

Roughly speaking, a decision problem $\Pi$ is in NP if for every $y \in Y_\Pi$ there exists a certificate $c_y$ so that, given $y$ and $c_y$, it can be verified in polynomial-time that $y$ is in fact a ‘yes’ instance of $\Pi$. A *polynomial-time reduction* of a decision problem $\Sigma$ to a decision problem $\Pi$ is a polynomial-time computable mapping $f : D_\Sigma \mapsto D_\Pi$ so that $f(p) \in Y_\Pi$ if and only if $p \in Y_\Sigma$. A problem $\Pi \in$ NP is said to be NP-complete if it admits a polynomial-time reduction from SAT. It follows that, to demonstrate $\Pi$ is NP-complete, it is enough to show that $\Pi$ admits a polynomial-time reduction from any NP-complete problem. To demonstrate NP-completeness, the general method used in this thesis is to take an arbitrary instance $p$ of a known NP-complete problem $\Sigma$, and construct an instance $p'$ of $\Pi$ so that $p' \in Y_\Pi$ if and only if $p \in Y_\Sigma$.

We refer the reader to the monograph by Garey and Johnson [14] for a thorough treatment of NP-completeness theory, and a broad list of NP-complete problems.

**Decision Problems**

Here, notable decision problems used in this thesis are defined. The corresponding optimization problems, if applicable, are found in the usual way. That is, by removing $k$ from the input and asking the question what is the maximum (minimum) value of the quantity which would have been at least (most) $k$.

When a decision problem has an optimization version, we denote the maximum (or minimum as appropriate) value of $k$ for which the answer to the decision problem is ‘yes’ by [short name of problem](input 1, input2, ...). For example, $\text{MaxDLA}(D)$ is the maximum value of an arrangement of digraph $D$, that is $\text{MaxDLA}(D) = \max\{\text{val}(\pi) : \pi \text{ is an arrangement of } D\}$.

**Maximum Directed Linear Arrangement (MaxDLA)**

**Input:** A digraph $D$, integer $k$.

**Question:** Is there an arrangement of $D$ with value at least $k$?

**Minimum Directed Linear Arrangement (MinDLA)**

**Input:** A digraph $D$, integer $k$.

**Question:** Is there an arrangement of $D$ with value at most $k$?

**Maximum Linear Arrangement (MaxLA)**

**Input:** A graph $G$, integer $k$.

**Question:** Is there an arrangement of $G$ with value at least $k$?
Minimum Linear Arrangement (MinLA)

**Input:** A graph $G$, integer $k$.

**Question:** Is there an arrangement of $G$ with value at most $k$?

Weighted Maximum Linear Arrangement (W-MaxLA)

**Input:** A graph $G$, a weight function $f : V(G) \mapsto \mathbb{Z}$, integer $k$.

**Question:** Is there an arrangement of $G$ with value at least $k$?

Edge Isoperimetric Problem (EIP)

**Input:** A graph $G = (V, E)$, integers $l$ and $k$.

**Question:** Is there a $S \subseteq V$ with $|S| = l$ whose edge boundary is at most $k$?

Boolean Satisfiability (SAT)

**Input:** A set $U$ of boolean variables, a set $C$ of clauses over $U$.

**Question:** Is $C$ satisfiable?

Planar 3-Satisfiability (Planar 3SAT)

**Input:** A set $U$ of boolean variables, a set $C$ of clauses over $U$, each of size exactly three and so that the variable graph$^6$ of $C$ is planar, integer $k$.

**Question:** Is $C$ satisfiable?

Maximum 2-Satisfiability (Max2SAT)

**Input:** A set $U$ of boolean variables, a set $C$ of clauses over $U$, each of size exactly two, integer $k$.

**Question:** Is there a subset $C'$ of $C$ with $|C'| \geq k$ that is satisfiable?

Planar Maximum 2-Satisfiability (Planar Max2SAT)

**Input:** A set $U$ of boolean variables, a set $C$ of clauses over $U$, each of size exactly two and so that the variable graph$^7$ of $C$ is planar, integer $k$.

**Question:** Is there a subset $C'$ of $C$ with $|C'| \geq k$ that is satisfiable?

Maximum Cut (MaxCut)

**Input:** A graph $G = (V, E)$, integer $k$.

**Question:** Is there a cut of $G$ with size at least $k$?

---

$^6$The *variable graph* of a set of clauses $C$ over variables $U$ is a graph with vertex bipartition $(U, C)$ so that $u \in U$ is adjacent to $c \in C$ exactly when $u$ or $\overline{u}$ is in $c$.

$^7$In the special case of 2SAT, clause vertices are suppressed in the variable graph. We say the variable graph of a set of clauses $C$ over variables $U$ has vertex set $U$, and $u, u' \in U$ are adjacent exactly when either of $u$ or $\overline{u}$ and either of $u'$ or $\overline{u'}$ occur together in some clause of $C$. 

12
Maximum Directed Cut (MaxDiCut)

**Input:** A digraph $D = (V, E)$, integer $k$.

**Question:** Is there a dicut of $D$ with size at least $k$?

Planar Maximum Cut (Planar MaxCut)

**Input:** A planar graph $G = (V, E)$, integer $k$.

**Question:** Is there a cut of $G$ with size at least $k$?

Planar Maximum Directed Cut (Planar MaxDiCut)

**Input:** A planar digraph $D = (V, E)$, integer $k$.

**Question:** Is there a dicut of $D$ with size at least $k$?

### 1.4 Organization of Following Chapters

The remainder of this thesis is organized as follows.

In Chapter 2, the complexity of the Maximum Cut, Minimum Linear Arrangement, and Maximum Directed Cut problems is discussed. In Section 2.1, we give a new proof that the MaxCut problem is NP-complete. In Section 2.2, we prove that the Minimum Linear Arrangement problem is NP-complete in a manner inspired by Even and Shiloach [12], which uses the MaxLA problem as an intermediary. In Section 2.3, we prove a new result, that the Maximum Directed Cut problem is NP-complete for planar digraphs. This is an interesting contrast to the undirected version of the problem, which, due to Orlova and Dorfman [33], and Hadlock [20], is polynomial-time solvable on planar graphs.

In Chapter 3, we introduce a new problem, Maximum Directed Linear Arrangement. In Section 3.1, we prove some NP-completeness results for MaxDLA and demonstrate some bounds on $MaxDLA(D)$. In Section 3.2, we introduce two notions essential for the algorithm in Section 3.3, which solves MaxDLA on $n$-vertex trees with degrees bounded by a constant $d$ in time $O(n^{4d})$. In Section 3.4, we modify the algorithm of the previous section to solve MinLA on graphs $G$ when $\overline{G}$ is a bounded-degree tree. This complements results of Goldberg and Klipker [16], Shiloach [38], and Chung [9], that MinLA is polynomial time solvable on trees.

In Chapter 4, we define and discuss nested-minimum and nested-maximum digraphs. In Section 4.1, we reprove a celebrated result of Harper on hypercubes, that they have an arrangement $\pi$ so that every cut of $\pi$ is a solution to the EIP; in other words, hypercubes are nested-minimum. Finally, in Section 4.2, we demonstrate three classes of digraphs with the nested-maximum property: tournaments, orientations of graphs with maximum degree at most two, and transitive acyclic digraphs.
Chapter 2

Complexity of Related Problems

2.1 Maximum Cut

Maximum Cut for multigraphs was shown to be NP-complete by Karp [27] in 1972, in a seminal paper using a reduction from Partition. The same result for (simple) graphs was shown by Garey, Johnson and Stockmeyer [13, pp. 240-242] in 1976, this time reducing from Max2SAT. We reprove the latter result here, again reducing from Max2SAT, but differently. Max2SAT was shown to be NP-complete by reduction from 3SAT, also in [13, p. 240].

Theorem 2.1.1. [Garey, Johnson and Stockmeyer [13]] Maximum Cut is NP-complete.

Proof. Maximum Cut is in NP because, given a subset \( A \) of the vertices of the input graph \( G \) as a certificate that \( G \) has a cut of size at least \( k \), it can be verified in polynomial time that \( A \) is indeed a subset of the vertices of \( G \) and that the number of edges with one end in \( A \) and the other end in \( \overline{A} \) is at least \( k \).

We show that Max2SAT is reducible to Maximum Cut (Max Cut). Consider an instance of Max2SAT with boolean variables \( U \), clauses \( C \) where \( |C| = l \), and integer \( k \). We construct graph \( G = (V, E) \) as follows:

1. Let \( V(G) = U \cup \{T\} \).

2. For each \( c \in C \), add a gadget consisting of three paths, so that the two variables in \( c \) and \( T \) are joined in a cycle. Do this so that every gadget includes exactly nine new edges, and therefore six new ‘gadget’ nodes. Also ensure that path lengths obey the following two parity rules.

   (a) The path connecting both variables has even length if and only if exactly one variable is positive in \( c \).

   (b) The path connecting a variable to \( T \) has odd length if and only if the variable is negative in \( c \).

Figure 2.1 shows an example construction of \( G \) for a small instance of Max2SAT.
The node $T$ will determine the ‘truth’ side of any cut, so that we may imagine that a cut of $G$ assigns ‘true’ to a variable in $U$ if and only if that variable is on the same side of the cut at $T$.

**Claim.** Recall that $l = |C|$. There exists a truth assignment to the variables in $U$ so that at least $k$ clauses in $C$ are satisfied if and only if $G$ has a cut of size at least $6l + 2k$.

Suppose there is a truth assignment $\tau$ to the variables of $U$ so that at least $k$ clauses in $C$ are satisfied. We construct a cut $D = E(\mathcal{T}, \mathcal{F})$ in $G$ as follows. First, set

$$\mathcal{T} = \{u \in U : \tau(u) = true\} \cup T,$$

$$\mathcal{F} = \{u \in U : \tau(u) = false\},$$

and finally, assign the ‘gadget’ nodes so that $D$ is maximized. The last part is simple to do because they are all degree two.

Now, because the gadgets are edge-disjoint, the contribution that each makes to $D$ can be evaluated separately. It can be quickly verified that each gadget contributes exactly eight edges to $D$ when its clause is satisfied, and exactly six edges otherwise. Therefore, $D$ has size at least $8k + 6(l - k) = 6l + 2k$.

Conversely, suppose there is a cut $D = E(\mathcal{T}, \mathcal{F})$ in $G$ with size at least $6l + 2k$. Without loss of generality, let $T \in \mathcal{T}$. We find a truth assignment $\tau$ to the variables in $U$ as follows. For all $u \in U$, set

$$\tau(u) = true$$

if $u \in \mathcal{T}$, and

$$\tau(u) = false$$

if $u \in \mathcal{F}$.

Since $D$ has size at least $6l + 2k$, it must be that at least $k$ gadgets contribute eight edges to $D$. But it can be quickly verified that when a gadget contributes eight edges to $D$, its
clause is satisfied under $\tau$. Therefore $\tau$ is a truth assignment to the variables in $U$ so that at least $k$ clauses in $C$ are satisfied.

This reduction also shows that MaxCut for apex graphs is NP-complete, which follows because Planar Max2SAT is NP-complete. A superior result appeared implicitly in a paper by Barahona [3, pp. 3251 - 3252] in 1982, that MaxCut for apex graphs which are cubic, except for the apex vertex, is NP-complete. The reduction used by Barahona is basically from Maximum Independent Set in a cubic planar graph, which was shown NP-complete in Garey, Johnson and Stockmeyer’s paper mentioned previously [13, pp. 262-265].

## 2.2 Minimum Linear Arrangement

Minimum Linear Arrangement (MinLA) is NP-complete for general graphs. This was shown by Garey Johnson and Stockmeyer in 1976 [13, pp. 244-246], and Even and Shiloach in 1975 [12, pp. 8-11]. We reprove their result here in two steps, as was done by Even and Shiloach: first showing that MaxCut is polynomial-time reducible to MaxLA and then that MaxLA is polynomial-time reducible to MinLA. This reduction illustrates the potential interest in the maximization version of the problem.

**Lemma 2.2.1.** MaxCut is polynomial-time reducible to MaxLA.

*Proof.* Consider an instance of MaxCut with graph $G = (V, E)$ on $n$ vertices, and positive integer $k$. We may assume $G$ has at least one edge, and that $k \leq \binom{n}{2}$ because otherwise the solution to MaxCut is trivial.

Construct a new graph $G'$ from $G$ by adding a set $S$ of $n^3$ isolated vertices to $G$. That is, $G' = (V(G) \cup S, E(G))$.

**Claim.** Graph $G$ has a cut of size at least $k$ if and only if $G'$ has a linear arrangement of size at least $kn^3$.

Suppose $G$ has a cut $C = E(A, B)$ of size at least $k$. Then, arrange the vertices of $G'$ so that all vertices in $A$ come before all vertices in $S$, which in turn come before all vertices in $B$. This arrangement has value at least $kn^3$ because the large cut $C$ is repeated for every vertex in $S$.

Now, suppose $G$ has no cut of size at least $k$. Then $G'$ also has no cut of size at least $k$. Hence an arrangement $\pi$ of $G'$ can have value at most $(k-1)(n^3 + n - 1)$ because every one of the $(n^3 + n - 1)$ cuts in $\pi$ has size at most $(k-1)$. But $k \leq \binom{n}{2} < n^2$ by assumption, so that

---

1As stated above, in 1976 Garey, Johnson and Stockmeyer showed that Max2SAT is NP-complete [13, pp. 240]. Conveniently, the reduction they used preserves planarity. The next year, Lichtenstein showed that Planar 3SAT is NP-complete [30, pp. 330-333], and so by Garey, Johnson and Stockmeyer’s reduction the NP-completeness of Planar Max2SAT immediately followed. This was observed by Guibas et al. in 1991 [19, p. 154].
\[ n^3 > (k-1)n. \] But this means \( \pi \) has value at most \( (k-1)(n^3+n-1) \leq kn^3 + (k-1)n - n^3 < kn^3. \]

**Lemma 2.2.2.** MaxLA is polynomial-time reducible to MinLA.

**Proof.** Consider an instance of MaxLA with graph \( G \) on \( n > 1 \) vertices, and positive integer \( k \).

**Claim.** Graph \( G \) has an arrangement of value at least \( k \) if and only if its complement \( \overline{G} \) has an arrangement of value at most \( \left( \frac{n+1}{3} \right) - k \).

To prove the claim we observe that the value of an arrangement of a complete graph on \( n \) vertices \( K_n \) is \( \sum_{i=1}^{n} i(n-i) \). Hence,

\[ \text{MaxDLA}(K_n) = \left( \frac{n+1}{3} \right), \quad (2.1) \]

(cf. [13, p244]) as can be proved, for example, by induction. This means that, for any arrangement \( \pi \) of \( G \),

\[ \text{val}_{G}(\pi) + \text{val}_{\overline{G}}(\pi) = \left( \frac{n+1}{3} \right). \quad (2.2) \]

To conclude proof of the lemma, suppose \( G \) has an arrangement \( \pi \) of value at least \( k \). Then

\[ \text{val}_{\overline{G}}(\pi) = \left( \frac{n+1}{3} \right) - \text{val}_{G}(\pi) \leq \left( \frac{n+1}{3} \right) - k. \quad (2.3) \]

Suppose \( \overline{G} \) has an arrangement \( \sigma \) of value at most \( \left( \frac{n+1}{3} \right) - k \). Then

\[ \text{val}_{G}(\sigma) = \left( \frac{n+1}{3} \right) - \text{val}_{\overline{G}}(\sigma) \geq \left( \frac{n+1}{3} \right) - \left[ \left( \frac{n+1}{3} \right) - k \right] = k. \quad (2.4) \]

Combining the above two lemmas yields the following theorem.

**Theorem 2.2.3** ([13, 12]). Minimum Linear Arrangement is NP-complete.

**Proof.** Minimum Linear Arrangement is in NP because, given an arrangement \( \pi = (v_1, v_2, ..., v_n) \) of the vertices of the input graph \( G \), it can be verified in polynomial time that \( \pi \) is indeed an arrangement of \( V(G) \), and that the value of \( \pi \) is at least integer \( k \).

But now, by application of Lemmas 2.2.1 and 2.2.2 respectively, it follows that MaxCut is polynomial-time reducible to MinLA.

We now show that a consequence of this reduction is that MinLA is NP-complete for split graphs. To the author’s knowledge this has not yet been documented [34].
The graphs constructed in the reductions of Lemmas 2.2.1 and 2.2.2 were obtained by adding \( n^3 \) isolated vertices, and by taking the complement respectively. We therefore make the following observation.

**Theorem 2.2.4.** Suppose \( \mathcal{C} \) is a class of graphs closed under the operation of adding an isolated vertex. If MaxCut is NP-complete on \( \mathcal{C} \) then so is MaxLA.

Observe that Theorem 2.2.4 has implications to the MinLA problem, because when MaxLA is NP-complete on a class of graphs \( \mathcal{C} \), then MinLA is NP-complete on \( \overline{\mathcal{C}} \). In particular, because MaxCut is NP-complete for split graphs [6], and because split graphs are closed under both complementation and adding an isolated vertex, it follows that split graphs are NP-complete for both MaxLA and MinLA.

**Theorem 2.2.5.** MinLA is NP-complete for split graphs.

### 2.3 Maximum Directed Cut

Maximum Directed Cut (MaxDiCut) is NP-complete in general. This follows from the undirected case, because the MaxDiCut problem contains the MaxCut problem. In this section we will see that MaxDiCut is NP-complete for planar digraphs. To my knowledge this result is new, and it contrasts that MaxCut is polynomial-time solvable for planar graphs [33, 20]. We also show that MaxDiCut remains NP-complete for planar graphs with maximum degree 14. At the end of this section, we observe that MaxDiCut is polynomial-time solvable when no vertex has degree greater than 2. (A stronger result, that those digraphs have a special nested-maximum arrangement, is discussed in Section 4.)

We will reduce Planar Max2SAT to Planar MaxDiCut in a way similar to the proof of Theorem 2.1.1.

**Theorem 2.3.1.** Planar MaxDiCut is NP-complete.

**Proof.** First, we observe that Planar MaxDiCut is in NP because, given a subset \( A \) of the vertices of the input digraph \( D \) as a certificate that \( D \) has a dicut of size at least \( k \), it can be verified in polynomial time that \( A \) is indeed a subset of the vertices of \( D \), and that the number of edges with tail in \( A \) and head in \( \overline{A} \) is at least \( k \).

We proceed by reduction from Planar Max2SAT. Consider an instance of Planar Max2SAT with boolean variables \( U \), clauses \( C \) where \( |C| = l \), and integer \( k \). We construct a planar digraph \( D = (V, E) \) as follows:

1. Let \( V(D) = U \).
2. For each \( c \in C \), add a gadget consisting of four edges and one or two ‘gadget’ nodes as follows:
Figure 2.2: Example construction of reduction digraph for clauses \((a \lor b), (c \lor d), (\bar{c} \lor \bar{d})\).  

(a) If exactly one variable is positive in \(c\), then connect both variables in \(c\) by two internally disjoint paths of length two, oriented from the positive variable to the negative variable.

(b) Otherwise, add a pair of symmetric edges between the variables of \(c\). Add one additional ‘gadget’ vertex \(v\) dominating the variables in \(c\) so that \(v\) is a sink when both variables of \(c\) are positive, and a source when both variables are negative.

Note that \(D\) is planar because the instance of Max2SAT is planar. Figure 2.2 shows an example construction of \(D\) for a small instance of Planar Max2SAT.

**Claim.** There is a truth assignment to the variables of \(U\) so that at least \(k\) clauses in \(C\) are satisfied if and only if \(D\) has a dicut of size at least \(2k\).

Suppose there is a truth assignment \(\tau\) to the variables of \(U\) so that at least \(k\) clauses in \(C\) are satisfied. We construct a dicut \(E = E(\mathcal{T}, \mathcal{F})\) in \(D\) as follows. First, set

\[
\mathcal{T} = \{u \in U : \tau(u) = true\}, \text{ and} \\
\mathcal{F} = \{u \in U : \tau(u) = false\},
\]

and finally place the ‘gadget’ vertices so the number of edges across \(E\) is maximized. The last part is simple to do because they are all degree two.

Now, because the gadgets are edge-disjoint, the contribution each makes to \(E\) can be evaluated independently. It can be quickly verified that gadgets whose clause is true under \(\tau\) contribute exactly two edges to \(E\). Hence \(E\) has size at least \(2k\).

Conversely, suppose \(D\) has a dicut \(E = E(\mathcal{T}, \mathcal{F})\) of size at least \(2k\). We may assume the gadget vertices are arranged so as to maximize \(E\). We find a truth assignment \(\tau\) to the variables of \(U\) as follows. For all \(u \in U\), set

\[
\tau(u) = true \text{ if } u \in \mathcal{T}, \text{ and} \\
\tau(u) = false \text{ if } u \in \mathcal{F}.
\]

Observe that each gadget is constructed so that it contributes either two or zero edges to any cut. The reader may quickly verify that (1) if a gadget contributes exactly two edges
to $E$, then its clause is satisfied under $\tau$. and (2) if a gadget contributes exactly zero edges to $E$, then its clause is not satisfied under $\tau$.

Since $E$ has size at least $2k$, and all edges of $D$ are in gadgets, it must be that at least $k$ gadgets contribute two edges to $E$, and so at least $k$ clauses of $C$ are satisfied under $\tau$. □

We now strengthen Theorem 2.3.1 to include the restriction that no vertex has degree exceeding fourteen. We reduce from Planar 3SAT via Planar Max2SAT to Planar MaxDi-Cut, following existing reductions. However, we will be careful about limiting the number of clauses in which each variable occurs, and what those clauses look like. For this reason, we define a restricted version of Planar 3SAT.

**Restricted Planar 3-Satisfiability (Restricted Planar 3SAT)**

**Input:** A set $U$ of boolean variables, a set $C$ of clauses over $U$ so that the following conditions are satisfied:

1. The variable graph of $C$ is planar.
2. Each variable in $U$ occurs in exactly three clauses of $C$ so that
   (a) two of the clauses contain exactly two literals each, with exactly one negated variable per clause.
   (b) the third clause contains exactly three literals.

**Question:** Is $C$ satisfiable?

Note that Restricted Planar 3SAT is in NP. We now prove that it is NP-complete following a reduction of [31, pp. 96-97] as done in [39].

**Lemma 2.3.2.** Planar 3SAT is polynomial-time reducible to Restricted Planar 3SAT.

**Proof.** Consider an instance of Planar 3SAT $\phi$ with variables $U$, clauses $C$, and integer $k$. Find a planar embedding of the variable graph in polynomial time [2].

We construct an instance of Restricted Planar 3SAT $\psi$ from $\phi$ as follows. For every $u \in U$, let $l$ be the number of occurrences of $u$ in the clauses of $C$. Replace $u$ in $U$ with $l$ new variables $u_1, ..., u_l$ and, to $C$, add $l$ new clauses $\overline{u_i} \lor u_{i+1}$ (mod $l$) for $i = 1, ..., l$. Replace each occurrence of $u$ in an original clause of $C$ with one of the distinct new variables, choosing carefully so as to preserve planarity of the variable graph, as per figure 2.3. Then $\psi$ is an instance of Restricted Planar 3SAT.

**Claim.** The original instance of Planar 3SAT $\phi$ is satisfiable if and only if the restricted version $\psi$ is satisfiable.

Suppose $\phi$ is satisfiable. Then the same truth assignment $\tau$ which satisfies $\phi$ can be used to find a truth assignment which satisfies $\psi$. Just assign each new variable $u_i$ of $\psi$ a truth value equal to $\tau(u)$. 

20
Conversely, suppose $\psi$ is satisfiable. Let $\tau$ be a truth assignment which satisfies the clauses of $\psi$. Then it must be the case that if new variables $u_i$ and $u_j$ of $\psi$ stem from the same original variable $u$ of $\phi$, then $\tau(u_i) = \tau(u_j)$. This is because of the new clauses added to $\psi$ for variable $u$, which act like a circle of implication. So assign to each original variable $u$ of $\phi$ a truth value equal to $\tau(u_1)$. This is a truth assignment which satisfies the clauses of $\phi$. \hfill \square

Now we define a restricted version of Planar Max2SAT.

**Restricted Planar Maximum 2-Satisfiability (Restricted Planar Max2SAT)**

**Input:** A set $U$ of boolean variables, a set $C$ of clauses over $U$, and integer $k$ so that the following conditions are satisfied:

1. The variable graph of $C$ is planar.
2. Each $c \in C$ has $|c| \leq 2$.
3. Each variable in $U$ occurs in at most six clauses of $C$, at most two of which contain either both positive or both negative variables.

**Question:** Is there a subset $C'$ of $C$ with $|C'| \geq k$ that is satisfiable?

Note that Restricted Planar Max2SAT is in NP. We now prove that it is NP-complete following a reduction of Garey, Johnson, and Stockemeyer [13, p. 240].

**Lemma 2.3.3.** Restricted Planar 3SAT is polynomial-time reducible to Restricted Planar Max2SAT.

**Proof.** Consider an instance of Restricted Planar 3SAT with variables $U$ and clauses $C$, where $C$ contains $l$ clauses of size 3, and $m = 3l$ clauses of size 2. We construct an instance of Restricted Planar Max2SAT with variables $U'$, clauses $C'$, and integer $k$ as follows. First,
set

\[ U' = U \cup \{d_1, d_2, \ldots, d_l\}, \text{ and} \]
\[ C' = \{c \in C : c \text{ has size two}\}. \]

Then add ten clauses to \( C' \) for every clause \( c_i = (a_i, b_i, c_i) \in C \), where \( i = 1, \ldots, l \) of size three in the following way. If \( c_i \) contains all negated variables, add the following ten clauses to \( C' \):

\[(a_i), (b_i), (c_i), (\overline{a_i} \lor \overline{b_i}), (\overline{b_i} \lor \overline{c_i}), (\overline{c_i} \lor \overline{a_i}), (a_i \lor d_i), (b_i \lor d_i), (c_i \lor d_i). \]

Otherwise, add the following ten clauses to \( C' \):

\[(a_i), (b_i), (c_i), (d_i), (\overline{a_i} \lor \overline{b_i}), (\overline{b_i} \lor \overline{c_i}), (\overline{c_i} \lor \overline{a_i}), (a_i \lor \overline{d_i}), (b_i \lor \overline{d_i}), (c_i \lor \overline{d_i}). \]

Finally, let \( k = 7l + m \). The reader can verify that this is an instance of Restricted Planar Max2SAT.

**Claim.** Recall that \( C \) contains \( l \) clauses of size 3, and \( m = 3l \) clauses of size 2. The clauses \( C \) are satisfiable if and only if \( 7l + m \) of the clauses in \( C' \) are satisfiable.

We begin with the crucial observation of Garey, Johnson and Stockmeyer [13], which the reader may verify. Let \( \tau \) be a truth assignment to \( U \). Then \( \tau \) satisfies a clause \( c \in C \) if and only if \( \tau \) may be extended to \( U' \) so that seven of the ten clauses in \( C' \) stemming from \( c \) are satisfiable. Further, if \( \tau \) does not satisfy \( c \), then any extension of \( \tau \) to \( U' \) can satisfy at most six of the ten clauses in \( C' \) stemming from \( c \).

Suppose \( C \) is satisfiable. Let \( \tau \) be a truth assignment to \( U \) that satisfies \( C \). By the observation above, \( \tau \) may be extended to \( U' \) so that seven of every ten clauses in \( C' \) associated with a three-variable clause in \( C \) are satisfied. Since this extension necessarily satisfies all \( m \) 2-literal clauses in \( C' \) that are also in \( C \), the extension of \( \tau \) satisfies \( 7l + m \) of the clauses in \( C' \).

Conversely, suppose \( 7l + m \) of the clauses in \( C' \) are satisfiable. Let \( \tau' \) be a truth assignment to \( U' \) that satisfies at least \( 7l + m \) of the clauses in \( C' \), and let \( \tau \) be \( \tau' \) restricted to the variables in \( U \). Observe that at most seven of the ten clauses in \( C' \) stemming from a single 3-variable clause in \( C \) can be simultaneously satisfied. Since there are only \( l \) such collections of ten clauses, it must be the case that \( \tau' \) satisfies seven of ten clauses for all such collections of ten clauses, and also all \( m \) two-variable clauses in \( C' \). But this means, by the observation above, that \( \tau \) is a truth assignment to \( U \) that satisfies \( C \).

Recall that \( \Delta(D) \) denotes the maximum number of edges incident to a single vertex in digraph \( D \).
Theorem 2.3.4. Planar MaxDiCut is NP-complete on planar graphs whose maximum degree is at most 14.

Proof. As previously observed, Planar MaxDiCut is in NP, therefore so is the restricted version. We proceed by reduction from Restricted Planar Max2SAT, in the same manner as Theorem 2.3.1. We will not repeat the details here, but observe how the restricted version of Planar Max2SAT gives rise to the degree restrictions.

Consider an instance of Restricted Planar Max2SAT with boolean variables $U$, clauses $C$, and integer $k$. We construct a planar digraph $D = (V, E)$ as in Theorem 2.3.1.

Recall that, $V = U$, and the degree of any $v \in V$ depends only on the clauses $U$ that contain $v$. Since $v$ may be in at most six clauses, and all of those clauses but perhaps two (those with two variables of the same sign), contribute two to the degree of $v$, and the remaining contribute three, it is clear that no vertex may have degree exceeding 14. 

\[\square\]

Figure 2.4: Example construction of reduction digraph for all clauses containing some variable $a$ in Restricted Planar 3SAT. In this example, the clauses in Restricted Planar 3SAT are $(a \lor b \lor c), (a \lor \overline{a}), (\overline{a} \lor y)$. The associated clauses in restricted Planar 2SAT are $(a), (b), (c), (d), (\overline{a} \lor \overline{b}), (b \lor \overline{a}), (\overline{a} \lor d), (b \lor d), (c \lor d)$; $(a \lor \overline{x}); (b \lor \overline{d}); (\overline{a} \lor y)$.

Figure 2.4 depicts the digraph arising from the above reductions applied to a small instance of Restricted Planar 3SAT.

Theorem 2.3.4 prompts the question, what is the lowest maximum degree for which Planar MaxDiCut remains NP-complete? The answer is somewhere between three and fourteen by the following observation.

Observation 2.3.5. MaxDiCut is polynomial-time solvable on graphs whose maximum degree is two.

Proof. Let $D$ be a digraph whose maximum degree is two. Then every $v \in V(D)$ is either a source, a sink, or has exactly one in-neighbour and one out-neighbour. We describe how to construct a cut $E(A, B)$ of $D$ which is maximum. First, place every source in $A$ and
every sink in $B$. We are left with, possibly, cycles or directed paths from $A$ to $B$. Due to the symmetry of a cycle, we may place one of its vertices in $A$. Every vertex so far has been placed without any sacrifice to the cut. That is, there exists a maximum cut of $D$ with vertices placed as described. It remains only to place the vertices of directed paths, all of which start in $A$, and end in either $A$ or $B$. But now it is easy to see how to place the vertices of a directed path $P$. Begin with the vertex in $P$ adjacent to the end of $P$ in $A$, and place vertices along $P$ alternately in $B$ then $A$.  

\end{proof}
Chapter 3

Maximum Directed Linear Arrangement

3.1 Complexity and Bounds

In this section, we state some NP-completeness results for the MaxDLA problem, and demonstrate certain bounds on \( \text{MaxDLA}(D) \).

We begin by recalling the Maximum Directed Linear Arrangement problem.

**Maximum Directed Linear Arrangement (MaxDLA)**

**Input:** A digraph \( D \), integer \( k \).

**Question:** Is there an arrangement of \( D \) with value at least \( k \)?

First, we extend Theorem 2.2.4 (on classes of graphs closed under the addition of isolated vertices) to digraphs, and then use the extension to show some NP-completeness results.

**Theorem 3.1.1.** Suppose \( C \) is a class of digraphs closed under the operation of adding an isolated vertex. If \( \text{MaxDiCut} \) is NP-complete on \( C \), then so is \( \text{MaxDLA} \).

The proof of Theorem 3.1.1 is similar to the proof of Theorem 2.2.4 and we skip it.

The following theorem summarizes some NP-completeness results for MaxDLA, illustrating the usefulness of Theorem 3.1.1.

**Theorem 3.1.2.** \( \text{MaxDLA} \) is NP-complete for

1. general digraphs,

2. planar digraphs with maximum degree at most 14,

3. strict split digraphs, and

4. split digraphs.
Proof. MaxDLA is in NP because, given an arrangement \( \pi \) of the vertices of the input graph \( D \) as a certificate that \( D \) has an arrangement of size at least \( k \), it can be verified in polynomial time that \( \pi \) is indeed an arrangement of the vertices of \( D \), and that the value of \( \pi \) is at least \( k \).

Proof of 1. MaxDLA is NP-complete for general digraphs because, by Theorem 2.2.1, the undirected version, MaxLA, is NP-complete, and because MaxDLA contains MaxLA, to which it reduces when each edge is symmetric.

Proof of 2. MaxDLA is NP-complete for planar digraphs with degree at most 14 because, by Theorem 2.3.4, MaxDiCut is NP-complete for those graphs, and by Theorem 3.1.1.

Proof of 3 and 4. MaxDLA is NP-complete for strict split digraphs and split digraphs by the following reasoning. First, MaxCut for split graphs is NP-complete by a theorem of Bodlaender [6]. Since the class of split graphs is contained in the intersection of strict split digraphs and split digraphs,\(^1\) it follows that MaxDiCut is NP-complete for those digraphs. Finally, by Theorem 3.1.1, MaxDLA is NP-complete for both strict split digraphs and split digraphs. \(\square\)

Next, we demonstrate some bounds on MaxDLA(D) for a digraph \( D \). We bound MaxDLA(D) in terms of (1) the number of vertices and edges in \( D \), (2) MaxDiCut(D), (3) MinDLA(D), and (4) the degrees of the vertices in \( D \).

First, we bound the size of a maximum linear arrangement of a digraph in terms of the size of its vertex and edge sets.

**Property 3.1.3.** Let \( D = (V, E) \) be a loopless multi-digraph with \( |V| = n \) and \( |E| = m \). Then
\[
\text{MaxDLA}(D) \geq \frac{1}{6}m(n+1).
\]
This bound is best possible, and equality holds for complete symmetric digraphs.

Proof. Denote by \( \Pi \) the set of all \( n! \) possible arrangements of \( D \), and consider the sum of their values
\[
\text{val}(\Pi) = \sum_{\pi \in \Pi} \text{val}(\pi).
\]
Let \( i, j \) be integers, and \( e = (u, v) \in E(D) \). For every \( 1 \leq i < j \leq n \), there are \((n-2)!\) arrangements of \( D \) which map both \( u \) to \( i \) and \( v \) to \( j \). Therefore, each edge of \( D \) contributes exactly \((n-2)!\text{val}(K_n)\) to \( \text{val}(\Pi) \). It follows that
\[
\text{val}(\Pi) = m(n-2)!\text{val}(K_n)
= m(n-2)!\binom{n+1}{3}.
\]

\(^1\)Every split graph, when each undirected edge is viewed as two symmetric directed edges, is a both a split digraph and a strict split digraph.
Thus the mean value of all \( n! \) arrangements of \( D \) is \( \frac{1}{n!} \cdot m(n - 2)! \left(\frac{n+1}{3}\right) = \frac{1}{6} m(n + 1) \). Since there must be an arrangement of \( D \) whose value achieves the mean, this completes the proof of the inequality.

We show that the bound is best possible by demonstrating that equality holds for complete symmetric digraphs. Indeed, the value of an arrangement of an \( n \)-vertex complete symmetric digraph is equal to \( \text{MaxLA}(K_n) \), which is \( \left(\frac{n+1}{3}\right) \) by Equation 2.1. This is exactly the lower bound \( \frac{1}{6} m(n + 1) \) when \( m = 2(n) \), the number of edges in a complete symmetric digraph.

In Section 4.2, we will see that, in Property 3.1.3, equality also holds for Eulerian tournaments.

Next, we relate the size of a maximum linear arrangement of a digraph \( D \) to the size of a maximum directed cut in \( D \).

**Theorem 3.1.4.** If \( D \) is a digraph on \( n \) vertices and \( \text{MaxDiCut}(D) = t \), then

\[
\frac{1}{2} nt \leq \text{MaxDLA}(D) \leq (n - 1)t.
\]

**Proof.** For the upper bound, observe that a linear arrangement of \( D \) contains exactly \( n - 1 \) cuts, none of which is larger than \( t \).

For the lower bound, let \( E(X, Y) \) be a maximum directed cut of \( D \), and construct \( D' \) from \( D \) by deleting edges not in \( E(X, Y) \). By construction, \( D' \) is an orientation of a bipartite graph with all edges directed from \( X \) to \( Y \). We build an arrangement \( \pi \) of \( D' \) with value at least \( \frac{1}{2} nt \). Let \( \pi_x = (x_1, x_2, ..., x_k) \) be an arrangement of \( X \) by non-increasing out-degree in \( D' \). Similarly, let \( \pi_y = (y_1, y_2, ..., y_l) \) be an arrangement of \( Y \) by non-decreasing in-degree in \( D' \). Then we claim \( \pi = (x_1, x_2, ..., x_k, y_1, y_2, ..., y_l) \) is an arrangement of \( D \) satisfying the lower bound.

Indeed, because \( E(X, Y) \) has size \( t \), and the vertices in \( X \) are arranged by non-decreasing out-degree, the value of the cut after \( x_i \) in \( \pi \) is at least \( \frac{t}{k} \). This means the sum of the first \( k \) cuts in \( \pi \) is at least \( \frac{t(k+1)}{2} \), and by symmetry the sum of the last \( l \) cuts in \( \pi \) is at least \( \frac{t(l+1)}{2} \). Summing these, and subtracting for the cut of size \( t \) between \( X \) and \( Y \) which was counted twice, we find that the value of \( \pi \) is at least \( \frac{t(k+1)}{2} + \frac{t(l+1)}{2} - t = \frac{1}{2} nt \). \( \square \)

The following corollary is immediate.

**Corollary 3.1.5.** The largest cut in a MaxDLA of digraph \( D \) has size at least \( \frac{1}{2} \text{MaxDiCut}(D) \).

**Proof.** Since the sum of cuts for an arrangement is at least \( \frac{1}{2} nt \), one of the cuts is at least \( \frac{nt}{2(n-1)} \geq t/2 \). \( \square \)

The next result (actually not a bound but an equality), relates the size of a minimum linear arrangement of a digraph to the size of a maximum linear arrangement of its complement. First, generalizing Equation 2.2 to digraphs, we state the following property.
**Property 3.1.6.** For an arrangement \( \pi \) of digraph \( D \),

\[
\text{val}_D(\pi) + \text{val}_{\overline{D}}(\pi) = \binom{n+1}{3}.
\]

Corollary 3.1.7 follows.

**Corollary 3.1.7.** For a digraph \( D \),

\[
\text{MaxDLA}(D) = \binom{n+1}{3} - \text{MinDLA}(\overline{D}).
\]

Furthermore, if \( \pi \) is a maximum arrangement of \( D \), then \( \pi \) is a minimum arrangement of \( \overline{D} \).

Finally, we bound the size of a maximum arrangement of a digraph \( D \) based on the degrees of its vertices. First, in Property 3.1.8, we bound the size of a cut in an arrangement of \( D \).

**Property 3.1.8.** If \( D \) is a digraph with out-degree sequence \( d_1^+ \geq d_2^+ \geq ... \geq d_n^+ \), and in-degree sequence \( d_1^- \leq d_2^- \leq ... \leq d_n^- \), then for any arrangement of \( D \) the size of the \( i \)th cut is limited by

\[
c_i \leq \min\left( \sum_{j=1}^{i} d_j^+, \sum_{j=i+1}^{n} d_j^- \right).
\]

**Proof.** We prove that \( c_i \leq \sum_{j=1}^{i} d_j^+ \). The proof that \( c_i \leq \sum_{j=i+1}^{n} d_j^- \) is similar. The argument is that the tail of any edge contributing to \( c_i \) is one of the first \( i \) vertices in that arrangement, and so \( c_i \) cannot be larger than the sum of the largest \( i \) out-degrees in \( D \).

\[
c_i = \sum_{j=1}^{i} l(v_j) = \sum_{j=1}^{i} \left( d_j^+(v_i) - |N(v_i) \cap S_i| \right) \leq \sum_{j=1}^{i} d_j^+
\]

The first equality follows from Equation 1.5, and the second from Equation 3.2. \( \square \)

Corollary 3.1.9 bounds the size of an arrangement of digraph \( D \) based on the degrees of vertices in \( D \). It follows from Equation 1.5 and Property 3.1.8.

**Corollary 3.1.9.** If \( D \) is a digraph, then

\[
\text{MaxDLA}(D) \leq \sum_{i=1}^{n-1} \min\left( \sum_{j=1}^{i} d_j^+, \sum_{j=i+1}^{n} d_j^- \right).
\]
3.2 The Weighted Maximum Linear Arrangement Problem, and the Signature of an Arrangement

In this section, we explore two concepts which will be essential for the algorithm in Section 3.3. Those concepts concern a new problem, which is a generalization of the MaxDLA problem, and the signature of an arrangement. The signature of an arrangement gives rise to a partial order which is further explored in Chapter 4.

The Weighted Maximum Linear Arrangement Problem

We begin with a generalization of the MaxDLA problem to a new problem, the Weighted Maximum Linear Arrangement problem. This abstraction is inspired by the concept of levels.

Let \( \pi = (v_1, v_2, ..., v_n) \) be an arrangement of digraph \( D \). Given vertex \( v_i \), its level in \( \pi \) can be computed as the number of its in-neighbours to its left subtracted from the number of its out-neighbours to its right \( l(v_i) = |N^+(v_i) \cap T_i| - |N^-(v_i) \cap S_i| \). This is the definition of directed cut and the definition of level in Equation 1.6. From this, by adding and subtracting \( |N^+(v_i) \cap S_i| \), it follows that the level of a vertex is
\[
    l(v_i) = d^+(v_i) - |N(v_i) \cap S_i|.
\]

Therefore, the levels of the vertices in an arrangement \( \pi \) can be calculated without knowing the direction of the edges of \( D \), as long as we know the underlying graph \( G \) of \( D \), and the out-degree of each vertex in \( D \). With this in mind, we generalize to a new problem: the Weighted Maximum Linear Arrangement (W-MaxLA) problem, which contains the MaxDLA problem.

The \textit{weighted} level of a vertex is defined as
\[
    l_{G,f}(v_i) = f(v_i) - |N(v_i) \cap S_i|, \tag{3.3}
\]
the value of the cut after vertex \( v_i \) is \( c_i = l(v_i) + l(v_2) + ... + l(v_i) \), and the value of a weighted arrangement is defined as
\[
    val_{G,f}(\pi) = \sum_{i=1}^{n} c_i = \sum_{i=1}^{n} (n - i + 1)l_{G,f}(v_i). \tag{3.4}
\]

We let \( \mathbb{Z} \) be the set of integers, and define the Weighted Maximum Linear Arrangement problem.

\textbf{Weighted Maximum Linear Arrangement (W-MaxLA)}

\textbf{Input:} A graph \( G \), a weight function \( f : V(G) \mapsto \mathbb{Z} \), integer \( k \).

\textbf{Question:} Is there an arrangement of \( G \) with value at least \( k \)?
Observation 3.2.1. Let $G = (V,E)$ be the underlying graph of digraph $D$, $f$ map the vertices of $G$ to their out-degree in $D$, and $\pi$ be an arrangement of $D$. Then the following are true:

1. $l_D(v) = l_{G,f}(v)$ for all $v \in V(D)$.
2. $val_D(\pi) = val_{G,f}(\pi)$
3. $MaxDLA(D,k) = W\cdot MaxLA(G,f,k)$.

There remains one subtlety when calculating the value of a weighted arrangement. In any arrangement of a digraph $D$, the last cut, $c_n = 0$. By abstracting to $W\cdot MaxLA$ this is no longer always the case. Hence, in Property 1.3.1 the summations go to $(n - 1)$, while in Equation 3.4 the summations go to $n$.

It is sometimes convenient to consider an instance of a MaxDLA problem as an instance of a W-MaxDLA problem. As an example, consider the following property.

Property 3.2.2. Let $G$ be a graph, and let $D$ and $D'$ be orientations of $G$ so that every vertex has the same out-degree in both $D$ and $D'$. If $\pi$ is an arrangement of $D$, then $val_D(\pi) = val_{D'}(\pi)$.

Proof. Let $f : V(G) \mapsto \mathbb{Z}$ be defined as $f(v) = d_D^+(v) = d_{D'}^+(v)$. Then $val_D(\pi) = val_{G,f}(\pi) = val_{D'}(\pi)$. \hfill \qedsymbol

Note that, in particular, reversing a directed cycle has no effect on the value of an arrangement.

The abstraction of MaxDLA to $W\cdot MaxLA$ will be an important part of the algorithm in Section 3.3.

The Signature of an Arrangement

The second important concept is the notion of the signature of an arrangement. The signature $s$ of an arrangement $\pi = (v_1, v_2, ..., v_n)$ is the $n$-tuple

$$s(\pi) = (s_1, s_2, ..., s_n),$$

where $s_i$ is the size of the $i^{th}$ cut in $\pi$, i.e., $s_i = c_i$ (cf. Equation 1.4). We specify the digraph $D$ or graph $G$ with weight function $f$ using subscripts when necessary, denoting their signatures by $s_D$ or $s_{G,f}$ respectively.

The value of $s$ is defined in the natural way, as the sum of its coordinates,

$$val(s) = \sum_{i=1}^{n} s_i,$$

so that $val(s(\pi)) = val(\pi)$. 

30
For a digraph $D$ (or graph $G$ with weight function $f$), let $S_D$ (or $S_{G,f}$) be the set of signatures of $D$ (or $G, f$). We omit the subscript when convenient. That is, $s \in S$ whenever there is an arrangement $\pi$ so that $s(\pi) = s$. For $s, s' \in S$ we write $s' \leq s$ whenever $s'_i \leq s_i$ for all $i$. This notation captures the idea of $s$ being no worse than $s'$ on every cut. It is easy to see that $S$ is a partial order under $\leq$.

We say an arrangement $\pi$ is maximal if $s(\pi)$ is maximal in the partial order $S$. The following are properties of maximal arrangements.

**Property 3.2.3.** Let $D$ be a disconnected digraph and let $H$ be one of its components. An arrangement $\pi$ is a maximal arrangement of $D$ only if $\pi$ restricted to $H$ is a maximal arrangement of $H$.

**Proof.** By contrapositive. Suppose $H$ was not maximal. Then choose an arrangement of $H$ that is strictly better than $\pi$ restricted to $H$, and rearrange the vertices of $H$ within $\pi$ accordingly. This new arrangement is strictly better than $\pi$. \qed

**Property 3.2.4.** The vertices of a maximal arrangement are arranged by levels, in non-increasing order.

**Proof.** Suppose, for a contradiction, that $\pi = (v_1, v_2, ..., v_n)$ is a maximal arrangement having vertices $v_i, v_j$ so that $i < j$ but $l(v_i) < l(v_j)$. We may assume, without loss of generality, $j = i + 1$. But then interchanging $v_i$ and $v_j$ increases the value of $c_i$, while leaving the other cuts unchanged. This contracts the maximality of $\pi$. \qed

**Property 3.2.5.** There is no edge between vertices at the same level in a maximal linear arrangement.

**Proof.** Suppose, for a contradiction, that $\pi = (v_1, v_2, ..., v_n)$ is a maximal arrangement having vertices $v_i, v_j$ so that $l(v_i) = l(v_j)$ and there is an edge between $v_i$ and $v_j$. By Property 3.2.4, the vertices between $v_i$ and $v_j$ in $\pi$ also have level $l(v_i)$, so we may assume $v_i$ and $v_j$ have no neighbours between them in $\pi$. But then swapping $v_i$ and $v_j$ in $\pi$ strictly increases the size of the cuts from $C_i$ to $C_{j-1}$, and does not change any other cut. This contradicts the maximality of $\pi$. \qed

The following is a corollary to Property 3.2.4 and Property 3.2.5.

**Corollary 3.2.6.** Vertices within the same level of a maximal arrangement may be interchanged without changing the signature of that arrangement.

### 3.3 Bounded-Degree Trees

In this section, we describe an algorithm solving MaxDLA on orientations of trees with degrees bounded by a constant. This same algorithm, with a slight modification described in Section 3.4, solves MinLA on graphs $G$ when $\overline{G}$ is a bounded-degree tree.
The algorithm will work with *sorted level signatures*, which are similar to signatures (of cuts, cf. Equation 3.5), but useful in the algorithm to save computation time. We first define *level signature*, and then *sorted level signature*.

The *level signature* $l$ of an arrangement $\pi = (v_1, v_2, ..., v_n)$ is the $n$-tuple of the levels of its vertices,

$$l(\pi) = (l_1, l_2, ..., l_n),$$  \hspace{1cm} (3.7)

where $l_i = l(v_i)$. The value of $l$ is defined in the natural way (cf. Equation 3.4),

$$\text{val}(l(\pi)) = \sum_{i=1}^{n} (n - i + 1)l_i$$ \hspace{1cm} (3.8)

so that $\text{val}(l(\pi)) = \text{val}(s(\pi)) = \text{val}(\pi)$. The level signature of $\pi$ contains the same information as the signature of $\pi$, and it is straightforward to convert between the two. Indeed, by Equation 1.6, if $s(\pi) = (c_1, c_2, ..., c_n)$ then $l(\pi) = (c_1 - 0, c_2 - c_1, ..., c_n - c_{n-1})$.

The *sorted level signature* $l^*$ of an arrangement $\pi = (v_1, v_2, ..., v_n)$ is obtained from $l(\pi)$ by sorting it into a non-decreasing order,

$$l^*(\pi) = (l^*_1, l^*_2, ..., l^*_n),$$ \hspace{1cm} (3.9)

where $l^*_1 \geq l^*_2 \geq ... \geq l^*_n$. The value of $l^*$ is defined analogously to the value of $l$,

$$\text{val}(l^*(\pi)) = \sum_{i=1}^{n} (n - i + 1)l^*_i.$$ \hspace{1cm} (3.10)

Note that, in general, $\text{val}(l(\pi))$ may be different from $\text{val}(l^*(\pi))$, however maximal arrangements are a special case. Recall (Property 3.2.4) that the vertices of a maximal linear arrangement are arranged by non-increasing order of levels. Hence, if $\pi$ is a maximal arrangement, then $l(\pi) = l^*(\pi)$.

Our goal is to describe an algorithm solving MaxDLA on directed tree $D$ (whose maximum degree is bounded by a constant $d$), and integer $k$. However, we describe instead an algorithm solving W-MaxLA for undirected tree $G$ (whose maximum degree is bounded by $d$), vertex weight function $f$ (which is also bounded by $d$), and integer $k$. The W-MaxLA problem contains the MaxDLA problem by Observation 3.2.1.

The algorithm is recursive. We make an observation which allows us to remove an edge from the tree, and hence break it apart in Step 2 and Step 3 of Algorithm 3.3.2.

**Observation 3.3.1.** Let $G = (V, E)$ be an undirected graph, $e = uv \in E(G)$, $f: V(G) \rightarrow \mathbb{Z}$ be a weight function, and $\pi$ be an arrangement of $V(G)$ with $\pi(u) < \pi(v)$. Define $G' = G - e$ and $f'$ as follows:

$$f'(x) = \begin{cases} f(x) - 1 & \text{if } x = v \\ f(x) & \text{otherwise.} \end{cases}$$ \hspace{1cm} (3.11)
Then the signature of $\pi$ is unchanged under either $G, f$ or $G', f'$, that is $s_{G,f}(\pi) = s_{G',f'}(\pi)$.

Proof. We prove that for every $x \in V(G)$, the level of $x$ in $\pi$ is unchanged under either $G, f$ or $G', f'$, that is $l_{G,f}(x) = l_{G',f'}(x)$.

If $x \neq u, v$ then

$$l_{G,f}(x) = f(x) - |\{y \in N_G(x) : \pi(y) < \pi(x)\}|$$

$$= f'(x) - |\{y \in N_{G'}(x) : \pi(y) < \pi(x)\}|$$

$$= l_{G',f'}(x).$$

The level of $u$ is also unchanged because

$$l_{G,f}(u) = f(u) - |\{y \in N_G(u) : \pi(y) < \pi(u)\}|$$

$$= f'(u) - |\{y \in N_{G'}(u) : \pi(y) < \pi(u)\}|$$

$$= l_{G',f'}(u),$$

where the second equality follows because $\pi(u) < \pi(v)$. Finally, the level of $v$ is unchanged because

$$l_{G,f}(v) = f(v) - |\{y \in N_G(v) : \pi(y) < \pi(v)\}|$$

$$= f'(v) + 1 - |\{y \in N_{G'}(v) : \pi(y) < \pi(v)\}| - 1$$

$$= l_{G',f'}(u),$$

where the second equality follows, again, because $\pi(u) < \pi(v)$.

The theorem follows because a cut is the sum of preceding levels (Equation 1.7) and because a signature is a tuple of cuts (Equation 3.5).

We describe an algorithm which, given a bounded-degree undirected tree $G$, and bounded weight function $f$ on its vertices, outputs a set of sorted level signatures of $G, f$. Figure 3.1 shows an example flow of this algorithm for a path of length three.

**Algorithm 3.3.2.** FindSignatures($G, f$)

INPUT: A tree $G = (V,E)$ on $n$ vertices with maximum degree $d$, and a weight function $f$ from $V(G)$ to the positive integers, where $f(v) \leq d$, with $d$ a constant.

OUTPUT: A set $S$ of sorted level signatures of $G, f$, so that the largest value of an element in $S$ is equal to $W$-$\text{MaxDLA}(G, f)$.

Step 1 If $G$ is an isolated vertex $v$, set $S = \{(f(v))\}$ and terminate.

Step 2 If $G$ is not an isolated vertex, find an edge $e = (u, v)$ that minimizes the size of the largest component when $e$ is deleted.
Step 3 Define two new weight functions $f_u$ and $f_v$ so that

$$f_u(x) = \begin{cases} f(x) - 1 & \text{if } x = u \\ f(x) & \text{otherwise} \end{cases}$$

(3.12)

and

$$f_v(x) = \begin{cases} f(x) - 1 & \text{if } x = v \\ f(x) & \text{otherwise} \end{cases}$$

(3.13)

Step 4 Recursively calculate the sorted level signatures of $G - e$ with both $f_u$ and $f_v$. Do this in the following way. Let $G_1$ and $G_2$ be the two components of $G - e$. Make four recursive calls as follows:

$$S_{1,u} = \text{FindSignatures}(G_1, f_u),$$
$$S_{2,u} = \text{FindSignatures}(G_2, f_u),$$
$$S_{1,v} = \text{FindSignatures}(G_1, f_v),$$
$$S_{2,v} = \text{FindSignatures}(G_2, f_v).$$

Step 5 Combine $S_{1,u}$ and $S_{2,u}$ to form $S_u$ by combining every sorted level signature in $S_{1,u}$ with every sorted level signature in $S_{2,u}$ as follows. If $l = (l_1, l_2, ..., l_p) \in S_{1,u}$ and $l' = (l'_1, l'_2, ..., l'_q) \in S_{1,u}$ then their new combined sorted level signature is obtained from $(l_1, l_2, ..., l_p, l'_1, l'_2, ..., l'_q)$ by sorting it into a non-increasing order. Delete duplicate signatures. Do the same for $S_{1,v}$ and $S_{2,v}$ to form $S_v$.

Step 6 Let $S = S_u \cup S_v$.

Step 7 Return $S$.

On completion of Algorithm 3.1, it is simple to check if there is a sorted level signature in the output $S$ with value at least $k$.

**Correctness**

We analyze the correctness of Algorithm 3.1. Let $S$, $G$, $f$, and $D$ be as described in the algorithm. It follows from Observation 3.3.1 that every element in the output $S$ is a sorted level signature of $G$, $f$. Therefore, by Observation 3.2.1, every element in $S$ is also a sorted level signature of $D$. We must show that the maximum value of a sorted level signature in $S$ is equal to $\text{MaxDLA}(D) = W-\text{MaxDLA}(G, f)$.

**Theorem 3.3.3.** Let $S$ be the final set returned by Algorithm 3.1 when tree $G$ and weight function $f$ are input to the algorithm. Then $W-\text{MaxDLA}(G, f)$ is the maximum value of a sorted level signature in $S$. 

34
Proof. Denote the maximum value of a sorted level signature in \( S \) by \( l^\text{max} \). First, we show that \( l^\text{max} \geq W-\text{MaxDLA}(G, f) \). Let \( \pi \) be an arrangement of \( G, f \) so that \( \text{val}(\pi) = W-\text{MaxDLA}(G, f) \). There is a path in the recursion where the new function defined at Step 3 is consistent with the order of the vertices in \( \pi \). This path results in a sorted level signature, included in \( S \), whose value is equal to \( W-\text{MaxDLA}(G, f) \). Therefore the maximum value of a sorted level signature in \( S \) is at least \( W-\text{MaxDLA}(G, f) \).

We show that \( l^\text{max} \leq W-\text{MaxDLA}(G, f) \). Let \( l^* = (l^*_1, l^*_2, ..., l^*_n) \in S \) be a sorted level signature whose value is \( l^\text{max} \). Let \( v_i \) be the vertex associated with \( l^*_i \). That is, at the base of the recursion in Step 1, \( v_i \) has level \( l^*_i \).

Claim. Let \( e = v_i v_j \in E(G) \) with \( i < j \). The path \( P \) in the recursion resulting in \( l^* \) chose (at Step 3) \( v_i \) before \( v_j \).

Suppose the claim is false, and \( P \) chose \( v_j \) before \( v_i \). Let \( P' \) be a path in the recursion making identical choices to \( P \) at every edge except edge \( e \). Suppose \( P' \) chose instead \( v_i \) before \( v_j \), and let \( l'^* \) be the sorted level signature resulting from \( P' \). Then the level of every vertex is the same in both \( l^* \) and \( l'^* \), except \( v_i \) and \( v_j \). The level of \( v_i \) in one greater, and the level of \( v_j \) is one less, in \( l'^* \) than in \( l^* \). But since \( v_i \) precedes \( v_j \) in \( l^* \), it means \( l'^*_i \geq l^*_j \).

It follows that \( \text{val}(l'^*) > \text{val}(l^*) \), which contradicts that \( \text{val}(l^*) = l^\text{max} \).

By the claim, \( l^* \) has vertices ordered consistently with the way the edges were arranged in Step 3 of the algorithm. Therefore \( l^* \) has the same value as the (unsorted) level signature determined by arranging the edges in the same way. Since an (unsorted) level signature has value no more than \( W-\text{MaxDLA}(G, f) \) this means \( l^* \) also has value no more than \( W-\text{MaxDLA}(G, f) \). Therefore \( l^\text{max} \leq W-\text{MaxDLA}(G, f) \).

\[
\square
\]

Running Time

We show that the running time is \( O(n^{4d}) \).

Step 2 can be done in time linear in \( n \) by, for example, moving from the leaves upward and orienting each edge of the tree toward the largest subtree when that edge is deleted. Step 3 can be done in constant time.

In Step 4 of Algorithm 3.3.2, four recursive calls are made, and the number of vertices in the tree for these calls has size at most \( \frac{(n-1)(d-1)}{d} + 1 \) where \( n = |V(G)| \). This means that at each step of recursion the size of the problem (in terms of number of vertices) is reduced by a factor of at least \( \frac{d}{d+1} \).

Step 5 and Step 6 take up the majority of computation time. But this remains polynomial because we are only keeping track of sorted level signatures. A sorted level signature is concerned only with the number of vertices at each level. There are \( 2d + 1 \) possible levels, so only \( O(n^{2d}) \) possible signatures. This means at Step 5, that \( |S_{1,u}| \) and \( |S_{2,u}| \) are \( O(n^{2d}) \) and so can be combined in time \( O(n^{4d}) \). At Step 6, again, \( |S| \) is \( O(n^{2d}) \), so this step can
also be done in time $O(n^{4d})$. The bound of $O(n^{4d})$ dwarfs the time required to sort and delete duplicates in Step 5 and Step 6.

This leads to a recurrence of $T(n) = 4T(\frac{d}{d+1}n) + O(n^{4d})$, and running time $O(n^{4d})$. Therefore, the running time of Algorithm 3.3.2 is $O(n^{4d})$.

![Flow of FindSignatures](image)

Figure 3.1: Flow of FindSignatures $(G, f)$ when solving for the above orientation of a path of length three.

### 3.4 Complements of Bounded-Degree Trees

We show how Algorithm 3.3.2 can be used to solve MinLA on undirected graphs $G$ when $\overline{G}$ is a tree with degrees bounded by a constant.

This complements results from the 1970’s and 1980’s showing that the MinLA problem is polynomial-time solvable on trees. The results on general trees began with a paper by Goldberg and Klipker [16] in 1976. They discovered an $O(n^3)$ algorithm solving MinLA on trees. Before that, work had been done on restricted trees such as special bounded-degree trees [36, 25] and (in 1978) complete $k$-level binary trees [8]. Goldberg and Klipker’s result
was improved to $O(n^{2.2})$ by Shiloach [38] in 1979, and further improved to $O(n^\lambda)$ where 
$\lambda$ is approximately $\log(3)/\log(2) \approx 1.6$ by Chung [9] in 1984. To the best of this author’s
knowledge, Chung’s algorithm has not been improved. Chung noted that there is a lower bound of $\Omega(n \log n)$ because sorting is required [9, p. 55].

We describe how to modify Algorithm 3.3.2 to solve MaxLA for bounded-degree undirected trees. Then by Corollary 3.1.7 this is equivalent to solving MinLA on their complements.

First, we modify Observation 3.3.1 by allowing graph $G$ to be a multigraph, with $k$ edges
between $u$ and $v$.

**Observation 3.4.1.** Let $G = (V, E)$ be an undirected multigraph, $e = uv \in E(G)$ with
multiplicity $k$, $f : V(G) \mapsto \mathbb{Z}$ be a weight function, and $\pi$ be an arrangement of $V(G)$ with
$\pi(u) < \pi(v)$. Define $G' = G - e$ and $f'$ as follows:

$$f'(x) = \begin{cases} 
  f(x) - k & \text{if } x = v \\
  f(x) & \text{otherwise}.
\end{cases} \quad (3.14)$$

Then the signature of $\pi$ is unchanged under either $G, f$ or $G', f'$, that is $s_{G,f}(\pi) = s_{G',f'}(\pi)$.

The proof of the above observation is similar to the proof of Observation 3.3.1, and so we skip it.

To solve MinLA on an undirected graph $G$ where $\overline{G}$ is a bounded-degree tree, and integer $k$, use Algorithm 3.3.2 with the following two modifications:

1. Input $\overline{G}, f$ where $f : V(G) \mapsto \mathbb{Z}$ so that $f(v) = d_{\overline{G}}(v)$.

2. In Step 3, decrement the weight function by two instead of one.

The first modification is based on, as precisely mentioned, the fact that an arrange-
ment is minimum for $G$ exactly when it is maximum for $\overline{G}$ (Corollary 3.1.7). The second
modification follows from Observation 3.4.1 by considering $\overline{G}$ as a symmetric digraph.

It follows that $G$ has an arrangement of size at most $k$ if and only if one of the level
signatures output from modified Algorithm 3.3.2 has value at least $(\frac{n+1}{3}) - k$.

**Theorem 3.4.2.** MinLA is polynomial-time solvable on graphs $G$ where $\overline{G}$ is a tree with
degree bounded by a constant.
Chapter 4

Nested-Minimum and Nested-Maximum Graphs

Inspired by a theorem of Harper exhibiting cuts of the hypercube minimum for every cardinality [21], we look for graphs with an arrangement so that every cut is maximum for its position in the arrangement.

First, we make these notions explicit. Recall the partial order $S$, coordinatewise on cut signatures, defined at the end of section 3.2. We have thus far considered maximal signatures, but in this chapter we will consider digraphs $D$ (or graphs $G$ with weight function $f$) for which the partial order $S$ has a greatest (or least) element. We call such digraphs (or graphs with weight function) nested-maximum and nested-minimum respectfully. We call an arrangement of $D$ (or $G, f$) whose signature is the greatest (least) element of $S$ a nested-maximum (respectfully nested-minimum) arrangement. In this language, Harper’s theorem [21] is that a hypercube is nested-minimum.

This chapter proceeds as follows. In Section 4.1, we prove Harper’s theorem, and observe that a hypercube is also nested-maximum. In Section 4.2, we demonstrate three classes of nested-maximum digraphs: tournaments, orientations of graphs $G$ with maximum degree at most 2, and transitive acyclic digraphs.

4.1 Harper’s Isoperimetric Inequality for the Hypercube: The Hypercube is Nested-Minimum

In this section, we prove a classical result of Harper in Theorem 4.1.2, that a hypercube is nested-minimum. We observe at the end of the section that a hypercube is also nested-maximum. We begin with some preliminaries and a lemma.

Let $Q_d$ be a hypercube as defined in Section 1.3. For $v = (v_1, v_2, ..., v_d) \in V(Q_d)$, we say $b(v)$ is the binary number $v_1v_2...v_d$. That is,

$$b(v) = 2^0v_d + 2^1v_{d-1} + ... + 2^{d-1}v_1 = \sum_{i=1}^{d} 2^{d-i}v_i.$$
For \( S \subseteq V(Q_d) \), we say
\[
b(S) = \sum_{v \in S} b(v).
\]
We define \( L_t \subseteq V(Q_d) \) as the first \( t \) vertices of \( Q_d \) when ordered lexicographically. That is, \( L_t \) is the set of vertices of size \( t \) which minimizes \( b(L_t) \). For \( x, y \in V(Q_d) \) where \( x = (x_1, x_2, ..., x_d) \) and \( y = (y_1, y_2, ..., y_d) \), we say \( x \) and \( y \) agree on coordinate \( i \) when \( x_i = y_i \).

**Lemma 4.1.1.** Let \( S \in V(Q_d) \) be of size \( t \), \( S \neq L_t \). Then either

1. there exists \( x, y \in V(Q_d) \) so that \( x \notin S, y \in S \), \( b(x) < b(y) \), and \( x, y \) agree on some coordinate, or
2. \( S = \{(0, ..., 0), (0, ..., 0, 1), ..., (0, 1, ..., 1), (1, 0, ..., 0)\} = L_t \cup (1, 0, ..., 0) - (0, 1, ..., 1) \).

**Proof.** Suppose 1 does not hold. We must show that 2 holds. Since \( S \neq L_t \), there exists \( x \notin S, y \in S \) such that \( b(x) < b(y) \). By assumption, \( x \) and \( y \) do not agree on any coordinate. But \( x \) is the unique vertex disagreeing with \( y \) on every coordinate. Therefore \( x \) is the unique vertex preceding \( y \) lexicographically that is not in \( S \), otherwise 1 would hold. Similarly, \( y \) is the unique vertex following \( x \) lexicographically that is in \( S \). This means \( y \) immediately follows \( x \) lexicographically. The only pair of vertices both disagreeing on every coordinate and occurring consecutively lexicographically are \( x = (0, 1, ..., 1) \) and \( y = (1, 0, ..., 0) \). Therefore \( S \) is as described in 2. \( \square \)

This author bases the proof of Theorem 4.1.2 on a presentation by Matt DeVos.

**Theorem 4.1.2** (Harper [21]). Recall that \( \delta(S) \) denotes the edge boundary of a subset of vertices \( S \). Let \( Q_d = (V, E) \) be a hypercube. If \( S \subseteq V \) has size \( t \), then \( \delta(S) \geq \delta(L_t) \).

**Proof.** Suppose the theorem is false. Let \( S \subseteq V(Q_d) \) with \( \delta(S) < \delta(L_{|S|}) \). Further suppose that \( d \) and \( S \) are chosen so that \( b(S) \) is minimum over all counterexamples to the theorem. Let \( t = |S| \).

Since \( S \neq L_t \), there exists a pair of vertices \( x \) and \( y \) so that \( x \notin S, y \in S \), and \( b(x) < b(y) \). If \( S \) is as in 2 of Lemma 4.1.1, it is easy for the reader to verify that \( \delta(S) \geq \delta(L_t) \). So we may assume that is not the case. Hence 1 of Lemma 4.1.1 holds. Let \( x, y \in V(Q_d) \) so that \( x \notin S, y \in S, b(x) < b(y) \), and \( x, y \) agree on coordinate \( i \).

Partition \( S \) into two sets as follows:

\[
A = \{v \in S : v_i = \overline{x_i}\}
\]
\[
B = \{v \in S : v_i = x_i\}.
\]

Define \( F = \{uv \in E(Q_d) : u \in A \text{ and } v \in B\} \). Now observe that
\[
\delta(S) = \delta(A \cup B) = \delta(A) + \delta(B) - 2|F|.
\] (4.1)
We will replace $S$ with a set $S'$ that is no worse than $S$, that is $\delta(S') \leq \delta(S)$. Observe that for every vertex in $A$ (the same is true for $B$) if we temporarily delete their $i^{th}$ coordinates and obtain a set $A_{\text{del}}$ that is a subset of vertices of a hypercube with dimension $d - 1$, then, by minimality of $S$,

$$\delta(L_{|A|}) \leq \delta(A_{\text{del}}).$$

Now, we define $A'$ from $L_{|A|}$ by reinserting $x_i$ in the $i^{th}$ coordinate. In other words, we let $A' = \{ v \in V : v_i = x_i \}$ and $(v_1, v_2, ..., v_{i-1}, v_{i+1}, ..., v_d) \in L_{|A|}$.

Now, $b(L_{|A|}) \leq b(A_{\text{del}})$, so it follows that

$$b(A') \leq b(A). \quad (4.3)$$

This is because $A'$ is obtained from $L_{|A|}$ in the same way that $A$ is obtained from $A_{\text{del}}$, which is by inserting $x_i$ in the $i^{th}$ coordinate.

Similarly, it follows from (4.2) that

$$\delta(A') = \delta(L_{|A|}) + |A| \leq \delta(A_{\text{del}}) + |A| = \delta(A). \quad (4.4)$$

Construct $B'$ from $B$ analogously to the construction of $A'$ from $A$, and say $S' = A' \cup B'$ and $F' = \{ uv \in E(Q_d) : u \in A' \text{ and } v \in B' \}$. Observe that, as in (4.3), (4.4) and (4.1) respectively, we have

$$b(B') < b(B), \quad (4.5)$$

$$\delta(B') \leq \delta(B), \text{ and} \quad (4.6)$$

$$\delta(S') = \delta(A' \cup B') = \delta(A') + \delta(B') - 2|F'|. \quad (4.7)$$

Importantly, note that because $y \in B$, the inequality in (4.5) is strict.

We will now show that $|F'| \geq |F|$. This follows because a vertex in $A$ is adjacent to at most one vertex in $B$. In particular, $a \in A$ is adjacent to $b \in B$ if and only if $a_j = b_j$ for all $j \neq i$. Hence $|F| \leq \min(|A|, |B|)$. But $|F'| = \min(|A'|, |B'|) = \min(|A|, |B|)$ by construction. Therefore,

$$|F'| \geq |F|. \quad (4.8)$$

Now combining (4.1), (4.4), (4.6), (4.8), and (4.7), it follows that

$$\delta(S) = \delta(A) + \delta(B) - 2|F| \geq \delta(A') + \delta(B') - 2|F'| = \delta(S'). \quad (4.9)$$

But, by (4.3) and (4.5),

$$b(S') = b(A') + b(B') < b(A) + b(B) = b(S).$$
Therefore, by minimality of $S$ and (4.9),

$$\delta(L_t) \leq \delta(S') \leq \delta(S).$$

A fact similar to Harper’s theorem is that a hypercube is also nested-maximum. This follows because a hypercube is a regular bipartite graph. We elaborate below. In the following discussion, we say the parity of a vertex $v = (v_1, v_2, ..., v_d) \in V(Q_d)$ is equal to the parity of $|\{i : v_i = 1\}|$ (the number of coordinates of $v$ equal to 1).

**Lemma 4.1.3.** If $G$ is a regular bipartite graph, then $G$ is nested-maximum. An arrangement $\pi$ of $G$ is nested-maximum if and only if, for some bipartition $(A, B)$ of $G$, $\pi$ places all vertices of $A$ before all vertices of $B$.

**Proof.** Considering $G$ as a symmetric digraph, this follows from Property 3.1.8. Indeed, for regular symmetric digraphs where every vertex has out-degree $k$, Property 3.1.8 means that the value of the $i$th cut of an arrangement is no larger than $\min(ik, (n - i)k)$. An arrangement of $G$ which places all the vertices of $A$ before all the vertices of $B$ achieves this upper bound, and is therefore nested-maximum. Note that a nested-maximum arrangement of $G$ contains $\text{MaxCut}(G)$. Any arrangement which fails to separate the vertices by a bipartition $(A, B)$ will not contain $\text{MaxCut}(G)$, and is therefore not nested-maximum.

**Theorem 4.1.4.** A hypercube $Q_d$ is nested-maximum. An arrangement $\pi$ of $Q_d$ is nested-maximum if and only if the first half of the vertices in $\pi$ all have the same parity.

**Proof.** By Observation 4.1.3 we need only show that $Q_d$ is a regular bipartite graph, and that all bipartitions of $Q_d$ partition the vertices by parity.

By construction, $Q_d$ is regular of degree $d$. Further, any two vertices of the same parity in $Q_d$ are not adjacent. Indeed, by the definition of hypercube, vertices $u, v \in V(Q_d)$ are adjacent if and only if they differ in exactly one coordinate. But differing in exactly one coordinate means they cannot have the same parity. Hence $Q_d$ is bipartite, and two bipartitions of $Q_d$ partition the vertices by parity (odd vs. even and even vs. odd). Because $Q_d$ is connected, these are the only two bipartitions of $Q_d$.

### 4.2 Nested-Maximum Digraphs

In this section, three classes of nested-maximum digraphs are presented: tournaments, orientations of graphs with maximum degree at most two, and transitive acyclic digraphs. Tournaments and transitive acyclic digraphs are also nested-minimum.
Tournaments

Recall that a tournament is an orientation of a complete graph. We show that a tournament is both nested-maximum and nested-minimum. We also state the values of a maximum and minimum arrangement of a tournament given its degree sequence. The proof that follows is a good example of the usefulness of abstracting to W-MaxDLA.

**Theorem 4.2.1.** Let \( D \) be a tournament on \( n \) vertices with out-degree sequence \( d_1^+ \geq d_2^+ \geq \ldots \geq d_n^+ \). Then \( D \) is both nested-maximum and nested-minimum. Furthermore, the maximum value of an arrangement of \( D \) is

\[
MaxDLA(D) = \sum_{i=1}^{n-1} (n - i)d_i^+ - \binom{n}{3},
\]

and it is achieved when \( D \) is arranged by non-increasing out-degree. The minimum value of an arrangement of \( D \) is

\[
MinDLA(D) = \sum_{i=1}^{n-1} id_i^+ - \binom{n}{3},
\]

and it is achieved when \( D \) is arranged by non-decreasing out-degree.

**Proof.** We abstract to W-MaxLA, and, by Observation 3.2.1, solve the equivalent problem of arranging the complete graph on \( n \) vertices \( K_n \) with vertex weight function \( f(v) = d^+(v) \).

Let \( \pi \) be any arrangement of \( K_n, f \), and let \( \pi^+ = (v_1, v_2, \ldots, v_n) \) be an arrangement of \( K_n, f \) by non-increasing out-degree in \( D \), that is, so that \( f(v_i) \geq f(v_j) \) whenever \( i \leq j \). We will show that the \( i^{th} \) cut of \( \pi \) is no larger than the \( i^{th} \) cut of \( \pi^+ \). That is, we will show that \( c_i(\pi) \leq c_i(\pi^+) \).

Observe that the value of the \( i^{th} \) cut of \( \pi \) is

\[
c_i(\pi) = \sum_{j=1}^{i} l(v_j) = \sum_{j=1}^{i} (d^+(v_j) - |N(v_j) \cap S_j|)
\]

The first equality is Equation 1.7, and the second follows from the meaning of level (Equation 3.2). But \( |N(v_j) \cap S_j| = j - 1 \) because we are arranging a complete graph. Therefore it follows that

\[
c_i(\pi) = \sum_{j=1}^{i} d^+(v_j) - \frac{1}{2}i(i - 1)
\]

Now it is easy to see that \( \sum_{j=1}^{i} d^+(v_j) \leq \sum_{j=1}^{i} d_j^+ \) and so

\[
c_i(\pi) \leq \sum_{j=1}^{i} d_j^+ - \frac{1}{2}i(i - 1) = c_i(\pi^+).
\]
Thus arranging $D$ by non-increasing out-degree achieves the best possible value for every cut. Hence $D$ is nested maximum, and the value of a maximum arrangement of $D$ is given by the sum of its cuts,

$$\text{MaxDLA}(D) = \sum_{i=1}^{n-1} c_i = \sum_{i=1}^{n-1} \left( \sum_{j=1}^{i} d_j^+ - \frac{1}{2} i(i-1) \right) = \sum_{i=1}^{n-1} (n-i)d_i^+ - \binom{n}{3}.$$ 

Because the complement of a tournament is also a tournament, it follows from Corollary 3.1.7 that a tournament is also nested-minimum, and a minimum arrangement of a tournament is achieved when arranged by non-decreasing out-degree. The value of a minimum arrangement is calculated in a similar way to the calculation of a maximum arrangement.

It is interesting to note that, for an an Eulerian tournament $T$ on $n$ vertices, every arrangement of $T$ has value $\text{MaxDLA}(T) = \text{MinDLA}(T) = \frac{1}{2} m(n+1)$. Which is (necessarily) equal to the lower bound from Property 3.1.3 on the value of any $n$-vertex, $m$-edge loopless multi-digraph. Therefore, of all the loopless multi-digraphs on $n$ vertices and $m = \binom{n}{2}$ edges, an Eulerian tournament achieves the worst possible value in terms of MaxDLA.

**Orientations of Graphs with Maximum Degree Two**

We show that orientations of graphs with maximum degree at most two are nested-maximum. We do this by proving something stronger, that for any undirected graph $G$ with maximum degree at most two, and any weight function $f : V(G) \to \mathbb{Z}$, then $G, f$ has a nested-maximum arrangement. By Observation 3.2.1, this means that any orientation of $G$ is nested-maximum.

We begin with a lemma showing a condition which, when satisfied, means it is safe to assume one end of an edge comes before the other in a maximal arrangement of $G, f$.

**Lemma 4.2.2.** Let $G = (V, E)$ be a digraph and let $f : V(G) \to \mathbb{Z}$ be a weight function. If there exists an edge $e = uv \in E(G)$ so that

$$d(u) - 1 \leq f(u) - f(v), \quad (4.12)$$

then for any arrangement $\sigma$ of $G$, there is an arrangement $\pi$ of $G$ so that $\pi(u) < \pi(v)$ and $\sigma \leq \pi$.

**Proof.** Let $\sigma$ be an arrangement of $G$. We may assume $\sigma$ is a maximal arrangement, and so, by Property 3.2.4, the vertices are arranged by non-increasing level. If $\sigma(u) < \sigma(v)$ then $\sigma$ satisfies the lemma, and we are done. So we may assume $\sigma(v) < \sigma(u)$, and by maximality of $\sigma$, that $l_\sigma(v) \geq l_\sigma(u)$. But by Corollary 3.2.6, if $l_\sigma(v) = l_\sigma(u)$ we may simply interchange $u$ and $v$ in $\sigma$ without altering any cut of $\sigma$. So it must be that

$$l_\sigma(v) > l_\sigma(u). \quad (4.13)$$
But now,
\[ l_\sigma(u) \geq f(u) - d(u) \geq f(v) - 1 \geq l_\sigma(v) - 1 \geq l_\sigma(u), \tag{4.14} \]
where the first and third inequalities follow from the definition of weighted level (Equation 3.3), the second by assumption (Inequality 4.12), and the last from Inequality 4.13. This means
\[ l_\sigma(v) - 1 = l_\sigma(u). \tag{4.15} \]

By Corollary 3.2.6, we may permute vertices at the same level in \( \sigma \) so that
\[ \sigma(v) = \sigma(u) - 1. \tag{4.16} \]

Now simply interchanging \( u \) and \( v \) in \( \sigma \) doesn’t alter any cut. Observe that when \( u \) and \( v \) are interchanged, their levels are interchanged as well. This new arrangement satisfies the requirements of the lemma.

**Theorem 4.2.3.** Let \( G = (V, E) \) be a graph with \( \Delta(G) \leq 2 \), and \( f : V(G) \rightarrow \mathbb{Z} \) be a weight function. Then \( G \) is nested-maximum.

**Proof.** We proceed by induction on \( m = |E(G)| \). As a base case, assume \( |E(G)| = 0 \). Then \( G \) is just isolated vertices, and so every arrangement of \( G \) is necessarily maximum. For the inductive step, assume \( |E(G)| \geq 1 \).

If \( G \) is not connected, we may, by induction, take a nested-maximum arrangement of each component of \( G \). By Property 3.2.3, any maximal arrangement of \( G \) restricted to a component of \( G \) must be a nested-maximum arrangement of that component. Since, by Property 3.2.4, the levels of a maximal arrangement of \( G \) are non-increasing, this means there is only one signature of a maximal arrangement of \( G \). Therefore \( G \) is nested-maximum.

Hence we may assume \( G \) is connected, and so \( G \) is either a cycle or a path.

We define Property \( P \) as follows. An edge \( e = uv \in E(G) \) has Property \( P \) if for any arrangement \( \sigma \) of \( G \), there is an arrangement \( \pi \) of \( G \) so that \( \pi(u) < \pi(v) \) and \( \sigma \leq \pi \).

First, we show that \( G \) always has an edge with Property \( P \). We proceed by cases.

**Case 1:** For all edges \( e = uv \in E(G) \), it is the case that \( f(u) = f(v) \). Then, because \( G \) is connected, \( f \) is constant. If \( G \) is a cycle, then by symmetry every edge satisfies Property \( P \). If \( G \) is not a cycle, then \( G \) is a path. Let \( u \) be an end of \( G \) and \( v \) be the neighbour of \( u \). We claim that \( e = uv \) satisfies Property \( P \). Indeed, \( f(u) = f(v) \) by assumption, but \( d(u) = 1 \). So \( d(u) - 1 = 0 = f(u) - f(v) \), and Lemma 4.2.2, shows \( e = uv \) has property \( P \).

**Case 2:** There exists an edge \( e = uv \in E(G) \) such that \( f(u) > f(v) \). Then, because \( d(u) \leq 2 \) it follows that \( d(u) - 1 \geq 1 \leq f(u) - f(v) \). Hence, from Lemma 4.2.2 it must be that \( e = uv \) has property \( P \).

Therefore, in all cases, \( G \) has an edge \( e = uv \) that satisfies Property \( P \). Note that in all cases, we may assume
\[ f(u) \geq f(v). \tag{4.17} \]
Now construct a new graph $G' = G - e$, and associate with $G'$ a new weight function $f': V(G) \mapsto \mathbb{Z}$ where

$$f'(x) = \begin{cases} f(x) - 1 & \text{if } x = v \\ f(x) & \text{otherwise.} \end{cases} \quad (4.18)$$

By this construction, and Observation 3.3.1, for any arrangement $\pi$ of $G$, where $\pi(u) < \pi(v)$, it is the case that $s_{G',f'}(\pi) = s_{G',f'}(\pi)$.

By the induction hypothesis, $G'$ with any weight function is nested-maximum, so let $\pi'$ be a nested-maximum arrangement of $G', f'$. We claim that $l_{\pi'}(u) \geq l_{\pi'}(v)$. Indeed,

$$l_{\pi'}(u) \geq f'(u) - d(u)_{G'} \geq f'(u) - 1 \geq f'(v) \geq l_{\pi'}(v), \quad (4.19)$$

where the first and last inequalities follow from the definition of weighted level (Equation 3.3), the second because $d(u)_{G'} \leq 1$ as $e$ was deleted, and the third from Equations 4.17 and 4.18. But this means that we may assume that $\pi'(u) < \pi'(v)$ because the vertices of a maximal arrangement are ordered by non-increasing level, and we may permute vertices within a level (Property 3.2.4 and Corollary 3.2.6 respectively).

Now we claim that $\pi'$ is a nested-maximum arrangement of $G$. Suppose not, then there exists an arrangement $\pi''$ so that $\pi''$ beats $\pi'$ on some cut. Since edge $e = uv$ has property $P$, we may assume $\pi''(u) < \pi''(v)$. But by Observation 3.3.1 this means that $s_{G,f}(\pi'') = s_{G',f'}(\pi'')$. Because $\pi'$ is nested-maximum for $G', f'$, it follows that $\pi''$ cannot beat $\pi'$ on any cut. Therefore $\pi'$ is also nested-maximum for $G$.

**Corollary 4.2.4.** Orientations of graphs $G$ where $G$ has maximum degree at most two are nested-maximum.

**Transitive Acyclic Digraphs**

Finally, we show that every transitive acyclic digraph is both nested-minimum and nested-maximum. Recall that a digraph $D = (V, E)$ is transitive when the relation $E$ is transitive. Trivially, transitive acyclic digraphs are nested-minimum because an arrangement where every edge is backward is possible. We show that they are also nested-maximum. Specifically, we show that a nested-maximum arrangement of a transitive acyclic digraph is obtained by ordering the vertices $v$ by non-increasing $d^+(v) - d^-(v)$.

**Theorem 4.2.5.** Let $D = (V, E)$ be a transitive acyclic digraph, and $\pi$ an arrangement of $V(D)$ by non-increasing $d^+(v) - d^-(v)$. Then $\pi$ is nested-maximum.

**Proof.** First we show that every maximal arrangement of $D$ is a topological sort. We proceed by contrapositive. Let $\pi = (v_1, v_2, ..., v_n)$ be an arrangement of $D$ with a backward edge. Choose a backward edge $e = (v_j, v_i)$ that minimizes $j - i$. Then, by transitivity, $v_i$ and $v_j$ must have no neighbour $v_k$ with $i < k < j$, otherwise our choice would not be minimum.
But this means the arrangement $\pi'$ obtained from $\pi$ by interchanging $v_i$ and $v_j$, that is $\pi' = (v_1, ..., v_{i-1}, v_j, v_{i+1}, ..., v_{j-1}, v_i, v_{j+1}, ..., v_n)$, is strictly better than $\pi$. Indeed, cuts $C_i$ to $C_{j-1}$ are strictly increased in $\pi'$ because they include the edge $e$, and all other cuts have the same value in both $\pi$ and $\pi'$. Hence $\pi$ is not maximal, and therefore every maximal arrangement of $D$ is a topological sort.

But this means the level of a vertex $v$ in a maximal arrangement is $l(v) = d^+(v) - d^-(v)$. Since vertices are arranged by non-increasing level in a maximal arrangement, it follows that every maximal arrangement has the same signature. The theorem follows. \qed
Bibliography


Appendix A

Summary of Select Complexity Results

Complexity results for Maximum Cut, Maximum Linear Arrangement, Minimum Linear Arrangement, and their directed counterparts are summarized for select graph classes. In the tables below, P means there exists a polynomial-time algorithm, NP-c means that the problem is NP-complete, and a blank entry means the complexity is, to the author’s knowledge, unknown.

Table A.1: Complexity of MaxCut, MaxLA, MinLA on Select Graphs

<table>
<thead>
<tr>
<th>Graph Class</th>
<th>MaxCut</th>
<th>MaxLA</th>
<th>MinLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planar Graphs</td>
<td>P[20, 33]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bipartite Graphs</td>
<td>P[1]</td>
<td>P[2]</td>
<td>NP-c [12]</td>
</tr>
<tr>
<td>Regular Bipartite Graphs</td>
<td>P[1]</td>
<td>P[2]</td>
<td>NP-c [12]</td>
</tr>
<tr>
<td>Complements of Bipartite Graphs</td>
<td>NP-c [6]</td>
<td>P[2]</td>
<td></td>
</tr>
<tr>
<td>Trees</td>
<td>P[1]</td>
<td>P[4]</td>
<td>P [16, 38, 9]</td>
</tr>
<tr>
<td>Bounded-degree trees</td>
<td>P[1]</td>
<td>P[4]</td>
<td></td>
</tr>
<tr>
<td>Complements of trees</td>
<td>P[3]</td>
<td>P[6]</td>
<td></td>
</tr>
<tr>
<td>Complements of bounded-degree trees</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hypercubes</td>
<td>P[1]</td>
<td>P[4]</td>
<td>P [23]</td>
</tr>
<tr>
<td>Interval Graphs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unit (Proper) Interval Graphs</td>
<td>P[7]</td>
<td></td>
<td>NP-c[10]</td>
</tr>
<tr>
<td>Comparability Graphs</td>
<td>NP-c[40]</td>
<td>NP-c[10]</td>
<td>P[26]</td>
</tr>
</tbody>
</table>

1 Trivial.  2 By Lemma 4.1.3.  3 By the proof of Lemma 2.2.2, because an arrangement maximum for $G$ is minimum for $\overline{G}$.  4 By Section 3.4.  5 Subclass of trees.
6 Subclass of complements of trees.  7 By Theorem 4.1.4.
8 Because MaxCut is NP-complete for this class of graphs, and by Theorem 2.2.4.
9 Superclass of Interval Graphs. 10 By [10] because the complement of an interval graph is a comparability graph [18, 17, Prop. 1.3]. Also by [40] and the proof of Lemma 2.2.2.
Table A.2: Complexity of MaxDiCut, MaxDLA, MinDLA on Select Digraphs

<table>
<thead>
<tr>
<th></th>
<th>MaxDiCut</th>
<th>MaxDLA</th>
<th>MinDLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planar Digraphs</td>
<td>NP-c(^1)</td>
<td>NP-c(^2)</td>
<td></td>
</tr>
<tr>
<td>Eulerian Planar Digraphs</td>
<td>P(^3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orientations of Trees</td>
<td>P(^4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orientations of Trees with Bounded Degree</td>
<td>P(^5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orientations of Hypercubes</td>
<td>P(^3)</td>
<td>P(^3)</td>
<td>P(^3)</td>
</tr>
<tr>
<td>Eulerian Orientations of Hypercubes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strict Split Digraphs</td>
<td>NP-c(^7)</td>
<td>NP-c(^7)</td>
<td>NP-c(^7)</td>
</tr>
<tr>
<td>Split Digraphs</td>
<td>NP-c(^7)</td>
<td>NP-c(^7)</td>
<td>NP-c(^7)</td>
</tr>
</tbody>
</table>

1 By Theorem 2.3.1.  
2 Because MaxDiCut is NP-complete for this class of digraphs, and by Theorem 3.1.1.  
3 MaxCut, MaxDLA, and MinDLA on Eulerian digraphs reduce to the undirected problem.  
4 This is a consequence of Courcelle’s Theorem [11], and is also solved by a simple dynamic programming algorithm.  
5 Subclass of trees.  
6 By Algorithm 3.1.  
7 Because Split Graphs are contained in the intersection of Strict Split Digraphs and Split Digraphs.