On End Vertices of Search Algorithms

by

Jan Gorzny
B.Math., University of Waterloo, 2011
M.Sc., University of Toronto, 2013

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Mathematics and Statistics

On End Vertices of Search Algorithms

by

Jan Gorzny
B.Math., University of Waterloo, 2011
M.Sc., University of Toronto, 2013

Supervisory Committee

Dr. Jing Huang, Supervisor
(Department of Mathematics and Statistics)

Dr. Peter Dukes, Departmental Member
(Department of Mathematics and Statistics)

Supervisory Committee

Dr. Jing Huang, Supervisor
(Department of Mathematics and Statistics)

Dr. Peter Dukes, Departmental Member
(Department of Mathematics and Statistics)

ABSTRACT

Given a graph $G = (V, E)$, a *vertex ordering* of G is a total order $v_1, v_2, \ldots, v_n$ of V. A graph *search algorithm* is a systematic method for visiting each vertex in a graph, naturally producing a vertex ordering of the graph. We explore the problem of determining whether a given vertex in a graph can be the end (last) vertex of a search ordering for various common graph search algorithms when restricted to various graph classes, as well as the related problem of determining if a vertex is an end-vertex when a start vertex is specified for the search. The former is referred to as the *end-vertex* problem, and the latter is the *beginning-end-vertex* problem. For the beginning-end-vertex problem, we show it is NP-complete on bipartite graphs as well as degree restricted bipartite graphs for Lexicographic Breadth First Search, but solvable in polynomial time on split graphs for Breadth First Search. We show that the end-vertex problem is tractable for Lexicographic Breadth First Search on proper interval bigraphs and for Lexicographic Depth First Search on chordal graphs. Further, we show that the problem is NP-complete for Lexicographic Breadth First Search and Depth First Search on bipartite graphs.

Contents

Supervisory Committee **ii**

Abstract **iii**

Table of Contents **iv**

List of Tables **vi**

List of Figures **vii**

List of Algorithms **viii**

Acknowledgements **ix**

1 Introduction **1**

1.1 Background 1

1.2 Notation & Relevant Graph Classes 2

1.3 Outline 6

2 Search Algorithms **8**

2.1 Breadth First Search (BFS) 10

2.2 Depth First Search (DFS) 11

2.3 Lexicographic Breadth First Search (LBFS) 12

2.4 Lexicographic Depth First Search (LDFS) 17

2.5 Maximal Neighborhood Search (MNS) 18

2.6 Relationships 19

3 The Beginning-End-Vertex Problem **21**

3.1 Breadth First Search on Split Graphs 23

3.2 Lexicographic Breadth First Search 25

3.2.1 Bipartite Graphs 25
3.2.2 Bipartite Graphs with Degree at Most 3 30

4 The End-Vertex Problem **32**
4.1 Depth First Search on Balanced Bipartite Graphs 33
4.2 Lexicographic Depth First Search on Chordal Graphs 36
4.3 Lexicographic Breadth First Search 40
4.3.1 Bipartite Graphs 40
4.3.2 Proper Interval Bigraphs 40

5 Future Work **57**

Bibliography **58**

List of Tables

Table 3.1 Complexity results of the beginning-end-vertex problem on various classes of graphs and for various search algorithms. 22

Table 4.1 Complexity results of the end-vertex problem on various classes of graphs and for various search algorithms. 34

List of Figures

Figure 2.1 Interesting search ordering situations. 9
Figure 2.2 A graph and various search order examples on it. 10
Figure 2.3 A graph with an asteroidal triple that has an LBFS order which is an admissible elimination ordering. 14
Figure 2.4 Forbidden subgraphs for HHD-free graphs. 14
Figure 2.5 The relationship between characteristic search ordering properties [13], with respect to triples of vertices $a <_\sigma b <_\sigma c$ along with the additional vertex d for some search σ. 20

Figure 3.1 A split graph: the shaded circle is K; the shaded rectangle is S. 23
Figure 3.2 The switch for the bipartite NP-completeness proof 26
Figure 3.3 The graph G_2. 28

Figure 4.1 DFS search trees 35
Figure 4.2 a can never be the end-vertex of an LDFS, despite the fact that its maximal clique can be a leaf of a clique tree. 36
Figure 4.3 A proper interval bigraph (left) and its intersection model (right). 40
Figure 4.4 Forbidden subgraphs for proper interval bigraphs. 41
Figure 4.5 A graph showing the necessity of an additional requirement, with a cut vertex (left), and without (right). 43
Figure 4.6 An example where the end-vertex does not have maximum eccentricity. 43
Figure 4.7 An illustration of the proof of Lemma 4.3.11. 45
Figure 4.8 A example where $\mathrm{ecc}(v) = \mathrm{diam}(G)$ and v is in a superior deep component of $G - N[w]$. 55

List of Algorithms

1 Generic Search 9
2 Breadth-First Search (BFS) 11
3 Depth-First Search (DFS) 12
4 Lexicographic Breadth Search (LBFS) 13
5 Lexicographic Depth First Search (LDFS) 17
6 Maximal Neighborhood Search (MNS) 19

7 2-Sweep LBFS 55

ACKNOWLEDGEMENTS

I would like to express my thanks to several parties, without whom I could not have completed this degree. First, I would like to thank my supervisor, Dr. Jing Huang, for his guidance and endless patience throughout this degree. I would also like to thank my committee members, Dr. Peter Dukes and Dr. Jørgen Bang-Jensen, for their invaluable comments.

To my friends both local and abroad: thank you. To Christopher van Bommel, Garrett Culous, Stefan Bard, and Samuel Churchill: thank you for working with me at times, and for providing much needed distractions from my work at other times. To my friends outside of Victoria, thank you for your patience and support from afar; it has not gone unnoticed.

To my family: thank you for your endless support. This degree would not have been possible without you.

I would also like to thank the Department of Mathematics and Statistics, as well as the Faculty of Graduate Studies of the University of Victoria for financial support.

Finally, I would like to thanks the rest of the Department of Mathematics and Statistics for all the opportunities provided, and for all of the assistance throughout this degree.

Chapter 1

Introduction

1.1 Background

Various graph classes have been shown to have characteristic vertex orderings. For example, interval graphs, chordal graphs, co-comparability graphs, and distance-hereditary graphs can all be characterized by vertex orderings (see e.g. Brandstädt et al. [4]). The last vertex of such an ordering, called an *end-vertex*, often has nice properties. These properties can be used to design efficient algorithms on these classes of graphs.

A graph *search algorithm* is a systematic method for visiting each vertex in a graph, naturally producing an ordering of the graph's vertices.

As an example, it is well known that the end-vertex of a Lexicographic Breadth First Search (LBFS) on a chordal graph is simplicial (it is contained in a unique maximal clique). This allowed Rose, Tarjan, and Lueker [40] to use LBFS as an efficient recognition algorithm for chordal graphs. In particular, they showed that a graph is chordal if and only if any order generated by LBFS on the graph is a *perfect elimination ordering*.

Corneil, Köhler, and Lanlignel [12] pose the problem of determining if such end-vertices for search algorithms can be identified in polynomial time on graph classes. Such results lead to easy inductive proofs, and new applications for these efficient and well-studied algorithms. The study of this question also leads to the natural question of determining if a particular vertex can be an end-vertex given a starting vertex: this is the related *beginning-end-vertex* problem. Results on this problem often lead to results regarding the end-vertex problem.

Of particular interest is determining when the problem becomes tractable for a given search algorithm. For example, Corneil, Köhler, and Lanlignel [12] show that for weakly chordal graphs, the end-vertex problem for LBFS is NP-complete, but that it can be solved in polynomial time for the more restricted class of interval graphs. Chordal graphs lie directly between these classes, but the problem remains open for this well-studied class.

In this work, we focus our study of the end-vertex problem primarily on the bipartite case for LBFS. We first establish that the problem remains NP-complete for general (and some other restricted) bipartite graphs (or *bigraphs*), and proceed to determine when the problem is tractable. *Proper interval* bigraphs are analogous to proper interval graphs. Interval graphs were shown by to be exactly the graphs that are chordal and do not contain an asteroidal triple (AT-free): a triple of vertices such that between any two vertices, there is a path that avoids the third. Hell and Huang [26] show that proper interval bigraphs are weakly chordal (the bipartite analogue to chordal graphs) and AT-free; for this class of graphs, we show that the problem can be efficiently solved.

LBFS is not the only search algorithm of interest. The common search algorithms Breadth First Search (BFS) and Depth First Search (DFS) can also be subjected to this study in the hopes of finding new applications. Further, Corneil and Krueger [13] recently showed that vertex orderings of various search algorithms have characteristic properties. As a result of these studies, a relatively new search algorithm, Lexicographic Depth First Search (LDFS), was defined, and an older search, Maximal Neighbourhood Search (MNS), was reintroduced to the community. We study each of these algorithms and provide results for all but MNS, although it is useful in our study.

1.2 Notation & Relevant Graph Classes

All the graphs in this thesis are finite and simple; therefore they will not contain multiple edges between vertices nor loops on vertices. For a graph $G = (V, E)$, n will denote $|V|$ and m will denote $|E|$, unless otherwise stated. If $u, v \in V(G)$ and they share an edge e, we say that e is incident with both u and v, and further that u and v are adjacent. Two adjacent vertices will be denoted $u \sim v$ and such a pair are also said to be *neighbours*. We will use $x \nsim y$ to indicate that vertices x and y are not neighbours. The *open* neighbourhood of a vertex v, denoted $N(v)$, is the set of all

neighbours of v and the *closed* neighbourhood of v is denoted $N[v]$ and is equal to $N(v) \cup \{v\}$. For $v \in V(G)$, $d(v)$ denotes the *degree* of v, which is the size of $N(v)$. We will use $\Delta(G)$ to denote $\max_{v \in V(G)} d(v)$ for a graph G. The neighbourhood of a set of vertices S, is $N(S) = (\cup_{v \in S} N(v)) \setminus S$.

The *complement* of a graph G, denoted $\overline{G}$, is the graph with the same vertices as G, in which two vertices are adjacent if and only if they are not adjacent in G.

For $v, u \in V(G)$, $d_G(u, v)$ denotes the *distance* between the two vertices u and v: the number of edges on a shortest path between u and v in G. Where context is clear, we will omit the subscript and simply write $d(u, v)$. For a pair of vertices u and v such that there is no (shortest) path between them, we say that $d(u, v) = \infty$. For a graph G, $\mathrm{diam}(G)$ denotes the *diameter* of a graph, which is defined to be $\max_{u,v \in V(G)} d(u, v)$. Two vertices u, v such that $d(u, v) = \mathrm{diam}(G)$ are said to *achieve the diameter* and we say that (u, v) is a *diametrical pair*. The *eccentricity* of a vertex, denoted $\mathrm{ecc}(v)$ is defined as the longest shortest path from v to any other vertex in the graph, i.e. $\mathrm{ecc}(v) = \max_{u \in V} d(v, u)$. If $d(x, y) = \mathrm{ecc}(x)$ for two vertices x and y, we say that y is an eccentric vertex of x.

If $G = (V, E)$ is a graph and $S \subseteq V$, then $G[S]$ will denote the subgraph of G induced by the set of vertices S, which has all edges $F \subseteq E$ such that both endpoints of an edge $f \in F$ are contained in S. A *component* of a graph G is a maximal subset of vertices $C \subseteq V$ such that between any two vertices $u, v \in C$, there is a path between u and v. If a graph contains exactly one component, it is said to be *connected*.

Let $S \subseteq V$ be a set of vertices in a graph $G = (V, E)$. A set S is a *stable set* or *independent set* if no two vertices in S are adjacent. The complement of S is $\overline{S} = V \setminus S$.

A *walk* is a sequence whose terms are alternately vertices and edges, $W = v_1, e_1, e_2, \ldots, e_{i-1}, v_i$, where each edge e_i is incident with the vertices v_i and v_{i+1}. The elements in a walk are not necessarily distinct. A *path* is a walk such that all vertices are distinct.

Let $P = v_1, v_2, \ldots, v_k$ be a path in G. If $G[P]$ has exactly $k - 1$ edges, then P is called *induced* or *chordless*. A path between vertices u and v may be referred to as a u, v-path. A vertex x *intercepts* (or *hits*) a path P if x is adjacent to at least one vertex on P; otherwise x is said to *miss* (or *avoid*) P. If P is a path, then $V(P)$ denotes the vertices on the path.

A *cycle* is a path whose start and end vertices are the same. A *chord* of a cycle C is an edge between two vertices on C that is itself not contained in C. A *complete*

graph is a graph where every two vertices are adjacent. We will use P_n, C_n, and K_n to denote respectively, the path, (chordless) cycle, and complete graphs on n vertices. A complete graph may also be called a *clique* (on n vertices).

A graph $G = (V, E)$ where $V = X \cup Y$ for disjoint sets X and Y is said to be *bipartite* if each of X and Y is an independent set. The pair (X, Y) is called a bipartition of G. Such a graph may also be called a *bigraph* for short. A well-known theorem of König characterizes bipartite graphs, stating a graph is bipartite if and only if it has no odd cycle (see, e.g. [43], Ch. 1.2).

Given a graph $G = (V, E)$, a *vertex ordering* of G is a total order $v_1, v_2, \ldots, v_n$ of V.

A vertex v is called *universal* to a set S of vertices if v is adjacent to all vertices in S and $v \notin S$, or all adjacent to all vertices in $S \setminus \{v\}$ if $v \in S$. A set $M \subset V$ is a *module* of G if, for all $v \in V \setminus M$, v is adjacent to all of M or to none of them. A module with size greater than one is said to be nontrivial. A *complete* bipartite graph is a bipartite graph where every vertex is universal to the other bipartition.

A vertex v is called *simplicial* if $N(v)$ is a complete subgraph of G, or equivalently if it is not the midpoint of an induced P_3. An ordering $v_1, v_2, \ldots, v_n$ of V is a *perfect elimination ordering* (PEO) if for all i, $1 \leq i \leq n$, v_i is simplicial on the graph induced on $\{v_1, \ldots, v_i\}$.

A graph is said to be *chordal* (or *triangulated*) if every cycle of length strictly greater than 3 posses a chord. A subset $S \subset V$ is a vertex separator for non-adjacent vertices u and v if the removal of S from the graph separates u and v into distinct connected components. Such a set may be refered to as a $u - v$ *separator*, and is called minimal if no proper subset of S is a $u - v$ separator. The following theorem characterizes chordal graphs and is due to the work of Fulkerson and Gross [21] and Dirac [17]:

Theorem 1.2.1. *Let G be an undirected graph. The following statements are equivalent:*

1. *G is chordal;*
2. *G has a perfect elimination ordering. Moreover,* any *simplicial vertex can start a perfect elimination ordering;*
3. *Every minimal vertex separator induces a complete subgraph of G.*

□

Several other characterizations of chordal graphs also exist, see e.g. Gavril [22], Walter [42], or Buneman [5] for a well-known characterization using the intersection graph of subtrees in a tree, Dirac [17] for characterizing chordal graphs as graphs for which every induced subgraph has a simplicial vertex, or Pelsmajer et al. [38] for a characterization in terms of edge sets for a related hypergraph.

A *chordal bipartite* graph is a bipartite graph with no induced cycle of length greater than 4. Chordal bipartite graphs can be characterized by their matrices (see, e.g. Hoffman et al. [27]), or equivalently, as shown by Golumbic and Goss [24], if every non-trivial induced subgraph has an edge xy such that $N[x] \cup N[y]$ induces a complete bipartite subgraph.

A *simple* vertex is a a vertex such that the closed neighbourhoods of its neighbours form a chain under inclusion. A graph is *strongly chordal* if it is chordal and every even cycle of length at least 6 has a *strong chord*, which is a chord joining vertices whose distance along the cycle is odd. Equivalently, a graph is strongly chordal if every induced subgraph has a simple vertex, a characterization which was proved by Farber [20]. Farber also proved that strongly chordal graphs have a characteristic *strong elimination ordering* on their vertices [20].

A graph is *weakly chordal* if it contains no induced cycle of length greater than four. A graph G is a split graph if both G and $\overline{G}$ are chordal, or equivalently if its vertices can be partitioned into a clique and an independent set. A graph is a *strongly chordal split graph* if it is both a strongly chordal graph and a split graph. A graph is a *path graph* (not to be confused with paths within a graph, or the path on n vertices, P_n) if it is the intersection graph of paths in a tree. Path graphs are a special case of the chordal graph intersection model, where all the subtrees are paths (or see e.g. Monma and Wei [36]).

An *asteroidal triple* (AT) is an independent triple of vertices x, y, and z such that between every pair of vertices, there is a path that misses the third. A graph without any asteroidal triple is said to be *AT-free*. Two vertices x, y are unrelated with respect to z if there exists an $x - z$-path P that avoids y and a $y - z$-path Q that avoids x. A vertex x is *admissible* if no pair of vertices is unrelated with respect to x. AT-free graphs allow an *admissible elimination ordering* (AEO): an ordering $v_1, v_2, \ldots, v_n$ such that for every i, $1 \leq i \leq n$, v_i is admissible in $G[v_1, \ldots, v_i]$.

For vertices u, v of a connected AT-free graph, $D(u, v)$ denotes the set of vertices

that intercept all u, v-paths. If $D(u, v) = V(G)$, then (x, y) is said to be a *dominating pair* of G. Thus two vertices u and v are unrelated with respect to a vertex x if $u \notin D(v, x)$ and $v \notin D(u, x)$. A vertex x on an AT-free graph G is said to be *pokable* if the graph obtained by adding a pendant vertex adjacent to x is AT-free. A *pokable dominating pair* is a dominating pair such that both vertices in the pair are pokable. A vertex x is a *pokable dominating pair vertex* if x is pokable and there exists a vertex y such that (x, y) is a dominating pair.

An *interval graph* is a graph for which vertices represent closed intervals on the real number line, and two vertices are adjacent if and only if their respective intervals intersect. Lekkerkerker and Boland proved that interval graphs are exactly the AT-free chordal graphs [33]. A interval graph is said to be *proper* if the family of intervals can be chosen so that no vertex's interval is properly contained in the interval representing another vertex.

A graph is called a *circular arc graph* if it is the intersection graph of a family of arcs on a circle; as with interval graphs, it is called a *proper circular arc graph* if the family can be chosen to be inclusion-free.

If $G = (X \cup Y, E)$, then G is an *interval bigraph* if there is a family F of intervals $I_v, v \in (X \cup Y)$ such that for all $x \in X$ and $y \in Y$, x and y are adjacent in G if and only if I_x and I_y intersect. G is a *proper* interval bigraph if F can be chosen so that no interval contains another.

A graph is a *permutation graph* if it is the intersection graph of lines whose end points are on two parallel lines.

Finally, we describe *lexicographic* ordering. Let $a_1, \ldots, a_s$ and $b_1, \ldots, b_t$ be two sequences of integers. Let i be the largest index such that $i \leq \min\{s, t\}$ and $a_1 = b_1$, $a_2 = b_2$,...,$a_i = b_i$; if $a_1 \neq b_1$, define i to be 0. We write $(a_1, \ldots, a_s) \prec (b_1, \ldots, b_t)$ and say that the sequence $a_1, \ldots, a_s$ is *lexicographically smaller* than $b_1, \ldots, b_t$ if $i < t$ and either $i = s$ or $a_{i+1} < b_{i+1}$. We write $(a_1, \ldots, a_s) \preceq (b_1, \ldots, b_t)$ if either $(a_1, \ldots, a_s) \prec (b_1, \ldots, b_t)$ or the two sequences are identical. For example, $(1, 3, 1) \prec (1, 3, 1, 1) \prec (2, 1) \prec (3)$ where (3) is *lexicographically largest* among those sequences. Note that $\prec$ is a total order of finite integer sequences.

1.3 Outline

In Chapter 2, we define the search algorithms necessary for our study. Each search algorithm presented is accompanied by relevant background results and useful theo-

rems. In Chapter 3, we study the beginning-end-vertex problem, and show that the problem is NP-complete for bipartite graphs (and degree restricted bipartite graphs) for Lexicographic Breadth First Search, but solvable in polynomial time on split graphs for Breadth First Search. In Chapter 4, we study the end-vertex problem. We show that it is NP-complete for Depth First Search on bipartite and balanced bipartite graphs, and that the problem is efficiently solvable for Lexicographic Depth First Search on chordal graphs. Chapter 4 shows that for (degree restricted) bipartite graphs, the problem is NP-complete. Chapter 4 also contains the main result of this thesis: an efficient solution to the end-vertex problem for Lexicographic Breadth First Search on proper interval bigraphs. Chapter 5 concludes the thesis with a brief discussion and some open problems.

Chapter 2

Search Algorithms

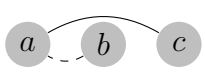
This chapter defines the search algorithms Generic Search, Breadth First Search, Depth First Search, Lexicographic Breadth First Search, Lexicographic Depth First Search, and Maximal Neighbourhood Search, which are used throughout this work and defined in the following sections. The pseudocode for all of these algorithms are adapted from Corneil and Krueger [13]. Additionally, theorems that have been instrumental for proving results on such algorithms will also be discussed in this chapter.

Throughout this thesis, σ will be used to denote a vertex ordering. Given two vertices u, v in a graph, the notation $u <_\sigma v$ indicates that u was visited before v in the ordering σ. If context permits, the subscript σ will be dropped. For a set of vertices X and an ordering σ, σ_X is the restriction of σ to the vertices in X. If σ is a vertex ordering, $\sigma(i)$ denotes the vertex in the ith position of σ.

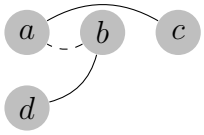
A *graph search* algorithm is a systematic method for visiting all vertices in a graph. After choosing an initial vertex (possibly arbitrarily), a search of a graph consists of following edges from the set of already visited vertices to a new (previously unvisited) vertex. There may be choice involved when deciding which edge to follow to a new vertex, and the methods for deciding which vertex to pick next give rise to some common search algorithms defined in the following sections. All search algorithms are a modification of the *Generic Search* process, described formally in Algorithm 1. Generic Search uses a set S to collect all unvisited nodes, then arbitrarily picks one, visits it (by numbering it), and then adds all of its neighbours that have not yet been visited to the set, and repeats until the set is empty. The generic search idea arose from Tarjan's description [41]. An ordering produced by generic search is called a *Generic Search* order.

Algorithm 1: Generic Search

```
Data: G = (V, E), a start vertex u.
Result: An ordering σ of the vertices of G
1 begin
2     S ← {s};
3     for i ← 1 to |V| do
4         pick and remove an unnumbered vertex v from S;
5         σ(i) ← v; //This assigns v to the number i
6         for each unnumbered vertex w adjacent to v do
7             add w to S;
```



(a) A triple of vertices in a search ordering with $a <_\sigma b <_\sigma c$



(b) A required neighbor $d <_\sigma b$.

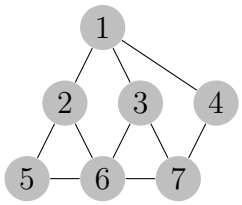
Figure 2.1: Interesting search ordering situations.

Since each search algorithm visits every vertex exactly once, the order in which it does this is naturally a vertex ordering. An important result of formalizing the search procedure is the characterizations of vertex orderings that can be produced by a particular search algorithm. Search ordering characterizations arose from the desire to understand how particular triples of vertices $a <_\sigma b <_\sigma c$, where $a \sim c$ but $a \nsim b$, occur as in Figure 2.1(a) for a search ordering σ. Such triples require the existence of a fourth vertex d, $d \sim b$, which occurs somewhere before b and uses its adjacency to force b to appear in the ordering before c, based on the mechanics of the particular search algorithm.

Since the search algorithms presented traverse a graph by visiting neighbors of vertices already visited, it is not possible that a search will visit a component that does not contain the initial vertex for the search. Disconnected graphs are searched by starting a search on an unvisited component after the entirely visiting another component. Thus we will assume that G is always connected.

Generic search vertex orderings are characterized by the following property, first proposed by Corneil and Krueger [13]:

Property 2.0.1 (Generic Search Ordering Property (Corneil and Krueger [13]))**.** *Given an ordering σ of $G = (V, E)$, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then there*



Generic Search Ordering $1, 4, 7, 2, 3, 6, 5$

BFS Ordering $1, 2, 3, 4, 5, 6, 7$

DFS Ordering $1, 2, 5, 6, 7, 3, 4$

LBFS Ordering $1, 2, 3, 4, 6, 5, 7$

LDFS Ordering $1, 2, 6, 3, 7, 4, 5$

MNS Ordering $1, 2, 3, 6, 5, 7, 4$

Figure 2.2: A graph and various search order examples on it.

exists a vertex d with $d < b$ such that $db \in E$.

Theorem 2.0.1. *For an arbitrary graph G, an ordering σ of V is a Generic Search order of G if and only if it has the Generic Search Ordering Property.* □

Every search studied in this thesis has such a characterization, and the relation amongst all search algorithms considered is summarized in Section 2.6. For brevity, only the characterizing property will be listed.

Charbit, Habib, and Mamcarz showed that any vertex of a graph can be an end-vertex for Generic Search if it is not a cut-vertex, and further that both of the problems studied in this thesis can be solved in linear time for generic search.

Theorem 2.0.2 (Charbit, Habib, and Mamcarz [7])**.** *A vertex t is the end-vertex of some generic search of $G = (V, E)$ if and only if it is not a cut-vertex of G.* □

Theorem 2.0.3 (Charbit, Habib, and Mamcarz [7])**.** *The end-vertex and the beginning-end-vertex problems for generic search can be solved in linear time.* □

An example execution of each search algorithm for an example graph is shown in Figure 2.2.

2.1 Breadth First Search (BFS)

Breadth First Search (BFS), Algorithm 2, modifies generic search by replacing the set S with a first-in-first-out queue. The result is that BFS searches graphs *level-by-level*: every vertex at distance k from the initial vertex is visited before any vertex at distance $k + 1$. BFS orderings are characterized by the following property.

Algorithm 2: Breadth-First Search (BFS)

```
Data: G = (V, E), a start vertex u.
Result: An ordering σ of the vertices of G
1 begin
2     initialize queue to {s};
3     for i ← 1 to |V| do
4         pop v from top of queue;
5         σ(i) ← v; //This assigns v to the number i
6         for each unnumbered vertex w adjacent to v do
7             if w not already in queue then
8                 push w on queue;
```

Property 2.1.1 (BFS Ordering Property (Corneil and Krueger [13]))**.** *Given an ordering σ of $G = (V, E)$, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then there exists a vertex d with $d < a$ such that $db \in E$.*

An interesting corollary to this characterization, used implicitly through this work and applied additionally to LBFS orderings, is the following:

Corollary 2.1.1 (Corneil and Krueger [13])**.** *Let G be a connected graph and σ be an ordering satisfying the BFS ordering property of G. Let $s = \sigma(1)$ and let t be any vertex. Then by repeatedly choosing the lowest labeled neighbour of t in σ until s is reached, we can find a shortest $s - t$-path in G.* □

BFS is well studied and has many applications, including recognition of graph classes, e.g. bipartite graphs (see e.g. Kleinberg and Tardos [30]), graph diameter (see e.g. Cormen, Leiserson, Rivest and Stein [8]), and finding shortest paths in graphs (using Corollary 2.1.1). BFS can be implemented in $O(n + m)$ time; see e.g Cormen, Leiserson, Rivest and Stein [8].

2.2 Depth First Search (DFS)

Depth First Search (DFS), Algorithm 3, modifies generic search by replacing the set S with a first-in-last-out stack. This change results in a search algorithm that travels as far from the initial vertex as is possible, and backtracks only as necessary. Depth first search was first popularized by Tarjan [41]. The following property characterizes DFS orderings.

Algorithm 3: Depth-First Search (DFS)

```
   Data: G = (V, E), a start vertex u.
   Result: An ordering σ of the vertices of G
1  begin
2      initialize stack to {s};
3      for i ← 1 to |V| do
4          pop v from top of stack;
5          σ(i) ← v; //This assigns v to the number i
6          for each unnumbered vertex w adjacent to v do
7              if w already in stack then
8                  remove w from stack;
9              push w on top of stack;
```

Property 2.2.1 (DFS Ordering Property (Corneil and Krueger [13]))**.** *Given an ordering σ of $G = (V, E)$, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then there exists a vertex d with $a < d < b$ such that $db \in E$.*

DFS is also well studied and has applications in graph connectivity (see e.g. Bondy and Murty [2]), planarity testing (Hopcroft and Tarjan [28]) and topological ordering (see e.g. Cormen, Leiserson, Rivest and Stein [8]), among others. Tarjan showed that depth first search can be implemented in $O(n + m)$ time [41].

2.3 Lexicographic Breadth First Search (LBFS)

Lexicographic Breadth First Search (LBFS), Algorithm 4, modifies breadth first search by replacing the queue with a priority queue. A vertex x has a higher priority than another vertex y if it has a neighbour that was visited before any neighbour of y, or if it has more visited neighbours than y. In order to achieve this implementation, it uses *lexicographic* ordering on labels assigned to vertices to break ties: it chooses the vertex with the lexicographically largest label.

LBFS is a layer-search algorithm which has the following characteristic ordering property.

Property 2.3.1 (LBFS Ordering Property (Corneil and Krueger [13]))**.** *Given an ordering σ of $G = (V, E)$, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then there exists a vertex d with $d < a$ such that $db \in E$ and $dc \notin E$.*

Algorithm 4: Lexicographic Breadth Search (LBFS)

```
Data: G = (V, E), a start vertex u.
Result: An ordering σ of the vertices of G
1 begin
2     assign the label ∅ to all vertices;
3     label(u) ← {|V| + 1};
4     for i ← 1 to |V| do
5         pick any unnumbered vertex v with the largest lexicographic label;
6         σ(i) ← v; //This assigns v to the number i
7         for each unnumbered vertex w in N(v) do
8             append (n − i) to label(w);
```

LBFS has been well studied, and many interesting theorems relate to the LBFS orderings. An $O(n+m)$ time implementation of LBFS is described by Golumbic in [23]. Applications of LBFS include graph class recognition, diameter determination, finding dominating pairs in AT-free graphs, and determining properties of powers of graphs. These applications, excluding results relating to graph powers, result in many well-known and useful theorems that are duplicated below as they are useful; for results regarding LBFS and graph powers see Brandstädt, Dragan, and Nicolai [3], Dragan, and Nicola [19], and Chang, Ho, and Ko [6].

Rose, Tarjan and Lueker [40] show that LBFS produces a linear time recognition algorithm for chordal graphs, by first proving that every LBFS end-vertex of a chordal graph is simplicial:

Theorem 2.3.1 (Rose, Tarjan, and Lueker [40])**.** *The end-vertex of an LBFS for a chordal graph is simplicial.* □

Corollary 2.3.2 (Rose, Tarjan, and Lueker [40])**.** *Let G be an arbitrary graph with LBFS σ. G is chordal if and only if σ is a perfect elimination ordering of G.* □

A pair of similar results were shown by Corneil, Olariu and Stewart [15] for AT-free graphs:

Theorem 2.3.3 (Corneil, Olariu and Stewart [15])**.** *The end-vertex of an LBFS of an AT-free graph is admissible.* □

Corollary 2.3.4 (Corneil, Olariu and Stewart [15])**.** *Let G be an AT-free graph with LBFS σ. Then σ is an admissible elimination ordering of G.* □

The converse of Corollary 2.3.4 is not true; there are graphs that contain asteroidal triples and have admissible elimination orderings that can be generated by LBFS. Figure 2.3 provides such an example: note that vertices $\{2, 3, 4\}$ form an asteroidal triple, and that $1, 2, 3, 4, 5, 6, 7$ is both an LBFS ordering and an admissible elimination ordering. However, Corneil and Köhler [11] proved that if all LBFS orderings on a graph G are admissible elimination orderings, then then G is AT-free.

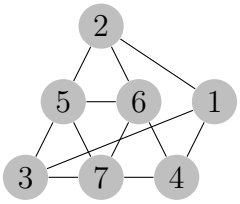
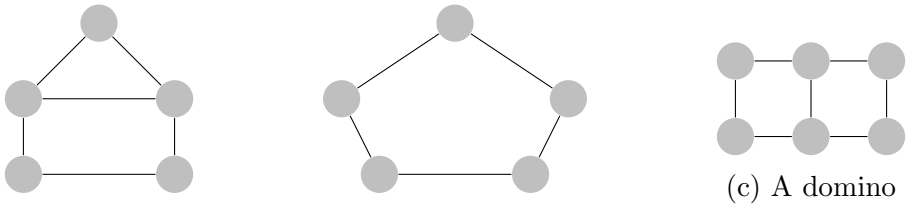


Figure 2.3: A graph with an asteroidal triple that has an LBFS order which is an admissible elimination ordering.

The eccentricity of the vertex visited last in an LBFS execution has also been studied for particular graph classes. A graph is *HHD-free* if it does not contain a House (a C_4 sharing an edge with a K_3, Figure 2.4(a)), Hole (an induced cycle C_k, $k > 4$, Figure 2.4(b)) or Domino (two C_4s sharing an edge, Figure 2.4(c)); a graph is *HH-free* if it contains no Hole or House. Let v be an LBFS end-vertex. Corneil et al. showed that for an interval graph G, $\text{ecc}(v) = \text{diam}(G)$ and $\text{ecc}(v) \geq \text{diam}(G) - 1$ for AT-free graphs [10]. Brandstädt, Dragan, and Nicolai showed chordal graphs have the same bound as AT-free graphs [3]. Dragan also showed that HHD-free graphs respect the $\text{ecc}(v) \geq \text{diam}(G) - 1$ bound and for a HH-free graph G, $\text{ecc}(v) \geq \text{diam}(G) - 2$ [18]. In general, Corneil et al. note that the eccentricity of the last vertex may be arbitrarily far from the graphs diameter [10]. The AT-free result is repeated in the following theorem as it will be useful in Section 4.3.2.



(a) A house (b) A hole ($k = 5$) (c) A domino

Figure 2.4: Forbidden subgraphs for HHD-free graphs.

Theorem 2.3.5 (Corneil, Dragan, Habib, and Paul [10]). *If G is an AT-free graph and v is the vertex visited last by some LBFS ordering σ of G, then $ecc(v) \geq diam(G) - 1$.* □

LBFS has been so well studied that notation and terminology have been defined in order to discuss particular aspects of its execution. In line 5 of Algorithm 4, we call the set of tied vertices a *slice*, which is often denoted S. Given two vertices u and v of an LBFS order σ with $u <_\sigma v$, $\Gamma^\sigma_{u,v}$ denotes the vertex-minimal slice with respect to σ that contains both u and v. A *module of a slice*, or M-slice of a slice S is S itself or any nontrivial module of S. For $u <_\sigma v$, any M-slice of $\Gamma^\sigma_{u,v}$ is referred to as an M-$\Gamma^\sigma_{u,v}$ slice. The following theorems make the preceding notions useful in many proofs.

Theorem 2.3.6 (Corneil, Olariu, and Stewart [16]). *Let σ be an arbitrary LBFS of a graph G. Let t be the first vertex of the connected component containing u of T, an $M - \Gamma^\sigma_{u,v}$. There exists a t,u-path in T all whose vertices, with the possible exception of u, are miseed by v. Moreover, all vertices other than u on this path occur before u in σ. (Such a path is called a* prior path.*)* □

Theorem 2.3.7 (The (Chordal) LBFS Theorem. Corneil, Olariu, and Stewart [16]). *Let G be a chordal graph and let S be an M-slice of an arbitrary LBFS ordering τ of G. Further let σ be an arbitrary LBFS ordering of G. The restriction of σ to S is an LBFS ordering of the graph induced by the vertices of S.* □

Corollary 2.3.8 (Corneil, Olariu, and Stewart [16]). *Let G be a chordal graph and let S be an M-slice of an arbitrary LBFS ordering τ of G. Further let σ be an arbitrary LBFS ordering of G. Then the last vertex of σ_S is an end-vertex of $G[S]$.* □

Partial characterizations of end-vertices for LBFS orders have also been developed. The following combines Theorem 2.3.1 and Theorem 2.3.3 to get a characterization for interval graphs using the characterization of interval graphs as AT-free chordal graphs, which was proved by Lekkerkerker and Boland [33]:

Theorem 2.3.9 (Corneil, Olariu, and Stewart [16]). *A vertex of an interval graph is an end-vertex if and only if it is both simplicial and admissible.* □

The sufficiency of this claim can be extended to arbitrary graphs:

Theorem 2.3.10 (Corneil, Köhler, and Lanlignel [12]). *Let G be an arbitrary graph. If x is a simplicial and admissible vertex of G, then there is an LBFS of G ending at x.* □

Which in turn can be slightly weakened in order to classify an even larger set of end-vertices for an arbitrary graph. A vertex is *semisimplicial* if it is not either of the middle vertices of an induced P_4.

Corollary 2.3.11 (Corneil, Köhler, and Lanlignel [12]). *For an arbitrary graph G, if x is semisimplicial and admissible, then, unless x is a universal vertex, there is an LBFS ending at x.* □

Further, identifying the admissible vertices in an arbitrary graph allows the set of possible end-vertices to be reduced:

Lemma 2.3.12 (Corneil, Köhler, and Lanlignel [12]). *Let G be an arbitrary graph and let x be an admissible vertex. Then x is either an end-vertex or adjacent to an end-vertex.* □

Given an LBFS ordering σ with end-vertex v, its neighborhood can only contain minimal separators that are totally ordered by inclusion:

Theorem 2.3.13 (Berry, Blair, Bordat, and Simonet [1]). *For any graph G (chordal or not), any LBFS ordering σ of G with end-vertex x, the minimal separators of $N(x)$ are totally ordered by inclusion.* □

Starting LBFS with an admissible vertex in an AT-free graph G provides a quick algorithm for finding a dominating pair in G:

Theorem 2.3.14 (Corneil, Olariu, and Stewart [15]). *Let $G = (V, E)$ be a connected AT-free graph and suppose that G contain no vertices unrelated with respect to vertex x of G. Consider the vertex ordering produced by an LBFS σ starting at x. Then, for all vertices $u, v \in V$ with $u <_\sigma v$, we have $u \in D(v, x)$.* □

For interval graphs, an end-vertex can always be visited last by starting at a suitable, different end-vertex of the graph. A similar result for proper interval bigraphs is proved in Section 4.3.2.

Lemma 2.3.15 (The Flipping Lemma (for Interval Graphs). Corneil, Olariu, and Stewart [16]). *Let G be an interval graph and let y and z be end-vertices of G. If there is an LBFS of G that starts at y and ends at z, then some LBFS of G starts at z and ends at y.* □

Lemma 2.3.16 (Corneil, Köhler, and Lanlignel [12]). *For an interval graph G with X being the set of end-vertices, for every $x \in X$, there is another vertex $y \in X$ and an LBFS starting at y that ends at x.* □

However, in general this result cannot be extended. The last result can be interpreted as follows: let $X_0 = V$ and for $i > 0$ define X_i to be

$$\{x \mid \exists y \in X_{i-1} \text{ and LBFS } \sigma \text{ that starts at } y \text{ and ends at } x\}.$$

Eventually there must be an integer k such that $X_k = X_{k-1}$. The last result shows that $k = 2$ for interval graphs, and the next shows that it can be arbitrarily large for the larger class of cocomparability graphs. A graph G is a *cocomparability* graph if there is a transitive orientation of the edges of the complement of G.

Theorem 2.3.17 (Corneil, Köhler and Lanlignel [12]). *For any constant c, there is a cocomparability graph with closure at least c.* □

2.4 Lexicographic Depth First Search (LDFS)

Lexicographic Depth First Search (LDFS), Algorithm 5, modifies depth first search by choosing neighbors by again favoring vertices with as many most recently visited vertices as possible. LDFS is less studied than the other algorithms described in this section, but LDFS orders have the following characteristic property.

Property 2.4.1 (LDFS Ordering Property (Corneil and Krueger [13])). *Given an*

Algorithm 5: Lexicographic Depth First Search (LDFS)

```
   Data: G = (V, E), a start vertex u.
   Result: An ordering σ of the vertices of G
1  begin
2      assign the label ∅ to all vertices;
3      label(u) ← {0};
4      for i ← 1 to |V| do
5          pick any unnumbered vertex v with the largest lexicographic label;
6          σ(i) ← v; //This assigns v to the number i
7          for each unnumbered vertex w in N(v) do
8              prepend i to label(w);
```

ordering σ of $G = (V, E)$, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then there exists a vertex d with $a < d < b$ such that $db \in E$ and $dc \notin E$.

Unlike the search algorithms described prior to this one, little is known about LDFS, and there are few known applications of LDFS. A necessary condition for end-vertices of LDFS was shown by Xu, Li, and Liang [44]. A *maximal clique module* is a set of vertices A if A is both a clique and a module and A is inclusion-maximal for both properties. A *moplex* is both a clique X and a module such that $N(X)$ is a minimal separator (a set S such that there is some pair of vertices a, b such that S is a minimal $a - b$ separator).

Theorem 2.4.1 (Xu , Li, and Liang [44])**.** *Let G be a graph with vertex set V of size n, σ an ordering of V generated by the LDFS algorithm, and X the maximal clique module containing $\sigma(n)$. Then X is a moplex of G whose vertices are numbered consecutively by σ.* □

Unlike the other search algorithms, LDFS does not have a $O(n + m)$ time implementation in general; rather Krueger [32] shows that LDFS has a worst case complexity of $O(\min\{n^2, n + m \log n\})$. However, for cocomparability graphs, Köhler and Mouatadid [31] show that it is possible to generate LDFS orderings in linear time.

LDFS also has fewer applications than the other search algorithms, although Mertzios and Corneil [35] use it to solve the longest path problem on cocomparability graphs.

2.5 Maximal Neighborhood Search (MNS)

Maximal Neighbourhood Search (MNS), Algorithm 6, is a generalization of both LBFS and LDFS, where priority is assigned by taking the next vertex whose label is simply maximal in size. Thus it is easy to see that ties occur with MNS more often than with either LBFS or LDFS, as the lexicographic ordering is no longer relevant. MNS orderings have the following characteristic property:

Property 2.5.1 (MNS Ordering Property (Corneil and Krueger [13]))**.** *Given an ordering σ of $G = (V, E)$, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then there exists a vertex d with $d < b$ such that $db \in E$ and $dc \notin E$.*

Algorithm 6: Maximal Neighborhood Search (MNS)

```
Data: G = (V, E), a start vertex u.
Result: An ordering σ of the vertices of G
1 begin
2     assign the label ∅ to all vertices;
3     label(u) ← {|V| + 1};
4     for i ← 1 to |V| do
5         pick any unnumbered vertex v with maximal label;
6         σ(i) ← v; //This assigns v to the number i
7         for each unnumbered vertex w in N(v) do
8             add i to label(w);
```

For chordal graphs, a characterization of MNS end-vertices is presented as the next theorem. A *substar* of a vertex x is the set of vertices of $N(C)$ where C is a connected component of $G - N[x]$.

Theorem 2.5.1 (Berry, Blair, Bordat and Simonet [1])**.** *For any chordal graph G and any vertex x of G, x is an MNS end-vertex if and only if x is simplicial and the substars of x are totally ordered by inclusion.* □

Maximal neighbourhood search can be used to recognize graph classes, including chordal graphs, graphs in $O(n + m)$ time; see Li and Wu [34].

2.6 Relationships

As mentioned at the beginning of this chapter, the search algorithms presented are refinements of existing algorithms, stemming from Generic Search. Moreover, this implies that results about a particular search algorithm may be applicable to their refined versions. In particular, this is true for the characterizing property of each search algorithm, as seen in Figure 2.5. For example, it shows that every LBFS search ordering is an MNS search ordering, and so any LBFS ordering must respect the properties of any MNS search ordering. This fact will be exploited in later proofs.

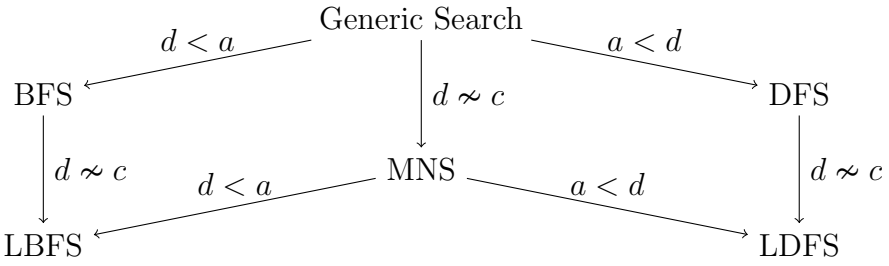


Figure 2.5: The relationship between characteristic search ordering properties [13], with respect to triples of vertices $a <_\sigma b <_\sigma c$ along with the additional vertex d for some search σ.

Chapter 3

The Beginning-End-Vertex Problem

The Beginning-End-Vertex Problem asks whether it is possible start a search at a particular vertex and end at another vertex.

Beginning-End-Vertex Problem for a search S:
Input: A graph $G = (V, E)$, and 2 vertices s and t.
Question: Is there an S-ordering σ of V such that $\sigma(1) = s$ and $\sigma(n) = t$?

Table 3.1 shows the complexity of solving the beginning-end-vertex problem for various classes of graphs and various search algorithms; a "?" indicates that no complexity results are known. The table shows that there are few results about the problem, especially when compared with the same table (Table 4.1) for the better-studied end-vertex problem. The majority of the results on the beginning-end-vertex problem are due to Charbit, Habib, and Mamcarz [7] and are focused on BFS, with one result for general graphs and LBFS proved by Corneil, Köhler, and Lanlignel [12].

Although the problem appears difficult in most of the studied cases, our results on split graphs for BFS shows that for particular graph classes and search algorithms, the problem may be tractable. It is also interesting to determine where this boundary lies for each search algorithm. The remainder of this chapter is dedicated to showing our three contributions to the table.

Beginning-End-Vertex Results	BFS	LBFS	DFS	LDFS	MNS
All graphs	NPC (Includes bipartite)	NPC [12]	?	?	?
Bipartite	NPC [7]	**NPC Sec. 3.2.1**	?	?	?
Bipartite ($\Delta \leq 3$)	NPC [7]	**NPC Sec. 3.2.2**	?	?	?
Balanced Bipartite	?	?	?	?	?
Proper Interval Bigraph	?	?	?	?	?
Weakly Chordal	NPC [7]	?	?	?	?
Chordal	?	?	?	?	?
Interval	?	?	?	?	?
Split	**P Sec. 3.1**	?	?	?	?
Str. Chordal Split	**P (Contained in split)**	?	?	?	?
Path graphs	?	?	?	?	?

Table 3.1: Complexity results of the beginning-end-vertex problem on various classes of graphs and for various search algorithms.

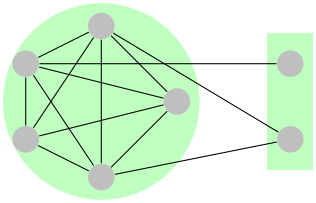


Figure 3.1: A split graph: the shaded circle is K; the shaded rectangle is S.

3.1 Breadth First Search on Split Graphs

Recall that a graph is a split graph if its vertices can be partitioned into a clique K and a stable set S. In this section we prove the following result by first characterizing the conditions for which there is a BFS execution from a start vertex s to an end-vertex t. An example split graph is shown in Figure 3.1.

Theorem 3.1.1. *The beginning-end-vertex problem for BFS is solvable in polynomial time on split graphs.*

Our characterization relies on the end-vertex characterization for BFS proved by Charbit, Habib, and Mamcarz, which requires the following definition. Let $G = (V, E)$ be a graph. For any $x \in V$ we denote by D_x the set of vertices y such that $N(y) \subsetneq N(x)$.

Theorem 3.1.2 (Charbit, Habib, and Mamcarz [7])**.** *Let $G = (V, E)$ be a graph, and let $t \in V$ be any vertex. A necessary condition for t to be the last vertex of some BFS-ordering σ of G is that there exists a neighbour x of t such that $D_t \subseteq N(x)$. If G is a split graph, then this condition is also sufficient.* □

In short, the statement says that all of the vertices that are dominated by an end-vertex t are all adjacent to some common vertex x. We characterize the conditions required for a BFS ordering from s to t in the next theorem. Note that this is more specific, as we care about the particular start vertex.

Theorem 3.1.3. *Let $G = (V, E)$ where $V = S \cup K$ for some stable set S and clique K. Let $s, t \in V$ where t satisfies the requirements of Theorem 3.1.2.*

- *If $s \in K$, $t \in K$, there exists a BFS ordering from s to t if and only if s is a universal vertex.*
- *If $s \in K$, $t \in S$, there exists a BFS ordering from s to t if and only if t $d(s, t) = 2$ or s is a universal vertex.*

- *If $s \in S$, $t \in K$, there exists a BFS ordering from s to t if and only if there is no vertex $u \in S$ such that $d(s,u) = 3$.*

- *If $s \in S$, $t \in S$, there exists a BFS ordering from s to t if and only if there is no vertex $u \in S$ such that $d(s,u) = 3$, unless $d(s,t) = 3$.*

Proof. We prove each case separately.

- $s \in K$, $t \in K$. First, suppose that s is universal. Then we can start BFS at s and since all vertices are tied (as they are all adjacent to s), we simply pick t last. Conversely, if s is not universal, there is some vertex w with $d(s,w) = 2$. But since $s \in K$ and $t \in K$, $d(s,t) = 1$. Thus, any BFS starting at s has to visit t before w; i.e. no BFS starting at s can end at t if s is not universal.

- $s \in K$, $t \in S$. First, suppose s is universal or $d(s,t) = 2$. If s is universal, then we can again pick all vertices after visiting s, making sure to pick t last. Alternatively, suppose $d(s,t) = 2$. Since K is a complete subgraph, when we start BFS at s, all vertices of K (and possibly some of S) are added to the queue after visiting s. By Theorem 3.1.2, if t is to be an end-vertex, we can find an x as described in the theorem. Then we can visit the vertices of the graph in the following order to get a legitimate BFS search: $s, (K \cap \overline{N[t]}) \setminus \{s,x\}, x, N(t), S \setminus (\{t\} \cup N(x)), (N(x) \cap S) \setminus \{t\}, t$. Note that $s, (K \cap \overline{N[t]}) \setminus \{s,x\}, x, N(t)$ are all contained within K, and remainder of the order is contained within S; further, the vertices visited in S are visited in an order such that neighbours of vertices visited earlier in K are visited first. Conversely, suppose that s is not universal and $d(s,t) \neq 2$. Since s is not universal, there is some vertex w with $d(s,w) \geq 2$. Since K is a clique, $w \in S$. Now if $d(s,t) = 1$, then there is no BFS starting at s ending at t: t would have to be picked before w. If $d(s,t) > 1$, then really $d(s,t) = 2$ since t is adjacent to some vertex $v \in K \cap N(s)$, so $d(s,t) = 1$ since $d(s,t) \neq 2$. Thus no BFS starting at s can end at t.

- $s \in S$, $t \in K$. Suppose t satisfies the end-vertex characterization and no u with $d(s,u) = 3$ exists. By the latter assumption, every vertex of S has a common neighbour $x \in K$, otherwise some vertex is distance 3 from s. Then the following is a legitimate BFS ordering of vertices: $s, x, N(x) \setminus \{t\}, t$. Conversely, suppose there exists a u such that $d(s,u) = 3$. Since $d(s,t) \leq 2$ (as $t \in K$), no BFS can end at t.

- $s \in S$, $t \in S$. Suppose t satisfies the end-vertex characterization and there is no vertex $u \in S$ such that $d(s,u) = 3$, unless $d(s,t) = 3$. Let $S' = \{s' | s' \in S \text{ and } d(s,s') = 2\}$. If $d(s,t) = 3$ then the following is a legitimate BFS ordering: $s, N(s), K \cap \overline{N[t]}, S', K \cap N(t), S \setminus \{s,t\}, t$, making sure to pick t last if there are any vertices $s' \in S$ with $N(s') = N(t)$. This order visits all of $N(s)$ after s, then everything in K not adjacent to t, then anything in S at distance 2 from s before finishing K, then S ending with t last. If $d(s,t) = 2$, then the following is a legitimate BFS ordering: $s, N(s) \cap K, K \setminus N(s), S \setminus \{s,t\}, t$, making sure to pick t last if there are any vertices $s' \in S$ with $N(s') = N(t)$. This order visits all of $N(s)$ after s, then everything else in K, then anything in S at distance 2 from s before finishing S, ending with t last. To conclude, we note that $d(s,t)$ cannot be greater than 3 since K is complete, and it cannot be 1 since S is a stable set. Conversely, suppose there exists a u such that $d(s,u) = 3$ and $d(s,t) = 2$. Since $d(s,t) = 2$, no BFS can end at t.

□

The main result can now be proved:

Proof of Theorem 3.1.1. It is straightforward to find the clique and stable set partitions of a split graph in linear time due to Heggernes and Kratsch [25]. Checking the distance between all pairs of vertices in a graph can be done in polynomial time, checking if a vertex is universal can be done in linear time, and Charbit, Habib, and Mamcarz [7] showed that checking if a vertex meets the characterization for BFS end-vertices on split graphs can also be done in linear time. □

3.2 Lexicographic Breadth First Search

3.2.1 Bipartite Graphs

In this section, we show the beginning-end-vertex problem for LBFS on bipartite graphs is NP-complete. To do so, we will use a reduction from 3-SAT (see e.g. Karp [29]). A variable x or its negation $\overline{x}$ is a *literal*, and a disjunction of literals is called a *disjunctive clause*. A *formula* is a conjunction or disjunction of clauses, and a formula is *conjunctive normal form* if it is a conjunction of disjunctive clauses. The reduction will use the definition of a special graph G_n defined below. Following that definition, we can observe some properties necessary to complete the reduction.

Boolean 3-Satisfiability (3-SAT)
Input: A boolean formula f in conjunctive normal form with three literals per clause.
Question: Is f satisfiable?

Theorem 3.2.1. *The beginning-end-vertex problem for LBFS is NP-complete on bipartite graphs.*

For every $n \in \mathbb{N}$, we define a graph G_n, which has one special vertex r_n called the root. It is constructed recursively as follows:

- G_0 is the graph with one vertex r_0.
- G_n is constructed from G_{n-1} by first adding a copy of the graph in Figure 3.2, adding a bridge between r_{n-1} and c_n. Finally, we attach a path of length $4n-3$ to y_n (respectively $\overline{y_n}$), and label its end-vertex x_n (respectively $\overline{x_n}$).

The construction of G_n relies on the gadget shown in Figure 3.2, which is what allows LBFS executions to choose to visit the left (y_n) or right ($\overline{y_n}$) side of the graph before the center vertex (c_n), but not both. For example, G_2 is shown in Figure 3.3. Note that the graph is clearly bipartite: odd levels form one bipartition, and even levels form the other. The following proposition shows that the reduction can be performed in polynomial time, as there are a polynomial number of vertices in G_n. We require one technical lemma to prove the recurrence relation.

Lemma 3.2.2. *The recurrence relation* $|G_{n+1}| = |G_n| + 8n + 12$ *with initial condition* $|G_0| = 1$*, has closed form* $|G_n| = 4n^2 + 8n + 1$.

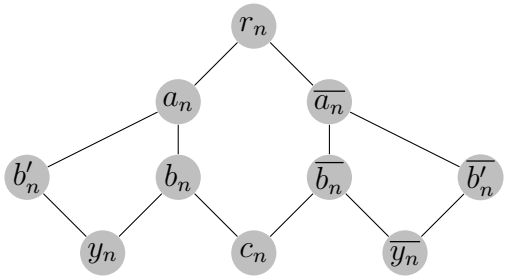


Figure 3.2: The switch for the bipartite NP-completeness proof

Proof. We prove this by induction. For $n+1 = 1$, $|G_1| = |G_0|+8(0)+12 = 1+12 = 13$, which agrees with the stated formula. So assume the result holds for all $n+1 \leq n'$ for some n' and consider $n'+1$. We have

$$|G_{n'+1}| = |G_{n'}| + 8n' + 12$$

which, applying the induction hypothesis to $|G_{n'}|$, is

$$= (4n'^2 + 8n' + 1) + 8n' + 12$$

$$= 4n'^2 + 16n' + 12 + 1$$

$$= 4(n'^2 + 4n' + 3) + 1$$

$$= 4(n'+1)(n'+3) + 1$$

$$= 4(n'+1)((n'+1)+2) + 1$$

$$= 4((n'+1)^2 + 2(n'+1)) + 1$$

$$= 4(n'+1)^2 + 8(n'+1) + 1$$

which is what we wanted. □

Proposition 3.2.3. *G_n is a bipartite graph that has $4n^2+8n+1$ vertices that are at distance at most $4n$ from r_n. There are $2n+1$ vertices at distance exactly $4n$ from r_n, and these are $x_1, \overline{x_1}, x_2, \overline{x_2}, \ldots, x_n, \overline{x_n}$ and r_0.*

Proof. We first show the size of the graph. G_0 has 1 vertex, so the formula holds in the base case. Assume the result holds for all G_n and consider G_{n+1}. Note that G_{n+1} has all the vertices of G_n, plus a copy of the graph in Figure 3.2, which has 10 vertices, plus two pendant paths of length $4(n+1)-3$, so we have

$$|G_{n+1}| = |G_n| + 10 + 2(4(n+1) - 3) = |G_n| + 8n + 12$$

with initial condition $|G_0| = 1$, which has the desired closed form solution by Lemma 3.2.2:

$$|G_n| = 4n^2 + 8n + 1$$

The second part of the proposition is immediate from the definition. □

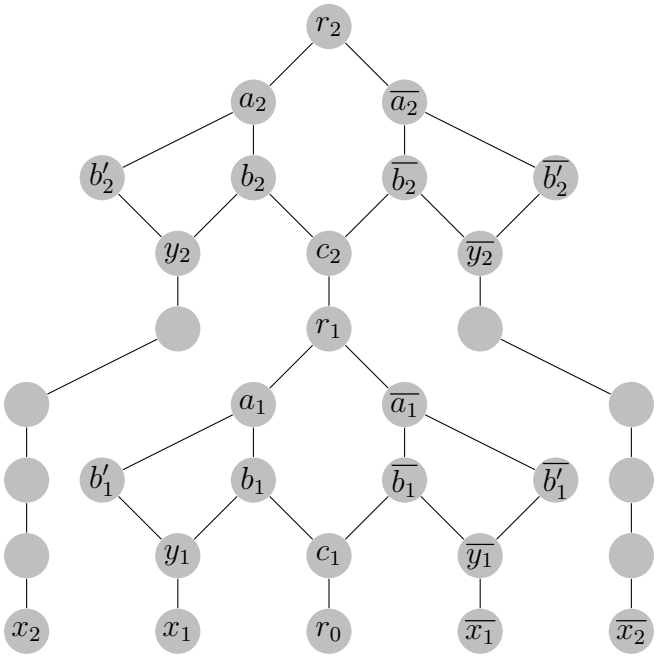


Figure 3.3: The graph G_2.

Proposition 3.2.4. *Consider an order on the vertices of G_n given by an execution of LBFS starting at r_n. For each $1 \leq i \leq n$ exactly one of x_i and $\overline{x_i}$ is before r_0. Moreover, each of the 2^n choices of one among x_i and $\overline{x_i}$ for each i, can be obtained as the set of vertices that appear before r_0 for some LBFS order of G_n.*

Proof. We prove this by induction on n; if $n = 0$ there is nothing to prove. Assume the result holds for all G_{n-1} and consider G_n. Starting at r_n, the LBFS either continues with a_n or $\overline{a_n}$. On the next layer, the vertices at distance two from r_n, there are b'_n, b_n, $\overline{b_n}$, and $\overline{b'_n}$. Assume without loss of generality a_n is visited before $\overline{a_n}$, so that the two vertices are numbered i and $i-1$, respectively. Now both of b'_n and b_n must be visited before the $\overline{b'_n}$ and $\overline{b_n}$ as $i > i-1$ as the labels on these vertices are respectively $(i), (i), (i-1), (i-1)$ (recall that the value LBFS assigns decreases after visiting a vertex) so b'_n and b_n have lexicographically largest labels. Assume without loss of generality that b'_n is visited before b_n and $\overline{b'_n}$ was visited before $\overline{b_n}$. On the next layer, the vertices at distance three from r_n, there are y_n, c_n, and $\overline{y_n}$ with labels respectively $(i-2, i-3), (i-3, i-5), (i-4, i-5)$. y_n must be visited first: it has the earliest neighbour (b'_n), and the next earliest neighbor (b_n), one of which (b'_n) must be missing from c_n, so c_n does not have priority (note that in fact, $(i-3, i-5) \prec (i-2, i-3)$). Further, c_n must be chosen second on this level, because it has an earlier neighbour than $\overline{y_n}$ (b_n). Thus in the next layer, we must have the neighbors of y_n (call this one y'_n), c_n, and $\overline{y_n}$ (call it $\overline{y'_n}$) appear in the order that these vertices were chosen; thus r_{n-1} can never be chosen last in this layer. Moreover, y'_n (or similarly $\overline{y'_n}$) is chosen after r_{n-1} if and only if on the last layer of the graph, x_n is after any of the descendants of r_{n-1}, in particular r_0. So it is true for $i = n$. By choosing to take a_n or $\overline{a_n}$, we can dictate which of y_n or $\overline{y_n}$ is visited before r_n, and thus which of x_n or $\overline{x_n}$ appears before r_0. To conclude, we see that the order of the middle $2n-1$ vertices on the last layer, namely those which are a descendant of r_{n-1} are determined by an LBFS on G_{n-1}, so by induction we are done. □

We are now ready to prove the main result.

Proof of Theorem 3.2.1. The proof uses a reduction from 3-SAT. Given $F = (x_1, \ldots, x_n, c'_1, \ldots, c'_m)$, an instance of 3-$SAT$ where x_i $(1 \leq i \leq n)$ are boolean variables and c'_j $(1 \leq j \leq m)$ are clauses, we construct a new graph G_F. G_F is constructed by taking G_n (as defined above) along with $m+1$ new vertices: $c'_1, \ldots, c'_m$ and t. Each c'_i vertex corresponds to a clause of the same name in F and has the neighbours in G_n exactly the vertices x_i or $\overline{x_i}$ that appear in it; t is adjacent to only r_0. Note that

since $d(r_n, x_i) = d(r_n, \overline{x_i}) = d(r_n, r_0) = 4n$, we have that $d(c'_i, r_n) = d(t, r_n) = 4n + 1$, and these are the only vertices at this distance from r_n.

We claim that there exists an LBFS execution on G_F starting at r_n and ending at t if and only if F is satisfiable.

For any LBFS starting at r_n, we associate a truth value to each variable $x_i \in F$ thanks to Proposition 3.2.4 as follows: we know that exactly one of x_i and $\overline{x_i}$ is before r_0 in this order, and we pick that literal to be true. Now since t is only adjacent to r_0, a vertex c'_i in the last layer will be before t in the order if and only if one of its neighbors (i.e. a literal appearing in it), was chosen before r_0 on the previous layer. So c'_i is before r_0 if and only if the associated clause is true with the assignment described above, which proves the only if direction.

Conversely, if F is satisfiable, then the second part of Proposition 3.2.4 allows us to get an LBFS that sorts all true literals before r_0 among the vertices as distance $4n$. Then the previous observation on the clauses implies that t must be the last vertex.

This concludes the proof. □

3.2.2 Bipartite Graphs with Degree at Most 3

The LBFS beginning-end-vertex problem remains NP-complete on bipartite graphs with degree at most 3. The proof is similar to the last section, but requires more care when adding clause vertices (note that all others in G_n have degree at most 3).

Theorem 3.2.5. *The beginning-end-vertex problem for LBFS is NP-complete on bipartite graphs with degree at most 3.*

Proof. Let F, an instance of 3-SAT be defined as in the proof of Theorem 3.2.1, and let $c(x_i)$ be the number of clauses in which x_i appears. Construct G'_F as follows: start with G'_F defined before, and to each vertex x_i of the construction, we attach a tree $T(x_i)$ such that: x_i is the root of the tree, $T(x_i)$ contains exactly $c(x_i)$ leaves that are all at distance $\lceil \log m \rceil$ from x_i, x_i has at most 2 children, and no vertex of $T(x_i)$ has degree bigger than 3.

For every clause vertex c_j such that c_j contains the literal x', x'', x''', we make the vertex that corresponds to c_j adjacent to one leaf of $T(x')$, one of $T(x'')$, and one of $T(x''')$, in such a way that all the leaves of the $T(x_i)$ have degree 2, and we attach a path of length $\lceil log(m) \rceil + 1$ to r_0, and we call the other end of this path t.

Now we can conclude the proof by showing that there exists an LBFS execution on G_F starting at r_n and ending at t if and only if F is satisfiable, which is similar to the proof of Theorem 3.2.1. □

Chapter 4

The End-Vertex Problem

The End-Vertex Problem asks whether a particular vertex of a graph can be visited last for some execution of a search. The problem does not specify where a search would be started, or what choices (or what ties were broken) would be made during execution, only that there are some choices and a start vertex that would lead the vertex to be numbered last. Proving results about the end-vertices of a search algorithm is useful for identifying particular graph classes (recall e.g., Theorem 2.3.1), and allows for easy inductive proofs of their correctness.

End-Vertex Problem for a search S:
Input: A graph $G = (V, E)$, and a vertex t.
Question: Is there an S-ordering σ of V such that $\sigma(n) = t$?

Table 4.1 shows the complexity of solving the end-vertex problem for various classes of graphs and various search algorithms. Again, a "?" indicates that no complexity results are known. Significantly more work has been done on this problem than on the beginning-end-vertex problem. Again, most results are due to Charbit, Habib, and Mamcarz [7], and Corneil, Köhler, and Lanlignel [12] prove several results for LBFS.

As with the beginning-end-vertex problem, research has focused on determining for the complexity of this problem for various graph classes. Of particular interest are the classes related to chordal graphs for LBFS: interval graphs are solvable in polynomial time, but weakly chordal graphs are NP-complete. For chordal graphs, the problem remains open. We focus instead on determining the boundary of this problem for LBFS when restricted to bipartite graphs. We show it is NP-complete for bipartite

graphs and degree-restricted bipartite graphs in Sections 4.3.1 and 4.3.1 respectively, and that it is polynomially solvable for proper interval bigraphs in Section 4.3.2. Additionally, we show that the problem is NP-complete for DFS on balanced bipartite graphs in Section 4.1, but can be solved efficiently for LDFS on chordal graphs in Section 4.2.

4.1 Depth First Search on Balanced Bipartite Graphs

We first consider for balanced bipartite graphs, which are connected bipartite graphs with $|X| = |Y|$ where (X, Y) is the bipartition. We show that the end-vertex problem is NP-complete for DFS on this class of graphs. Our proof uses ideas presented in [7]. The proof uses a reduction from the Hamiltonian Path Problem for bipartite graphs, which is NP-complete even for the smaller class of chordal bipartite graphs, as shown by Müller [37]. A *Hamiltonian path* is a path in a graph G that visits every vertex of G.

Hamiltonian Path (for Bipartite Graphs)
Input: A bipartite graph G.
Decide: Does G have a Hamiltonian path?

A *depth-first search tree* (DFS-tree) is a spanning tree T of a graph G formed by the execution of DFS on G. The vertex added to T at each stage which is one which is adjacent to a most recent addition to the tree as possible, which is expected given that DFS uses a stack. An example is provided in Figure 4.1(a). By redrawing a tree, it is always possible to hang the tree in such a way that the last vertex visited by DFS is the rightmost leaf in the tree (not including the root): when executing DFS, simply add a vertex to T as the rightmost child of its parent. Figure 4.1(b) shows a DFS-tree where the last vertex visited is the rightmost leaf of the tree.

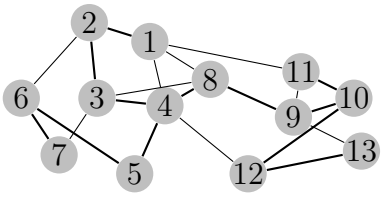
We are now ready to state and prove the main result of this section.

Theorem 4.1.1. *The end-vertex problem for DFS is NP-complete on balanced bipartite graphs.*

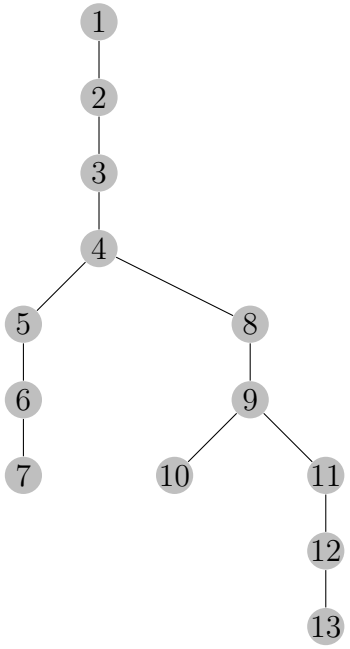
Proof. The proof uses reduction from the Hamiltonian Path Problem for bipartite graphs. Given a graph $G = (X \cup Y, E)$, we build G' an instance of the DFS end-vertex problem by adding a vertex u which is universal to X. We claim that G admits

End Vertex results	BFS	LBFS	DFS	LDFS	MNS
All graphs	NPC [7]	NPC [12]	NPC [7]	NPC [7]	?
Bipartite	NPC [7]	**NPC Sec. 4.3.1**	**NPC (Contains balanced bipartite)**	?	?
Bipartite ($\Delta \leq 4$)	NPC [7]	**NPC Sec. 4.3.1**	?	?	?
Balanced Bipartite	?	?	**NPC Sec. 4.1**	?	?
Proper Interval Bigraph	?	**P Sec. 4.3.2**	?	?	?
Weakly Chordal	NPC [7]	NPC [12]	NPC [7]	NPC [7]	?
Chordal	?	?	NPC [7]	**P Sec. 4.2**	P [1]
Interval	?	P [12]	?	**P (Contained in chordal)**	P (Contained in chordal)
Split	P [7]	P [7]	NPC [7]	P [7]	P (see [7])
Str. Chordal Split	P [7]	P [7]	NPC [7]	P [7]	P (see [7])
Path graphs	?	?	NPC [7]	**P (Contained in chordal)**	P (see [7])

Table 4.1: Complexity results of the end-vertex problem on various classes of graphs and for various search algorithms.



(a) A DFS search tree, shown in bold.



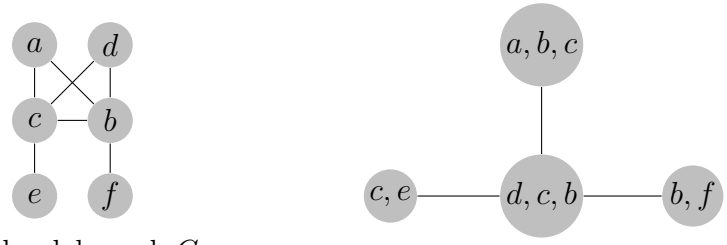
(b) The same DFS-tree, drawn so that vertex 13 is the rightmost leaf.

Figure 4.1: DFS search trees

a Hamiltonian path if and only if there exists a DFS of G' ending at u.

If G admits a Hamiltonian path $p = v_1, \ldots, v_n$, then there exists a DFS-tree of G that is a path. If $v_n \in X$ then the path $v_1 \ldots v_n, u$ is still a Hamiltonian path; if $v_n \in Y$ then $v_1 \in X$ because the graph is balanced. The path $v_n, \ldots, v_1, u$ is a DFS order of G' ending at u, which proves the if direction of the claim above.

For the other direction, assume there is a DFS order σ of G' with $\sigma(n) = u$. Since u is universal to X, we can really think of u as being included in Y. Since u is last in the DFS run, u is the rightmost leaf. Note that the leftmost leaf l is in Y. Let r_p be u's parent in the search tree. Let l_p be leftmost leaf's parent in the tree. Because the leftmost leaf is in Y, l_p is in X, so l_p is adjacent to u. If $l_p \neq r_p$, then after l was visited, l_p should visit all of its neighbours, including u; but this did not happen, so we have a contradiction. If $l_p = r_p$, then we must have that there is a (proper) path from the root to l, because we must have that all leaves (except the root) have the same parent, otherwise G is not balanced, or u was not rightmost. Since u is in Y, we must have that the root of the tree was in X, so the path from the root to l, along with an edge from the root to u (which is universal to X), forms a Hamiltonian path. □



(a) A chordal graph G.

(b) A clique tree of G.

Figure 4.2: a can never be the end-vertex of an LDFS, despite the fact that its maximal clique can be a leaf of a clique tree.

Corollary 4.1.2. *The end-vertex problem for DFS on bipartite graphs is NP-complete.*

□

4.2 Lexicographic Depth First Search on Chordal Graphs

We characterize end-vertices of LDFS on chordal graphs, and show that the complexity of determining if a vertex meets the characterization is solvable in polynomial time.

Recall that every chordal graph is the intersection graph of subtrees in a tree. This intersection model gives rise to a *clique tree* representation of a chordal graph G: any tree T whose vertices are the maximal cliques of G and such that for every two maximal cliques C, C', each clique on the path from C to C' in T contains $C \cap C'$. An example is shown in Figure 4.2. Recall further that LDFS is a refinement of MNS, and thus by Theorem 2.5.1, every end-vertex must be simplicial.

Given that DFS end-vertices are always leaves of some DFS search tree, a natural question to ask was whether every simplicial vertex in a leaf of a clique tree for a chordal graph can be an LDFS-end. Unfortunately, this is not the case: Figure 4.2(a) shows an example G, with corresponding clique tree in Figure 4.2(b); the vertex a is never an end-vertex for LDFS. This is unsurprising: vertex a fails to satisfy the substar inclusion property required by Theorem 2.5.1.

By Theorem 2.4.1, an LDFS end-vertex must be in a moplex, but some moplexes do not contain an LDFS end-vertex; see Figure 4.2(a): $\{a\}$ is a moplex, but cannot end any LDFS run.

However, translating Theorem 2.5.1 into the language of moplexes provides a

characterization of LDFS end-vertices that we require. This shows that the MNS end-vertex characterization is both necessary and sufficient for LDFS as well on chordal graphs.

Theorem 4.2.1. *Let G be a chordal graph, and $v \in V(G)$ be a simplicial vertex and $K_v = N[v]$. Then v is an LDFS end-vertex if and only if there exists M_v, a moplex containing v, such that if $C_1, \ldots, C_k$ are the components of $G - N(M_v)$, then for all $1 \leq i \leq k$, there exists a vertex $c_i \in C_i$ such that $N(M_v)$ contains a minimal $v - c_i$-separato,, and further, that these minimal vertex separators are totally ordered by inclusion.*

We require the following technical lemmas. The first was first proved by Rose [39]. The second was used without proof by Charbit, Habib, and Mamcarz [7]. We give a proof of it here for completeness.

Lemma 4.2.2 (Rose [39])**.** *Let $G = (V, E)$ be a chordal graph, and let C induce a connected subgraph such that $N(C)$ is a clique. Then there exists $z \in C$ such that $N(C) \subseteq N(z)$.* □

Lemma 4.2.3 (Charbit, Habib, and Mamcarz [7])**.** *After visiting a vertex $x \in V(G)$, LDFS will visit a maximal (with respect to inclusion) clique of G that contains x, unless x is contained in exactly one maximal clique and all other vertices of the clique have already been visited.*

Proof. Let K' be a clique containing x such that K' has the lexicographically largest label possible on any unvisited vertex other than x in it; if all cliques are tied, pick any one. After visiting x, LDFS must pick the neighbor y with lexicographically largest label of x; $y \in K'$ since LDFS must pick the largest label, and by choice of K', K' contains it. If y completes the traversal of a clique, we are done, so assume it does not; in particular, $|K'| > 2$. Now for all unvisited $y' \in N(x)$, $\text{label}(y') \preceq \text{label}(y)$. Consider the next vertex. All $z \in N(y)$ have leading lexicographic number $\sigma(y) = \sigma(x)+1$. All $z \in N(x) \cap N(y)$ have second lexicographic digit as $\sigma(x)$. All $z \in N(y) \setminus N(x)$ have second lexicographical digit at most $\sigma(x) - 1$. Thus, LDFS must choose a common neighbour z of x and y; namely, a vertex in K'. Now we can look at the labels of $z' \in N(z)$; the largest lexicographically digit is $\sigma(z) = \sigma(x) + 2$. two largest are $\sigma(y)$ and $\sigma(x)$ if $z' \in K'$; otherwise, it is missing at least one of these, so the vertices in K' will be preferred. We can repeat this until all vertices in K' are visited. □

The proof of Theorem 4.2.1 is broken up into two lemmas.

Lemma 4.2.4. *Let G be a chordal graph, and $v \in V(G)$ be a simplicial vertex. If v is an LDFS end-vertex then there exists M_v, a moplex containing v, such that if $C_1, \ldots, C_k$ are the components of $G - N(M_v)$, then for all $1 \leq i \leq k$, there exists a vertex $c_i \in C_i$ such that $N(M_v)$ contains a minimal $v - c_i$-separator, and further, that these minimal vertex separators are totally ordered by inclusion.*

Proof. Assume we have an LDFS ordering σ where v is the last vertex in the ordering. Note that every LDFS order is a valid MNS order. By Theorem 2.5.1, the substars of v are totally ordered by inclusion. Let $M_v = \{v\} \cup Y$ where $Y = \{y | N[y] = N[v]\}$. Let $C_1, \ldots, C_\alpha$ be the components of $G - N[v]$. Let $c_i \in C_i$ be the vertex z as defined by Lemma 4.2.2 for each component C_i. Then M_v is clearly a clique, a module, and the separators of v and each vertex c_i for each $1 \leq i \leq \alpha$ are also exactly the minimal separators contained in the substars of v. □

Lemma 4.2.5. *Let G be a chordal graph, and $v \in V(G)$ be a simplicial vertex and $K_v = N[v]$. If there exists M_v, a moplex containing v, such that if $C_1, \ldots, C_k$ are the components of $G - N(M_v)$, and for all $1 \leq i \leq k$, there exists a vertex $c_i \in C_i$ such that $N(M_v)$ contains a minimal $v - c_i$-separator, and further, that these minimal vertex separators are totally ordered by inclusion, then v is an LDFS end-vertex.*

Proof. Let denote S_i the minimal v, c_i-separator for all i. Consider S_i such that $|S_i|$ is minimal; start LDFS at any vertex in S_i. Since M_v is the moplex containing v, and M_v is a module, everything else in the graph is either adjacent to everything in M_v or none of it; in particular, anything adjacent to everything in M_v is adjacent to v; therefore, $N(M_v)$ is contained in K_v since v is simplicial. Thus S_i is entirely contained in K_v for all i. LDFS can visit the entire set S_i by Lemma 4.2.3 since it is a clique (as it is a minimum separator for v and any vertex in C_i; minimum separators in chordal graphs are cliques), and all labels will be tied until S_i is completely visited. After visiting the last vertex in S_i, all vetices in $K_v \setminus S_i$, as well as all vertices in C_i that are separated by this set, are tied. Therefore, LDFS can choose a vertex in C_i adjacent to all of S_i, which exists by Lemma 4.2.2. Now since C_i is connected, LDFS must continue to traverse this component until it is entirely visited (the other possible vertices are those in S_i, but those are visited already). Then LDFS must jump back to continue traversing the graph; it must do so on the vertex with the largest label that still has unvisited neighbours. Since C_i is entirely visited, the last vertex LDFS (the

last vertex visited would have the highest label) visited with unvisited neighbours is the last vertex it visited in S_i. Now if S_i is a separator for another component $C_{i'}$, LDFS can visit this component next; this can be repeated until the components separated by S_i have all been visited. Now the last vertex in S_i still has an unvisited neighbour: a vertex in S_j where S_j is the next larger separator; traverse all such neighbours in $S_j \setminus S_i$. Now LDFS can choose C_j, and this process can be repeated until S_k is maximal. Then the only remaining unvisited vertices are in K_v; they are also all tied for the largest label. Visit them in any order such that v is last. □

Proof of Theorem 4.2.1. This follows from lemmas 4.2.4 and 4.2.5. □

Corollary 4.2.6. *Let G be a chodal graph, and $v \in V(G)$. The following are equivalent:*

1. *v is an MNS end-vertex;*
2. *v is an LDFS end-vertex;*
3. *There exists M_v, a moplex containing v, such that if $C_1, \ldots, C_k$ are the components of $G - N(M_v)$, then then for all $1 \leq i \leq k$, there exists a vertex $c_i \in C_i$ such that $N(M_v)$ contains a minimal $v - c_i$-separator, and further, that these minimal vertex separators are totally ordered by inclusion.*

□

Using this characterization, we can conclude the following:

Theorem 4.2.7. *The end-vertex problem for LDFS is solvable in polynomial time on chordal graphs.*

Proof sketch. Given a chordal graph G, we can check that a vertex satisfies the characterization in polynomial time. Checking if a vertex is simplicial can be done in $O(n^2)$ at worst. Checking that the inclusion property is satisfied can be done by first identifying M_v and then deleting $N(M_v)$ and checking whether or not the neighbourhoods of the components have a total ordering (e.g. by topological sort). □

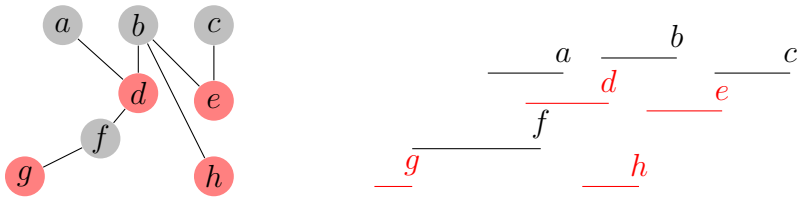


Figure 4.3: A proper interval bigraph (left) and its intersection model (right).

4.3 Lexicographic Breadth First Search

4.3.1 Bipartite Graphs

We now show that for arbitrary bipartite graphs, the LBFS end-vertex problem is NP-complete. The proof uses a reduction from the beginning-end-vertex problem on bipartite graphs, which is shown to be NP-complete in Theorem 3.2.1. The reduction is applicable for arbitrary graph classes for which adding a pendant does not change class membership. It is applicable to both BFS and LBFS.

Theorem 4.3.1. *For (L)BFS, the beginning-end-vertex problem reduces to the end-vertex problem.*

Proof. Given $G = (V, E), s, t \in V$, an instance of the beginning-end-vertex problem for (L)BFS, we build G', an instance of the end-vertex problem by adding a pendant path P of length $\text{diam}(G) + 1$ to s. Now every (L)BFS that ends in t will have to begin at some vertex of P, and will visit s as the first vertex of G. □

Corollary 4.3.2. *Given a bipartite graph G and a vertex v of G, it is NP-complete to decide if there exists an execution of LBFS on G such that v is the end-vertex.*

Proof. This follows from Theorem 4.3.1 and Theorem 3.2.1. □

Corollary 4.3.3. *Given a bipartite graph with $\Delta(G) \leq 4$, G and a vertex v of G, it is NP-complete to decide if there exists an execution of LBFS on G such that v is the end-vertex.*

Proof. This follows from Theorem 4.3.1 and Theorem 3.2.5. □

4.3.2 Proper Interval Bigraphs

Recall that a bipartite graph H, with a fixed bipartition (X, Y) is an interval bigraph if the vertices of H can be represented by a family of intervals $I_v, v \in X \cup Y$, so that,

for $x \in X$ and $y \in Y$, x and y are adjacent in H if and only if I_x and I_y intersect. An example of a proper interval bigraph is shown in Figure 4.3. The following theorem (cf. Hell and Huang [26]) provides an important characterization of proper interval bigraphs.

Theorem 4.3.4. *The following statements are equivalent, for a bipartite graph H:*

1. *H is a proper interval bigraph;*
2. *H is AT-free;*
3. *H does not contain an induced cycle of length at least six or any graph in Figure 4.4.*

□

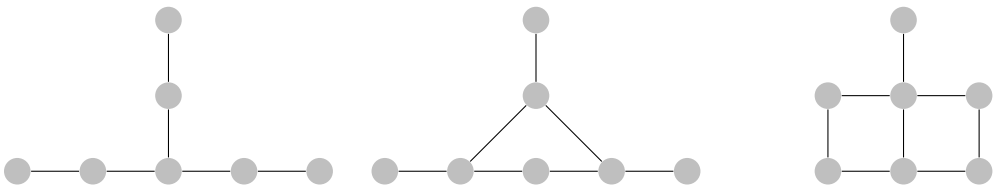


Figure 4.4: Forbidden subgraphs for proper interval bigraphs.

According to the above theorem, proper interval bigraphs are AT-free graphs. The following theorems by Corneil et al. are useful our study of end-vertices on proper interval bigraphs.

Theorem 4.3.5 (Corneil, Olariu, and Stewart [15])**.** *Let G be a connected AT-free graph. Then a vertex is admissible if and only if it is a pokable dominating pair vertex of G.* □

Theorem 4.3.6 (Corneil, Olariu, and Stewart [14])**.** *In every connected AT-free graph some dominating pair achieves the diameter.* □

We also use the following proposition, which is similar to Lemma 4.2.2.

Proposition 4.3.7. *Let G be a proper interval bigraph, and let C induce a connected subgraph of $G - N[w]$ for $w \in V(G)$. Then there exists $z \in C$ such that $N(C) \subseteq N(z)$.*

Proof. To see this, let $x, y \in C_1$ such that $N(x) \cap N(C_1)$ and $N(y) \cap N(C_1)$ are incomparable; if no such pair exists, take the vertex $u \in C_1$ such that $N(u) \cap N(C_1)$ is maximal and we are done. Let $x' \in (N(x) \cap N(C_1)) \setminus N(y)$ and $y' \in (N(y) \cap N(C_1)) \setminus N(x)$. Since both x and y are in C_1, they are connected via a path P that does not intersect $N(C_1) \cap N(w)$; P has length at least 2 since they are at the same distance from w. Note that P is not adjacent to x' or y' otherwise these vertices would be closer to w. Now $d(x, x') = d(x', w) = 1$ and similarly $d(y, y') = d(y', w) = 1$, so $w - y' - y - P - x - x' - w$ is a cycle of length at least 6, contradicting Theorem 4.3.4. □

We call such a z a *superior* vertex of C.

Let $w \in V(G)$ be a vertex. Let C be an arbitrary component of $G - N[w]$. We say that C is a *deep* component of $G - N[w]$ if there exists a vertex $w' \in C$ such that $d(w, w') = \text{ecc}(w)$. We say that C is a *superior* deep component of $G - N[w]$ if there exists deep component C' such that $N(C) \supsetneq N(C')$.

Proposition 4.3.8. *If σ is an LBFS of a proper interval bigraph, and $v = \sigma(n)$, then there does not exist $u \in V(G)$ such that $N(u) \subsetneq N(v)$.*

Proof. Suppose to the contrary that such a u exists. Then when LBFS choses between u and v, since u will have a strictly smaller neighbourhood than v, so v should have been chosen before u. This contradicts v being the end-vertex of σ. □

Our main result is the following characterization for end-vertices of LBFS on proper interval bigraphs:

Theorem 4.3.9. *Let $v \in V(G)$ for a proper interval bigraph G. Then there is some LBFS ordering σ such that $\sigma(n) = v$ if and only if*

1. *$ecc(v) \geq diam(G) - 1$*
2. *v is not a cut vertex*
3. *v is admissible*
4. *there does not exist $u \in V(G)$ such that $N(u) \subsetneq N(v)$*
5. *there exists a vertex w such that v is an eccentric vertex of w, and when $ecc(v) = diam(G) - 1$, the component of $G - N[w]$ containing v is not a superior deep component.*

Consider the graphs in Figure 4.5. In each, the vertex labeled v satisfies all of the conditions except for the last. However, no LBFS ordering ends at either of these vertices. Thus the last condition is necessary.

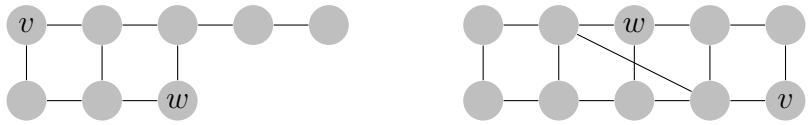


Figure 4.5: A graph showing the necessity of an additional requirement, with a cut vertex (left), and without (right).

Because proper interval bigraphs are analogous to interval graphs, it is natural to ask if the end-vertex of LBFS for proper interval bigraphs must have maximum eccentricity, like those of interval graphs (as shown by Corneil, Dragan, Habib, and Paul [10]). We first asked this for the restricted case when a proper interval bigraph has no cut vertices; but unfortunately, this is not true, as seen in Figure 4.6, and thus not true for proper interval bigraphs in general.

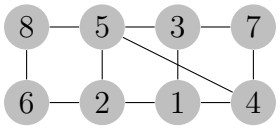


Figure 4.6: An example where the end-vertex does not have maximum eccentricity.

Before proving the theorem, we state and prove two technical lemmas.

Lemma 4.3.10. *Let G be a proper interval bigraph and w such that $ecc(w) > 2$. Suppose some eccentric vertex of w is admissible in G. Then $G - N[w]$ contains at most 2 deep components.*

Proof. Let v be such an admissible eccentric vertex of w. Suppose to the contrary that there are at least 3 deep components in $G - N[w]$. Assume without loss of generality that $a \in C_2$ and $b \in C_3$ for vertices a, b with $d(w, a) = d(w, b) = \text{ecc}(w)$. Look at shortest $w - a$, $w - b$, and $w - v$-paths, call them A, B, and V, respectively. Each of these paths have length at least 3, so $N(a) \cap B = \emptyset$ and $N(b) \cap A = \emptyset$. But then $A \cup V$ avoids b and $B \cup V$ avoids a, so a, b are unrelated with respect to v, contradicting that v is admissible. □

Lemma 4.3.11. *Let G be a proper interval bigraph. Let v with $ecc(v) = diam(G) - 1$ satisfy the conditions of Theorem 4.3.9. If (u', v') are a diametrical dominating pair*

with $d(v, v') \leq d(v, u')$, then v and v' are adjacent. Moreover w (as in condition 5 of Theorem 4.3.9) is contained in a shortest $u' - v'$-path.

Proof. Note first that $u' \neq v$ and $v' \neq v$, as otherwise $\text{ecc}(v) = \text{diam}(G)$. Denote $\text{diam}(G)$ by k.

Suppose to the contrary that v and v' are not adjacent (i.e. $d(v', v) \geq 2$). First, we show that P, an arbitrary shortest $u' - v'$-path, cannot contain v; suppose to the contrary $v \in P$. Since v' was closer than u' (or tied), we have $d(u', v) \geq 2$. Then u' and v' are unrelated with respect to v, a contradiction. So $v \notin P$; in fact, v is not on any shortest path P. Since (u', v') are a dominating pair, $N(v) \cap P \neq \emptyset$. Let $x \in N(v) \cap P$ closet to v'. Now $\text{ecc}(v) = k - 1$, so $d(v, u') \leq k - 1$. If x is adjacent to v', then $d(x, u') = k - 1$ via any shortest $u' - v'$-path. Further, if $d(u', v) < k - 1$, then we can find a shorter $u' - v'$-path using v and x, a contradiction. So we must have that $d(u', v) = k - 1$. But now we have an odd walk, $u' - x' - v - u'$, as it has length $(k - 1) + (k - 1) + 1 = 2(k - 1) + 1$. This contradicts the fact that the graph is bipartite.

For the second part of the lemma, consider a shortest $v' - w$-path R. Then $v \notin R$ as otherwise $d(w, v') > d(w, v)$, contradicting v being an eccentric vertex of w. Note that every $u' - v'$-path dominates G because (u', v') are a dominating pair. Consider a shortest such path Q that contains v. At least one such Q exists because otherwise the only way from u' to v is through v', which means that $\text{ecc}(u') = d(u', v') + 1 = k + 1$, a contradiction. If u' gets to v via a path T of distance $k - 2$, then $T \cup \{v\}$ is a path of length $k - 1$ from u' to v', also a contradiction. We will show that there exists a different $u' - v'$-shortest path that contains w.

If w and v' are adjacent, then v' is adjacent to both v and w and consequently that $d(w, v) = 2$. Since v is an eccentric vertex of w, we must also have that $d(w, u') \leq 2$. If $d(w, u') = 1$, then since w is adjacent to both v' and u', $d(v', u') = d(v, w) = 2 = k$, a contradiction to $\text{ecc}(v) < k$. If $d(w, u') = 2$, Now w must be adjacent to something on Q, as otherwise w and u' are unrelated with respect to v as $d(w, v) = 2$ via v' and $d(w, u') = 2$ via Q (note that $v \notin Q$ as otherwise $d(u', v') = 1$, also a contradiction). This contradicts v being admissible. So we can always find a $q \in N(w) \cap Q$, and we can take Q from u' to q, then q to w, and w to v' via R.

Now suppose w and v' are not adjacent. First note that $R \cap Q = \{v'\}$. To see this, note that if this is not the case, let $x \in (R \cap Q) \setminus \{v\}$ be a vertex closest to w. Then because v is before v' on Q, $d(x, v) < d(x, v')$; if not, taking Q to x and then the shorter path v' is shorter than taking Q to x and then taking the path from x to

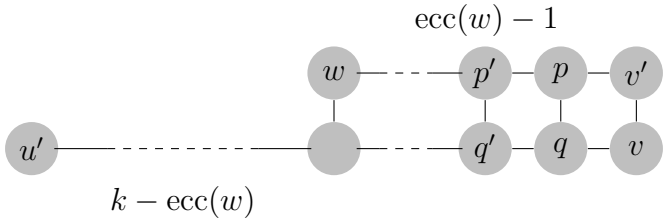


Figure 4.7: An illustration of the proof of Lemma 4.3.11.

v' containing v, a contradiction. This implies $d(w, v) < d(w, v')$, a contradiction to v being an eccentric vertex of w. Note also that $d(w, v') = \text{ecc}(w) - 1$, since by the first part of the lemma v and v' are adjacent: any shorter and we would have a $w - v$-path with distance less than $\text{ecc}(w)$. We claim $d(u', v) = k - 1$, indeed, if $d(u', v) < k - 1$ there exists a $u' - v'$-path shorter than length k, contradicting $d(u', v') = k$, and if $d(u', v) > k - 1$, $\text{ecc}(v) > k - 1$, a contradiction.

By the last claim (and by choice of Q), at least one such cycle contains v', and a vertex $q \in N(v) \setminus \{v'\}$ which is also on Q as every two vertices at distance d from w must have a common neighbour at distance $d - 1$ from w. Since (u', v') are a dominating pair, we must have that Q dominates each vertex in R. Thus p, the neighbor of v' on R must be adjacent to q: if it is adjacent to v, the graph contains a 3-cycle, if it is adjacent to a vertex farther away from v on Q then Q was not a shortest path (or if it's one further than q, there is a 3-cycle). So $p - q - v - v' - p$ form a 4-cycle.

Let p' be the neighbor of p in R that is not v' (i.e. of distance 2 vertex from v' on R) and let q' be the neighbor of q on Q that is not v (i.e. of distance 2 vertex from v on Q). Now p' must be adjacent to q': if it is adjacent to q, the graph contains a 3-cycle, if it is adjacent to a vertex farther away from q on Q then Q was not a shortest path (or if it's one further than q, there is a 3-cycle), and it cannot be adjacent to v otherwise $d(w, v') > d(w, v)$, contradicting v being an eccentric vertex of w.

We can repeat this argument until $p' = w$ with some q', and then use the path $u' - q' - w' - v'$. Note that $d(q', v) = k' - 1$ for some k', which is also $d(w, v')$, so this path has distance $k - (k' - 1 + 1) + 1 + (k' - 1) = k - k' + 1 + k' - 1 = k$, which is $d(u', v')$. □

The construction used in the last proof is illustrated in Figure 4.7.

The remaining lemmas in this section are used to modularize the proof of our characterization.

Lemma 4.3.12. *Let G be a proper interval bigraph and $v \in V(G)$. If v satisfies all the conditions of Theorem 4.3.9, and (w, v) is a dominating pair of G, where v is an eccentric vertex of w, then there exists an LBFS ordering σ of G that ends at v that starts at w.*

Proof. We provide a direct construction. Start an LBFS at w; v must be at distance $k' = \text{ecc}(w)$, from w. Let $L_{k'}$ be set of all vertices at distance k' from w. It suffices to show that LBFS can visit every vertex of $L_{k'} \setminus \{v\}$ before v. If this set is empty, we are done, so assume it is not. Since v satisfies condition 4 of Theorem 4.3.9, there is no vertex with a strictly smaller neighborhood than v. So let $u \in L_{k'} \setminus \{v\}$. Since u is hit by any $w - v$-path, it follows that $N(u) \cap N(v) \neq \emptyset$. If $N(v) \subseteq N(u)$, then anytime LBFS could visit v, LBFS could visit u instead. The only alternative is if there is a u is such that $N(u)$ and $N(v)$ are incomparable. But then any $w - v$-path fails to dominate the vertex in $N(u) \setminus N(v)$, a contradiction. Thus v is an LBFS end-vertex. □

The rest of the proof will require the following definition. Given a vertex y and an LBFS ordering σ, an $a-b$-path is said to be *y-majorizing* if all the vertices on the path are were labeled before y in σ (that is, all vertices on the path were numbered with a larger number than y so that for all vertices u on the path, $u <_\sigma v$).

Lemma 4.3.13. *Let G be a proper interval bigraph and $v \in V(G)$. If v satisfies all the conditions of Theorem 4.3.9, and $ecc(v) = diam(G) - 1$, then there exists an LBFS ordering σ of G that ends at v.*

Proof. The proof consists of a series of claims. In view of Lemma 4.3.12, we may assume (w, v) is not a dominating pair. This implies in particular that $\text{ecc}(w) > 1$.

Let $\text{ecc}(w) = k'$ and $\text{diam}(G) = k$. Let (u', v') be a diametrical dominating pair, which exists by Theorem 4.3.6. Without loss of generality, $d(v', v) \leq d(u', v)$. By Lemma 4.3.11, v is adjacent to v'. Let $L^w_{k'}$ be the set of vertices at distance k' from w. If $L^w_{k'} = \{v\}$, then we are done (any LBFS starting at w ends at v). So assume $L^w_{k'}$ contains a vertex $u(\neq v)$.

Let $C_1, \ldots, C_\alpha, C_{\alpha+1}, \ldots, C_\beta$ be the components of $G - N[w]$, where C_i, $1 \leq i \leq \alpha$, are deep components, and C_j, $\alpha + 1 \leq j \leq \beta$, are not deep components. Assume without loss of generality that $v \in C_1$. Let c_i be the superior vertex for a component C_i. Let S consist of the vertices such that w is their only neighbour. We will show that v can be an LBFS end-vertex. Consider the following order of sets of vertices

(order the vertices within each set arbitrarily). If a vertex in a set has already been visited because it also belongs to an earlier ordered set, skip it.

$$\{w\}, S,$$

$$(N(w) \cap N(C_2)) \setminus (N(w) \cap N(C_1)), \ldots, (N(w) \cap N(C_\alpha)) \setminus (N(w) \cap N(C_1)),$$

$$(N(w) \cap N(C_{\alpha+1})) \setminus (N(w) \cap N(C_1)), \ldots, (N(w) \cap N(C_\beta)) \setminus (N(w) \cap N(C_1)),$$

$$N(w) \cap N(C_1)$$

Claim 1. *The above order is a valid LBFS prefix.*

Proof. Note that since G is bipartite it has no odd cycles, and thus no pair of vertices in $N(w)$ are adjacent: $N(w)$ is an independent set. Thus $N(w)$ can be ordered in any order, and since this labels all the vertices of $N(w)$ exactly once, this is a valid LBFS prefix. □

This labels all the vertices at distance 1 from w. At distance 2 (from w), each c_i will be visited first in each C_i, and otherwise proceed as necessary according to LBFS.

Claim 2. *If $ecc(w) = 2$, the prefix described above results in v being the last vertex for some LBFS with this prefix.*

Proof. Note that every deep component must be a single vertex, otherwise there are two vertices at distance 2 with an edge between them, resulting in an odd cycle. Thus, every other deep component C_i $1 < i \leq \alpha$ is exactly c_i. Each c_i has at least one vertex that was ordered before any vertex in $N(C_1) = N(v)$, and so those other components have a higher priority, or they are tied with v, and we can still pick them before v. □

Claim 3. *Unless $ecc(w) > 2$, then there are most two deep components in $G - N[w]$.*

Proof. This is Lemma 4.3.10. □

As a result of the last two claims, we will now assume $\text{ecc}(w) \geq 3$ and $\alpha = 2$.

Claim 4. *$u' \in C_2$, and $d(u', w) = k'$.*

Proof. We claim first that $u' \in C_2$. Suppose to the contrary that $u' \in C_1$. Since v and v' are adjacent they are both in C_1, there is a shortest path between u' and v' entirely contained in C_1. Any such path would fail to dominate any vertex in C_2. So we must have that $u' \in C_2$.

Now we show $d(u', w) = k'$. Suppose to the contrary that $d(u', w) < k'$. So $d(u', w) = k' - 1$ otherwise the eccentric vertices of w in C_2 would be missed by a $u' - v'$-path that takes a shortest $u' - w$-path, and then a shortest $w - v'$-path. Note that we must also have $d(w, v') = \text{ecc}(w) - 1$, by Lemma 4.3.11. So if $d(u', w) = k' - 1$ and $d(v', w) = k' - 1$, $k \leq 2(k' - 1)$ since there is a path $u' - w - v'$ of at least this length. Since v is farther from w than v', $k \leq 2(k' - 1) + 1 = 2k' - 1$.

Consider an eccentric vertex u of w in C_2 (at least one exists since C_2 is a deep component). Now $d(u, v) = k - 2$: if it is shorter, then we can find a u', v'-path of length at most $k - 3 + 2 = k - 1$, contradicting that they achieve the diameter; if $d(u, v) = k - 1$, then since $d(u', v) = k - 1$, and u' is adjacent to u, we have an odd cycle $v - u' - u - v$. It cannot be larger than $k - 1$, as $\text{ecc}(v) \leq k - 1$. The shortest possible paths between u and v, which must have length $k - 2$, occurs when there is u, v-path that does not use w, and contains a vertex in $N(w)$. Thus $k' - 1 + k' - 1 = 2k' - 2 \leq k - 2$. This implies $2k' \leq k$.

Combining these results gives $2k' \leq k \leq 2k' - 1$. □

Claim 5. *There exists a v-majorizing $u' - w$-path, or there exists a w' such that there exists a v-majorizing $u' - w'$-path.*

Proof. Let P be any shortest $u' - w$-path. If $c_2 \in P$, then we are done: amongst all vertices in L_2^w, c_2 (any superior vertex in C_2) has the highest priority (due the the above order, or is tied, and we can still proceed), so it must be picked first. Then $N(c_2) \cap P \cap L_3^w$ must be chosen before any vertex in C_1. Since P is a shortest path and $d(u', w) = k'$, P contains exactly one vertex at every distance from w. Further, the first vertices visited at each distance i from w are those adjacent to some vertex on P at distance $i - 1$ from w, eventually $L_{k'}^w \cap C_2$ is visited before any vertex at distance k' in C_1 is visited. Thus P is v-majorizing.

If P does not contain c_2, let $c' \in P \cap L_2^w$. If $N(c') \cap N(w)$ contains a vertex that is not a neighbor of C_1, then P will be majorizing by the order above.

Suppose instead that there does not exist a v-majorizing $u' - w$-path. Let $w' = c_2$. We will show that there exists a v-majorizing $u' - w'$-path. The rest of the

proof can be repeated with w' provided v is an eccentric vertex of w', and remains in a non-superior deep component of $G - N[w']$.

First, we verify that that $d(c_2, v) = d(w, v)$. There must be a shortest $u'-v'$-path T that does not use w (as otherwise $d(u', v) > d(u', v')$, a contradiction). The path T must contain a vertex $t \in N(w) \cap (N(c_2) \cap N(C_1))$ since t must have a neighbour in both components. But $d(t, w) = d(t, c_2) = 1$, so $d(v, w) = d(v, t) + 1 = d(v, c_2)$.

Note c' and c_2 are in the same component of $G - N[w]$, so they have a path between them that avoids w. Such a path path has length at most 2. To see this, consider a shortest path Q between c' and c_2 entirely contained in C_2, and let $s \in Q$ be the vertex on the path that is closest to c_2. Now if s is not adjacent to everything in $N(c') \cap N(c_2)$, there is a long cordless cycle. So it must be adjacent to all such vertices. Pick $z \in N(c') \cap N(c_2)$; $s \sim z$. Now let $z' \in N(c_2) \setminus N(c')$, which exists as otherwise c' was a superior vertex. Now w, z', z, c_2, s, c' form a forbidden subgraph of proper interval bigraphs. This shows that for any two vertices in the same component, they have a common neighbour farther from w.

Second, we verify that that $ecc(c_2) = d(c_2, v)$. Suppose to the contrary that this is not true: then there is a vertex f such that $d(f, c_2) > d(v, c_2)$. If $f \in C_2$, $d(f, c_2) \leq d(f, w)$ as the above note states there is a common vertex between c_2 and f', the vertex such that $d(f', w) = 2$ and f' is on a shortest $d(f, w)$ path. So f must be in C_1. Again, let f' be the vertex such that $d(f', w) = 2$ and f' is on a shortest $d(f, w)$ path. It must be the case that $f' \notin N(c_2)$, as otherwise $d(c_2, f) = d(w, f)$ a contradiction. So f' must not be adjacent to anything in C_2; so either $N(C_2) \subsetneq N(C_1)$, contradicting v not being in a superior component, or $N(C_1)$ and $N(C_2)$ are incomparable.

If $N(C_1)$ and $N(C_2)$ are incomparable with $f' \in N(C_1) \setminus N(C_2)$ and $g \in N(C_2) \setminus N(C_1)$, $g \in N(c_2)$. Now $g \nsim c'$, as otherwise there would have been a v-majorizing $w - u'$-path. But then v is not admissible: take a path $g - w$ and append to it any shortest $w - v$ path and it avoids c_2, and $c_2 - t$ and the rest of T to get a path that avoids g.

Let x be the center of such a shortest path Q that avoids w. Since $x \in C_2$ of $G - N[w]$, it is not adjacent to anything in C_1 of $G - N[w']$, and thus $N(C_2)$ of $G - N[w']$ is not strictly contained in $N(C_1)$ of $G - N[w']$.

When the ordering is restarted, x is ordered before any vertex that has a neighbor in C_1, and it is on a shortest $w' - u'$-path. Thus P' is v-majorizing and is our desired path. □

Claim 6. *There exists a v-majorizing $u-w$ path for some LBFS with the above prefix for all $u \in C_2 \cap L^w_{k'}$. Further, any vertex $x \in C_2$ with $d(w,x) = k'$ is visited before we consider the only vertex v in C_1 at distance k' from w with the prefix described above.*

Proof. First, note that for any such u, $N(u) \cap N(u') \neq \emptyset$. Since u' is a dominating pair vertex, u is hit by any $w - v$-path, so it follows that $N(u) \cap N(u') \neq \emptyset$. Thus a v-majorizing $u - w$ path is found by taking the one found in the last claim up to the vertex in $x \in N(u')$ and then picking u instead of u'; if this path is not adjacent to x, then we have found a path that fails to dominate, a contradiction. Using this v-majorizing path, it is clear that every u must be visited before v. □

Thus all vertices in $L^w_{k'}$ contained in C_2 are visited before v. Suppose that there is a $u \in L^w_{k'}$ in C_1 ($u \neq v$). If no such u exists, by the last claim, any LBFS will end at v.

Claim 7. *If all vertices $u \in C_2$ at distance k' from w are such that $N(u) = N(v)$, v can be visited last by LBFS.*

Proof. Since $N(u) = N(v)$, u and v must have the same labels. LBFS breaks ties arbitrarily; visit v last. □

There are two final cases we must consider. Either $N(v) \subsetneq N(u)$, in which case any vertex u always has priority over v and v will be visited after$u$, or the neighborhoods of u and v are incomparable.

Claim 8. *There does not exists a u such that $N(v) \cap N(u) \neq \emptyset$ and $N(v)$ and $N(u)$ are incomparable.*

Proof. Suppose to the contrary that such a vertex u exists. Let $x \in N(v) \setminus N(u)$ and $y \in N(u) \setminus N(v)$. Let P be a shortest $w - v$-path. Since v is a dominating pair vertex, P dominates every vertex in $L^w_{k'}$. It follows that $N(x) \cap P \neq \emptyset$ and similarly for y. Further, they must share a vertex on P in their neighborhoods: it must be the vertex earlier than v' on P, otherwise x and y would be closer to w. Call this vertex z. Now $z \neq w$, since the v and u would be in different components of $G - N[w]$. So there must be a $z' \in P$ that could be w; but then we have a forbidden subgraph of proper interval bigraphs, a contradiction. □

Thus v is an LBFS end-vertex. □

Lemma 4.3.14. *Let $v \in V(G)$ for a proper interval bigraph G such that there is some LBFS ordering σ of G that ends at v. If $ecc(v) = diam(G) - 1$, then the component of $G - N[\sigma(1)]$ containing v is not a superior deep component.*

Proof. Suppose to the contrary this is not true; i.e. assume that $\text{ecc}(v) = \text{diam}(G)-1$, and v is in a superior deep component. Let (u', v') be a diametrical dominating pair (which exist by Theorem 4.3.6), and assume without loss of generality that $d(v, v') \leq d(v, u')$. Let $w = \sigma(1)$. If $G - N[w]$ is not disconnected, there is nothing to prove. So suppose that it is disconnected with components $C_1, \ldots, C_\alpha, C_{\alpha+1}, \ldots, C_\beta$ where C_i $1 \leq i \leq \alpha$ are deep components. Assume without loss of generality that $v \in C_1$. Let $k = \text{diam}(G)$, $k' = \text{ecc}(w)$, and $L_i^w = \{x | d(x, w) = i\}$.

Claim 1. *v is adjacent to v'.*

Proof. This is Lemma 4.3.11. □

Claim 2. $d(u', v) = k - 1$

Proof. If $d(u', v) < k - 1$, then this path (along with Claim 1 and the fact that it cannot be larger than $\text{ecc}(v) = k - 1$) provides a path from u' to v' with length less than k, a contradiction. □

Claim 3. *Unless $k' = 2$, then there are most two deep components in $G - N[w]$.*

Proof. This is Lemma 4.3.10. □

If $k' = 2$, then each deep component is a single vertex. Since by assumption to the contrary, v is in a superior deep component, it contains a nieghbour in $N(w)$ that another deep component y lacks; thus it must be chosen before y, contradicting v being the last vertex.

Claim 4. $u' \in C_2$

Proof. If $k = 1$, $ecc(v) = 0$ contradicting the graph being connected. If $k = 2$, each component is a single vertex, and since v is adjacent to v' with $d(w, v) > d(w, v')$ we must have $d(w, v) = 2$. This is a contradiction as $\text{ecc}(v) \neq k$. For $k > 2$, we must have $u' \in C_2$, otherwise any shortest u', v'-path entirely contained in C_1 fails to dominate some vertex in C_2 (namely, a vertex at maximal distance from w in that component). □

Claim 5. *There exists a shortest $u' - v'$-path that does not use w.*

Proof. By the last claim, $u' \in C_2$. Suppose to the contrary that every shortest path U uses w; then $d(u', v) = 2 \cdot \text{ecc}(w)$ since both v and u' are at maximum distance from w. But then $d(u', v') = d(w, u') + d(w, v') = \text{ecc}(w) + (\text{ecc}(w) - 1)$ (since v and v' are adjacent but v is at distance $\text{ecc}(w)$ from w), contradicting (u', v') achieving the diameter. □

Claim 6. *Some vertex $c \in C_1$ is such that $N(c) = N(C_1)$.*

Proof. This follows from Lemma 4.3.7. □

Claim 7. *Let R be a shortest $u' - v'$-path that does not contain w. Then $c \in R$.*

Proof. We can always find an R which contains c: if $c \notin R$, then there is a $u' - v'$-path that misses some vertex in $N(C_1) \cap N(w)$ (everything that is not on R in $N(C_1) \cap N(w)$), since it does not use w. □

Claim 8. $k = (2 \cdot k') - 1$

Proof. $d(w, v) = k'$ and $d(w, u') = k'$ since both v and u' are eccentric vertices of w (v is there because it is the end-vertex of σ, and u' is there because otherwise we can find a farther vertex from v'). Then $k = d(u', v) + 1 = d(u', x) + 1 + d(v, c) + 1 \leq (k' - 1) + 1 + (k' - 2) + 1 = 2k' - 1$ where $x \in N(C_1) \cap N(C_2)$. On the other hand, $k = d(u', v') \geq d(u', w) + d(w, v') = k' + (k' - 1) = 2k' - 1$ (since there is some shortest $u' - v'$-path that contains w by Lemma 4.3.11). Combining these gives the desired result. □

Claim 9. $d(c, v) \neq d(c, v')$

Proof. Suppose to the contrary that $d(c, v) = d(c, v')$. Let C be a $c - v$-path and C' be a $c - v'$-path. Then since v and v' are adjacent, $c - v - v' - c$ is an odd walk, contradicting the graph being bipartite. □

Claim 10. $d(c, v) = k' - 2$

Proof. First, note that it cannot be shorter: if it were, we could could find a path from w to v of length at most $k' - 3 + 2 = k' - 1$, contradicting $d(w, v) = k'$.

Since there is some k' length path Q from v to w, it must pass through $N(w)$ at some vertex q; since c is a superior vertex, q and c are adjacent. So following Q from q to v has distance $k' - 1$; so we have a path of length k' from c to v by following c to q and then Q to v. Let $q' \in N(q) \cap Q \cap L_2^w$ (i.e. the neighbor of q on Q at the same

distance from w as c), and q'' be the next vertex on the path (i.e. $q'' \in N(q') \cap Q \cap L_3^w$). If q'' and c are adjacent, we are done: $d(q'', v) = d(w, v) - 3 = k' - 3$ and using this, along with c, provides the desired path. If not, let $c' \in N(c) \cap R \cap L_3^w$; q' and c' must be adjacent to avoid a cycle of length at least 6.

Note that $d(c, v') \leq k' - 1$, as $d(u', c) \geq (k' - 1) + 1 = k'$ and these distances must add to $2k' - 1$. So we can repeat this argument along Q and R; when R terminates, since v and v' are neighbors, add v to the end of R from c to v'. (If R and Q ever coincide, follow R to where they join, continue to v', and then add v.) □

Claim 11. $d(c, v') = k' - 1$

Proof. Note that $d(c, v') \leq k' - 1$, as $d(u', c) \geq (k' - 1) + 1 = k'$ and these distances must add to $2k' - 1$. If $d(c, v') < k' - 1$, then we could find a $u' - v'$-path with length $d(u', c) + d(c, v') = k < k' + k' - 1 = 2k' - 1$, contradicting that the diameter is exactly this value. □

Claim 12. *There exists a shortest $u' - v'$-path R that contains v.*

Proof. By the last two claims, v is closer to c than v'; follow any shortest path from c to v and then append v'. □

Claim 13. *There exists a shortest $c - v$-path R' that does not contain v'.*

Proof. Follow R as found in Claim 12, and stop when you get to v. □

Claim 14. *R' contains exactly one vertex in each set L_m^w, $k' \geq m \geq 1$.*

Proof. There are exactly $k' - 2$ vertices on the path and $d(c, v) = k' - 2$. □

Claim 15. *Let $x \in R' \cap N(w)$. There exists a u'-majorizing $x - v$-path.*

Proof. Since $N(c) \cap N(w)$ is maximal in the graph, c is the first vertex at distance 2 from w visited. Since $c \in R'$, then c's neighbor on R' is always visited first in a set L_i^w. Since R' contains exactly one vertex in each set L_i^w for each i, its neighbor in L_{i+1}^w is always the first vertex visited in that set. □

Claim 16. *There exists a u'-majorizing $w - v$-path.*

Proof. By the last claim, there is a u'-majorizing $x - v$-path; union this with w (whose label must be the biggest, as it is the first vertex in the ordering), and we get the desired path. □

Claim 17. *v could not have been the end-vertex.*

Proof. The existence of the path from the last claim certifies this. □

The last claim shows a contradiction, so we are done. □

We are now ready to prove Theorem 4.3.9.

Proof of Theorem 4.3.9. First, suppose that v satisfies all of the conditions listed in Theorem 4.3.9. If $\text{ecc}(v) = \text{diam}(G) - 1$, apply Lemma 4.3.13. Assume $\text{ecc}(v) = \text{diam}(G)$. Let σ' be some LBFS ordering starting with v, with $\sigma'(n) = x$. Since v is admissible, by Theorem 2.3.14, (x, v) is a dominating pair of G. Moreover, v is an eccentric vertex of x. Therefore, by Lemma 4.3.12, we are done.

We now prove the other direction.

1. This is Theorem 2.3.5.
2. This is Theorem 2.0.2.
3. This follows from Theorem 2.3.3 and the fact that proper interval bigraphs are AT-free (Theorem 4.3.9).
4. This is Proposition 4.3.8.
5. If $\text{ecc}(v) = \text{diam}(G) - 1$, this is shown in Lemma 4.3.14. If $\text{ecc}(v) = \text{diam}(G)$, take $w = \sigma(1)$.

□

We can now prove the desired complexity result.

Theorem 4.3.15. *The LBFS end-vertex problem for proper interval bigraphs is solvable in polynomial time.*

Proof. To check if a vertex is a cut vertex, remove it from the graph and compute the number of new components. Corneil, Olariu, and Stewart showed that the set of all admissible vertices in an AT-free graph can be obtained in linear time [15]. Checking the neighbourhood inclusion condition is and eccentricity straightforward. The last condition can be checked by first collecting the set of vertices for which the last vertex would be eccentric. Then remove each candidate start vertex and its closed neighbourhood from the graph, checking if there is one for which components' neighbourhoods satisfy the inclusion property. □

Algorithm 7: 2-Sweep LBFS

Data: $G = (V, E)$
Result: A vertex v
begin
  Let w be an arbitrary vertex;
  $u \leftarrow$ the last vertex numbered by LBFS(w);
  $v \leftarrow$ the last vertex numbered by LBFS(u);
  Return v;

If v is an LBFS end-vertex with $\text{ecc}(v) = \text{diam}(G)$, there may be a vertex w such that v is in a superior deep component of $G - N[w]$. This is illustrated in Figure 4.8.

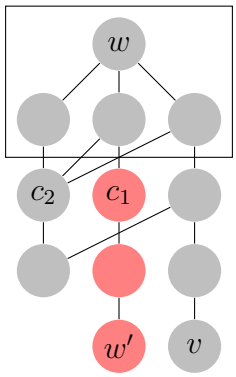


Figure 4.8: A example where $\text{ecc}(v) = \text{diam}(G)$ and v is in a superior deep component of $G - N[w]$.

We conclude the section with a lemma similar to the flipping lemma for interval graphs (Lemma 2.3.15). Conditions for when an LBFS ordering can be flipped are characterized in the next lemma. This requires the following results and the introduction of *2-Sweep LBFS*, Algorithm 7. The algorithm is performed by starting LBFS at an arbitrary vertex, followed by another execution of LBFS starting at the end-vertex of the first execution. Note that unlike other algorithms that use LBFS in successive sweeps, we do not care about the entire order, or use the first order to determine how to break ties in slices (as in e.g. LBFS+ [9]).

Proposition 4.3.16 (Corneil, Dragan, Habib, and Paul [10])**.** *Let G be an arbitrary graph and u be the vertex of G visited last by a LBFS. If $diam(G) = 2$, then $ecc(u) = diam(G)$.* □

Proposition 4.3.17 (Corneil, Dragan, Habib, and Paul [10])**.** *Let G be an AT-free graph with $diam(G) = k > 2$. If $ecc(v) = k - 1$, where v is the vertex returned by Algorithm 7, and u', v' achieve the diameter where $d(u, u') \leq d(u, v')$, then*

1. $d(u, v) = d(u, v') = d(u', v) = k - 1$
2. $uu' \in E$ *and* $vv' \in E$.

□

Lemma 4.3.18 (Flipping lemma for proper interval bigraphs)**.** *Let G be a proper interval bigraph and let $y, z \in V$ be two vertices that satisfy Theorem 4.3.9. If $\sigma = y \dots z$ (where σ is LBFS ordering), then there exists an LBFS ordering $\sigma' = z \dots y$ if and only if $ecc(y) = diam(G) = ecc(z)$.*

Proof. First, suppose that $\text{ecc}(y) = \text{diam}(G) = \text{ecc}(z)$. By Theorem 2.3.14, $D(y, z) = V$ meaning (y, z) is a dominating pair of G, so by Lemma 4.3.12 we are done.

Suppose instead that we have an ordering $\sigma' = z \dots y$. Since each of y and z is an eccentric vertex of the other, we have that $\text{ecc}(y) = \text{ecc}(z)$ (to see this, note that σ implies that $d(y, z) = k' = \text{ecc}(y)$ for some k', and σ' implies that $\text{ecc}(z) = d(y, z) = k'$ as there is nothing farther than y).

If $\text{diam}(G) = 2$, then by Proposition 4.3.16, we are done. So assume that $\text{diam}(G) > 2$ and to the contrary that $\text{ecc}(y) = \text{diam}(G) - 1 = \text{ecc}(z)$. Let k denote $\text{diam}(G)$. Note that the existence of σ and σ' can be seen as an instance of Algorithm 7 with $w = y$, $u = z$, $v = y$. By Theorem 4.3.6, some pair (u', v') achieves the diameter; clearly $z \neq u'$ and $y \neq v'$. Assume without loss of generality, $d(u', z) \leq d(v', z)$.

By Proposition 4.3.17, we have that

$$d(z, y) = d(z, v') = d(u', y) = k - 1$$

and $zu' \in E$ and $yv' \in E$. But this means that $v', y \in L^z_{k-1}$, and they are adjacent, contradicting G being bipartite. □

Chapter 5

Future Work

Our results show a boundary of where the end-vertex problem becomes tractable for LBFS when restricted to bipartite graphs. In particular we show that it is polynomial time solvable for proper interval bigraphs. It would be interesting to determine if the problem remains tractable in the larger class of chordal bigraphs.

We have shown that for chordal graphs, the LDFS end-vertices coincide with MNS end-vertices. For these graphs the end-vertex problem is solvable in polynomial time for MNS and LDFS. However, recognizing LBFS end-vertices remains an open problem on chordal graphs. We expect that for LBFS the problem is also tractable, although it remains to be proven.

The beginning-end-vertex problem, as shown in the case of bipartite graphs, provides valuable insight as to the difficulty of the related end-vertex problem. However, little work has been done on the problem. Solving this problem may be easier than the end-vertex problem, and would still provide a complexity result where applicable. Despite this, no results exist for LDFS, or MNS, even though these algorithms lack corresponding end-vertex results.

All of the "?" entries in Tables 3.1 and 4.1 are open problems. Additionally, it would be interesting to find new applications for these problems. Could these results be used to show that other common graph theoretic problems (e.g. graph class recognition) on these classes are also tractable through new applications of these algorithms?

Bibliography

[1] A. Berry, J. R. S. Blair, J-P. Bordat, and G. Simonet. Graph extremities defined by search algorithms. *Algorithms*, 3(2):100–124, 2010.

[2] J. A. Bondy and U. S. R. Murty. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[3] A. Brandstädt, F. F. Dragan, and F. Nicolai. Lexbfs-orderings and powers of chordal graphs. *Discrete Mathematics*, 171(1):27–42, 1997.

[4] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph classes: a survey*, volume 3. SIAM, 1999.

[5] P. Buneman. A characterisation of rigid circuit graphs. *Discrete Mathematics*, 9(3):205 – 212, 1974.

[6] J-M. Chang, C-W. Ho, and M-T. Ko. Lexbfs-ordering in asteroidal triple-free graphs. In *Algorithms and Computation*, pages 163–172. Springer, 1999.

[7] P. Charbit, M. Habib, and A. Mamcarz. Influence of the tie-break rule on the end-vertex problem. *Discrete Mathematics & Theoretical Computer Science*, 16(2):57–72, 2014.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to algorithms. *MIT Press*, 5(3):55, 2001.

[9] D. G. Corneil. Lexicographic breadth first search–a survey. In *Graph-theoretic concepts in computer science*, pages 1–19. Springer, 2005.

[10] D. G. Corneil, F. F. Dragan, M. Habib, and C. Paul. Diameter determination on restricted graph families. *Discrete Applied Mathematics*, 113(2):143–166, 2001.

[11] D. G. Corneil and E. Köhler. Unpublished manuscript. Cited in [12].

[12] D. G. Corneil, E. Köhler, and J. Lanlignel. On end-vertices of lexicographic breadth first searches. *Discrete Applied Mathematics*, 158(5):434–443, 2010.

[13] D. G. Corneil and R. M. Krueger. A unified view of graph searching. *SIAM Journal on Discrete Mathematics*, 22(4):1259–1276, 2008.

[14] D. G. Corneil, S. Olariu, and L. Stewart. Asteroidal triple-free graphs. *SIAM Journal on Discrete Mathematics*, 10(3):399–430, 1997.

[15] D. G. Corneil, S. Olariu, and L. Stewart. Linear time algorithms for dominating pairs in asteroidal triple-free graphs. *SIAM Journal on Computing*, 28(4):1284–1297, 1999.

[16] D. G. Corneil, S. Olariu, and L. Stewart. The lbfs structure and recognition of interval graphs. *SIAM Journal on Discrete Mathematics*, 23(4):1905–1953, 2009.

[17] G. A. Dirac. On rigid circuit graphs. In *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, volume 25, pages 71–76. Springer, 1961.

[18] F. F. Dragan. Almost diameter of a house-hole-free graph in linear time via lexbfs. *Discrete applied mathematics*, 95(1):223–239, 1999.

[19] F. F. Dragan and F. Nicolai. *LexBFS Orderings of Distance Hereditary Graphs*. UD, Fachbereich Mathematik, 1995.

[20] M. Farber. Characterizations of strongly chordal graphs. *Discrete Mathematics*, 43(2):173–189, 1983.

[21] D. Fulkerson and O. Gross. Incidence matrices and interval graphs. *Pacific journal of mathematics*, 15(3):835–855, 1965.

[22] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974.

[23] M. C. Golumbic. *Algorithmic graph theory and perfect graphs*, volume 57. Elsevier, 2004.

[24] M. C. Golumbic and C. F. Goss. Perfect elimination and chordal bipartite graphs. *Journal of Graph Theory*, 2(2):155–163, 1978.

[25] P. Heggernes and D. Kratsch. Linear-time certifying algorithms for recognizing split graphs and related graph classes. *Reports in Informatics*, 328, 2006.

[26] P. Hell and J. Huang. Interval bigraphs and circular arc graphs. *Journal of Graph Theory*, 46(4):313–327, 2004.

[27] A. J. Hoffman, A. W. J. Kolen, and M. Sakarovitch. Totally-balanced and greedy matrices. *SIAM Journal on Algebraic Discrete Methods*, 6(4):721–730, 1985.

[28] J. Hopcroft and R. Tarjan. Efficient planarity testing. *Journal of the ACM (JACM)*, 21(4):549–568, 1974.

[29] R. M. Karp. *Reducibility among combinatorial problems*. Springer, 1972.

[30] J. Kleinberg and É. Tardos. *Algorithm design*. Pearson Education India, 2006.

[31] E. Köhler and L. Mouatadid. Linear time lexdfs on cocomparability graphs. In *Algorithm Theory–SWAT 2014*, pages 319–330. Springer, 2014.

[32] R. M. Krueger. *Graph searching*. University of Toronto, 2005.

[33] C. Lekkerkerker and J. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1):45–64, 1962.

[34] P. Li and Y. Wu. A four-sweep lbfs recognition algorithm for interval graphs. *Discrete Mathematics & Theoretical Computer Science*, 16(3):23–50, 2014.

[35] G. B. Mertzios and D. G. Corneil. A simple polynomial algorithm for the longest path problem on cocomparability graphs. *SIAM Journal on Discrete Mathematics*, 26(3):940–963, 2012.

[36] C. L. Monma and V. K. Wei. Intersection graphs of paths in a tree. *Journal of Combinatorial Theory, Series B*, 41(2):141 – 181, 1986.

[37] H. Müller. Hamiltonian circuits in chordal bipartite graphs. *Discrete Mathematics*, 156(13):291 – 298, 1996.

[38] M. J. Pelsmajer, J. Tokaz, and D. West. New proofs for strongly chordal graphs and chordal bipartite graphs. *preprint*, 2004.

[39] D. J. Rose. Triangulated graphs and the elimination process. *Journal of Mathematical Analysis and Applications*, 32(3):597–609, 1970.

[40] D. J. Rose, R. Tarjan, and S. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on computing*, 5(2):266–283, 1976.

[41] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[42] J. R. Walter. Representations of chordal graphs as subtrees of a tree. *Journal of Graph Theory*, 2(3):265–267, 1978.

[43] D. B. West. *Introduction to graph theory*, volume 2. Prentice Hall Upper Saddle River, 2001.

[44] S-J. Xu, X. Li, and R. Liang. Moplex orderings generated by the lexdfs algorithm. *Discrete Applied Mathematics*, 161(13-14):2189 – 2195, 2013.