Domination : Offline, Online, Any Time

Jesse Racicot

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Science (Computer Science) at

Concordia University

Montréal, Québec, Canada

May 2021

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Jesse Racicot**

Entitled: **Domination : Offline, Online, Any Time**

and submitted in partial fulfillment of the requirements for the degree of

Master of Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Jaroslav Opatrny

_____ External Examiner
Dr. Nadia Hardy

_____ Examiner
Dr. Jaroslav Opatrny

_____ Thesis Supervisor(s)
Dr. Hovhannes Harutyunyan

_____ Thesis Supervisor(s)
Dr. Denis Pankratov

Approved by _____
Dr. Lata Narayanan, Chair
Department of Computer Science and Software Engineering

_____ May 12, 2021

_____
Dr. Mourad Debbabi, Interim Dean
Gina Cody School of Engineering and Computer Science

Abstract

Domination : Offline, Online, Any Time

Jesse Racicot

Given a *graph* $G = (V, E)$, a subset $D \subseteq V$ is called a *dominating set* if each vertex $v \in V$ either belongs to D or is adjacent to some vertex in D. The typical objective is to find a dominating set of minimum size. Depending on the context, the problem may be viewed from an algorithmic perspective or a purely mathematical one. The following thesis explores the topic of domination from these two different perspectives, where the former is split further into two settings. In particular, we study the topic as a computational problem within an *offline setting* where an algorithm is given an input graph in its entirety and alternatively, in an *online setting* where an algorithm is forced to make irrevocable decisions with limited information about an input graph. We also approach the topic as a purely mathematical problem as we study the domination number of a well-known family of graphs known as the *Knödel* graphs.

Prior to this thesis, research on the dominating set problem in an online setting was sparse. We consider an online setting where a vertex is revealed to an algorithm and the choice to add this vertex or not is a finality. In this setting, an adversary must reveal the entire neighborhood of a vertex to an algorithm while keeping the revealed portion of the graph connected at all times. We present algorithms that achieve 2-competitiveness on trees, 2.5-competitiveness on cactus graphs, $(t-1)$ on $K_{1,t}$-free graphs, and $\Theta(\sqrt{\Delta})$ for maximum degree Δ graphs. Moreover, we show that all of those competitive ratios are tight. Then, we study several more general classes of graphs, such as threshold, bipartite planar, and series-parallel graphs, and show that they do not admit competitive algorithms (that is, when competitive ratio is independent of the input size). Our results are compared with earlier results in a different input model, where a vertex is revealed alongside

its restricted neighborhood: those neighbors that are among already revealed vertices. Thus, conceptually, our results quantify the value of knowing the entire neighborhood at the time a vertex is revealed as compared to the restricted neighborhood.

The family of graphs known as the Knödel graphs are studied extensively with an emphasis on determining closed form expressions for certain graph parameters. Our main contribution is the novel use of techniques from elementary number theory to establish an upper bound on the *domination number* of the Knödel graph on n vertices. In particular, we show that whenever we find a prime p dividing n with $p \leq \lceil \log n \rceil$ such that 2 is a *primitive root* modulo p then there is a dominating set of size $\frac{n}{p}$. Moreover, if we suppose that 2 is a primitive root modulo p^k, where p^k divides n and $\phi(p^k) < \lceil \log n \rceil$ then we can construct a dominating set of size $\frac{2n}{p^k}$.

Contents

List of Figures

Chapter 1

Introduction

Planning a party that people enjoy is far from an exact science but planning one that people attend can be. Oftentimes, we can both understand "what" influences someone and "who" influences them. Perhaps a monetary amount, or another quantifiable object of value, may be enough to drag someone to a party (although an additional fee is probably required if you want them to dance). Even more coercive than money, is the influence of their close friends, who may have already been bought out. As a thought experiment, suppose we could examine a group of people, and determine an exact price for each person, that would ensure they attend a party. Moreover, suppose that we can determine for each person in the group, the friends of theirs that they will convince to come (granted that this person has already been payed). That is, we can safely assume a person will attend if they have been payed or are directly influenced by someone who has been paid. You may find that some demand a higher price but when they do attend, they bring with them many others, making your dollars well-spent. Although, this is not always the case as some people cost an "arm and a leg" and bring only their own (i.e. isolated individuals who trust the dollar more than their friends). If you want to ensure that everyone attends the party, you can select some number of people you will pay, obtain the guarantee that their friends will show, and work out whether this accounts for everyone. If the "network" is adversarially interconnected, consists of more than a few hundred people, and you wish to pay as little as possible then you probably won't figure this out in your lifetime.

Much like this example, there are many other applications in the area of network communication that allow certain physical details of the network to be ignored and instead only the nodes and the

intercommunication lines between pairs of nodes are considered. This abstraction of a network can be modeled as a graph and this motivating example is a concrete application of the minimum weight *dominating set* problem.

A *graph*[1] is a pair $G = (V, E)$ where V is a finite set of elements called *vertices* and E is a collection of unordered pairs of V called *edges*. Thus, the term "vertex" substitutes for the term "node", "edge" for "communication line", and "graph" for "network". Given two vertices $u, v \in V$ we say that u is *adjacent* to v if $\{u, v\} \in E$. A subset $D \subseteq V$ is called a *dominating set* if each vertex $v \in V$ either belongs to D or is adjacent to some $u \in D$. One can always obtain a trivial dominating set by selecting $D = V$, and per the definition, this is as large as a dominating set can possibly be. Therefore the goal is typically to find dominating sets of small size, with the primary aim being to determine $\gamma(G)$, the size of a minimum dominating set in G.

More than a century ago, de Jaenisch (1862) asked for the minimum number of queens that need to be placed on a chessboard so that every square is either occupied by a queen or attacked by a queen. If we consider the $8 \times 8 = 64$ squares of the chessboard as vertices of a graph where two squares are adjacent if either can attack the other with a queen's move, then this question is precisely the problem of finding a minimum dominating set in this "queen's graph". This is often considered the origins of the dominating set problem although no explicit mention of graphs were made. Berge (1962) is credited for first introducing the idea of a dominating set in a pure mathematical setting, where he defined the "coefficient of external stability" as the size of a minimum dominating set in G although Ore (1962) was the first to introduce the terms "dominating set" and "domination number" of a graph. Cockayne and Hedetniemi (1977), two of the largest contributors on the topic, introduced the widely used notation $\gamma(G)$.

The topic of domination is also widely studied by computer scientists due to important practical and theoretical applications, such as establishing surveillance service Berge (1962), routing and transmission services in (wireless) networks Das and Bharghavan (1997), as well as broadcasting Harutyunyan (2008); Harutyunyan and Liestman (2012). Finding a minimum sized dominating set in a graph is a difficult computational problem. In particular, given a graph G and an integer k, deciding whether $\gamma(G) \leq k$ is an NP-complete problem Garey and Johnson (1979).

[1] In this thesis we only deal with simple undirected graphs.

The set cover problem was shown to be NP-complete by Karp (1972), and given the similarities between the set cover and dominating set problems, some may regard this as the origins for domination as a computational problem. It is not only NP-complete to decide whether $\gamma(G) \leq k$, but there cannot be any constant-factor approximations for the optimization problem under commonly believed complexity assumptions. In particular, Lund and Yannakakis (1994) show that if there is a polynomial-time algorithm that achieves approximation ratio $c \log n$ for $c < 1/4$ then $NP \subset DTIME(n^{O(poly \log n)})$ and Feige (1998) proved that if a polynomial-time algorithm achieves approximation ratio $(1 - \epsilon) \ln n$ for any $\epsilon > 0$ then $NP \subset DTIME(n^{O(\log \log n)})$.

In modern times, some networks are extremely large and classical techniques can sometimes be of little practical use. Oftentimes, a practitioner is inclined to design algorithms that are contingent on only a small amount of local structure instead of the entire global structure of a network De Meo, Ferrara, Fiumara, and Provetti (2014); Shun, Roosta-Khorasani, Fountoulakis, and Mahoney (2016). Thus, in essence we have designed an algorithm that is supplied with limited information, e.g., only the local structure, as well as having restrictions on its decision making. The general problem of withholding information from an algorithm, demanding an irrevocable choice from said algorithm, and then comparing the performance with that of an optimal algorithm (i.e. sees the input in its entirety), is the interest of the area of online computation. The origins of online computation are due to Graham (1966) who analyzed an online greedy algorithm for the makespan problem. Sleator and Tarjan (1985) argued for worst-case analysis as opposed to the more popular average case analysis of online algorithms at the time. The terms competitive analysis and competitive ratio were introduced in Karlin, Manasse, Rudolph, and Sleator (1988). An extensive treatment of the area of online computation can be found in the following books; Borodin and El-Yaniv (1998); Komm (2016). The study of the dominating set problem in an online setting has received considerably less attention than the offline setting (i.e. one in which an algorithm is provided as input a graph in its entirety) with few papers published. In fact, the entirety of the research comprises of the following papers; Böckenhauer, Hromkovič, Krug, and Unger (2021); Boyar, Eidenbenz, Favrholdt, Kotrbcík, and Larsen (2019); Eidenbenz (2002); King and Tzeng (1997); Kobayashi (2017).

The remainder of the thesis is organized as follows: Chapter 2 provides the necessary preliminaries for the thesis along with some basic, yet important facts about domination. Chapter 3 explores

domination in the classical offline setting; The chapter is a brief survey of offline domination. In Chapter 4, the problem is explored in an online setting; The entire chapter comprises of an original contribution which introduces and explores the problem in a previously unexplored setting. Chapter 5 presents an original result on the domination number of a well-known family of graphs, the Knödel graphs. Chapter 6 concludes with a summary of the thesis and potential directions of future research.

Chapter 2

Preliminaries

In this section we provide graph theoretic definitions and some basic results that help ease the reader's intuition on the topic of domination. For an overview of graph theory, three classical books are Berge (1962); Harary (1969); Ore (1962) and a more modern text is Bollobás (1998). The reader interested in domination specifically can refer to Haynes, Hedetniemi, and Slater (1998). The first section consists of definitions that are used throughout the thesis along with some basic, yet fundamental results about dominating sets. The second section provides results on some common graph families. The first section can be considered the necessary preliminaries for the thesis whereas the second is primarily intended to be pedagogical.

2.1 Definitions

Unless otherwise specified, the term graph $G = (V, E)$ refers to an undirected graph without loops at a vertex and at most one edge between two vertices. We sometimes use the notation $V(G)$ and $E(G)$ to denote the vertex set and edge set of G, respectively. If $|V| = n$ and $|E| = m$ we say that G is a graph on n vertices with m edges. Given two vertices $u, v \in V$ we say that u is *adjacent* to v (or u and v are *neighbors*) if $\{u, v\} \in E$ and that the edge $\{u, v\}$ is *incident* on u and v. We define the *open neighborhood* of v, denoted by $N(v)$, to be set of neighbors of v, that is, $N(v) = \{u \in V \mid \{u, v\} \in E\}$. The *closed neighborhood* of v, denoted by $N[v]$, is defined as $N(v) \cup \{v\}$. The degree of a vertex, denoted by $deg(v)$, is the number of neighbors of v, that

is, $deg(v) = |N(v)|$. We say that v is *isolated* whenever $deg(v) = 0$ and that v is a *leaf* when $deg(v) = 1$. The *minimum* and *maximum* degree over all vertices in V is denoted by $\delta(G)$ and $\Delta(G)$, respectively. Typically, only δ and Δ are used when the graph that is in context is clear.

A subset $D \subseteq V$ is called a *dominating set* if each vertex $v \in V$ either belongs to D or is adjacent to some $u \in D$. The *domination number* of G, denoted by $\gamma(G)$, is the size of a smallest dominating set of G. A dominating set with $\gamma(G)$ vertices is called a *minimum dominating set* or a γ-set of G.

As a brief aside, we provide a list of observations followed by two basic bounds on the domination number of a graph. The upper bound is attributed to Ore (1962) and the lower bound was originally proved by Berge (1962).

Observation 2.1.1. *Let $G = (V, E)$ be a graph on n vertices;*

- *$\gamma(G) = 1$ if and only if there is some $v \in V$ with $deg(v) = n - 1$.*
- *$\gamma(G) = n$ if and only if every vertex in V is isolated.*
- *If $v \in V$ is isolated then it belongs to every dominating set of G.*
- *If $v \in V$ is a leaf then there is a γ-set of G that does not contain v and contains the unique neighbor of v.*

Theorem 2.1.1. *Ore (1962) If G is graph on n vertices with no isolated vertices then $\gamma(G) \leq \frac{n}{2}$.*

Theorem 2.1.2. *Berge (1962) If G is graph on n vertices then $\gamma(G) \geq \left\lceil \frac{n}{\Delta+1} \right\rceil$.*

We extend the definitions of open and closed neighborhoods of subsets by taking unions. That is, given a subset $S \subseteq V$ we define $N(S) = \bigcup_{v \in S} N(v)$ and $N[S] = \bigcup_{v \in S} N[v] = N(S) \cup S$. A *path* from u to v is a sequence of distinct vertices $(v_0, v_1, ..., v_k)$ where $v_0 = u$, $v_k = v$, and $\{v_i, v_{i+1}\} \in E$ for $0 \leq i \leq k - 1$. Often, we will refer to the edges that appear in the path since it is unambiguous to do so given the definition of a graph we use.[1] The *length* of a path is the number of edges in the path. The *distance* between u and v, denoted by $d(u, v)$ is the length of a minimum-length path from u to v. If there is no path from u to v we define $d(u, v) = \infty$.

[1] Since we only consider graphs where there is at most one edge between two vertices, a sequence of vertices given by a path uniquely determines a sequence of edges.

With the preceding definitions in mind, we have the following equivalent definitions for a dominating set.

Observation 2.1.2. *Let $G = (V, E)$ be a graph and $D \subseteq V$. Then D is a dominating set of G if and only if any of the following hold:*

(1) For all $v \in V \setminus D$ we have $|N(v) \cap D| \geq 1$

(2) For all $v \in V$ we have $|N[v] \cap D| \geq 1$

(3) $N[D] = V$

(4) For all $v \in V \setminus D$ there is some $u \in D$ with $d(u, v) = 1$.

A graph G is said to be *connected* if there is a path between any two vertices in G. The subgraph of G *induced* on $S \subseteq V$, which we denote by $\langle S \rangle$, is the subgraph of G with exactly the vertices of S and the edges in G that have both endpoints in S. That is, $\langle S \rangle = (S, E_S)$ where $E_S = \{\{u, v\} \in E \mid u, v \in S\}$. A *connected component* of G is an induced subgraph that is maximally connected (i.e. adding any other vertex to the component results in a graph that is no longer connected). The following observation shows that it is straightforward to carry over domination results from connected graphs to those that are not connected.

Observation 2.1.3. *If $G_1, ..., G_k$ are the $k \geq 1$ connected components of G then $\gamma(G) = \sum_{i=1}^{k} \gamma(G_i)$.*

Consider a dominating set $D \subseteq V$ and a vertex $v \in V \setminus D$. If a message is to be communicated along a path from v to some vertex $u \in D$, v can share the message to a neighbor $w \in D$, which is guaranteed to exist, and w may then propagate the message to u along a shortest path.[2] If D is chosen so that pairwise distances (between vertices within D) is minimized then communication between every pair of vertices in V can be efficient. Below is an observation that formalizes the preceding comments. Hopefully, it provides the reader with a rough sense of how a dominating set, when chosen appropriately, can be useful in networking communication.

Observation 2.1.4. *Let $G = (V, E)$ be a graph and $D \subseteq V$ be a dominating set. If $u \in D$ and $v \in V \setminus D$ then $d(u, v) \leq \max\{d(u, w) \mid w \in D\} + 1$.*

[2]Remark that a shortest path from w to u may contain edges that are incident on vertices in $V \setminus D$.

The *diameter* of a graph, denoted by $diam(G)$ is the maximum distance between any pair of vertices in $V(G)$. One might imagine a non-trivial relationship between domination number and diameter of a graph. A graph with diameter 1 is a graph where all vertices are pairwise adjacent, therefore it has a dominating set of size one. A graph with diameter 2 is slightly more interesting but one can still make positive claims about dominating sets. The following result is attributed to Haynes et al. (1998).

Theorem 2.1.3. *Haynes et al. (1998) If* $diam(G) = 2$ *then* $\gamma(G) \leq \delta(G)$.

Proof. Let $v \in V$ be an arbitrary vertex, we show that $N(v)$ is a dominating set. In particular, we show that any $x \in V \setminus N(v)$ has a neighbor in $N(v)$. Clearly, if $x = v$ then we are done so we take $x \in V \setminus N(v)$ to be different from v. Remark that it is not adjacent to v by definition. Since $diam(G) = 2$ there must be a path of length 2 from x to v. That is, x must share a common neighbor with v and thus x is adjacent to some vertex in $N(v)$. To finish the claim, simply consider a vertex v with $deg(v) = \delta(G)$. □

Consider a connected graph and a pair of vertices u, v. Notice that any vertex along a shortest path from u to v has at most two neighbors on the path since a third neighbor along the path would yield a shorter path from u to v. Similar reasoning yields that any vertex not on the path has at most three neighbors that lie on this shortest path. This observation allowed Haynes et al. (1998) to give a lower bound on the domination number, as a function of the diameter.

Theorem 2.1.4. *Haynes et al. (1998) If* G *is connected then* $\gamma(G) \geq \left\lceil \frac{diam(G)+1}{3} \right\rceil$.

Proof. Let $diam(G) = k$ and assume that $k \geq 3$ since the statement is trivial for $k < 3$. Take vertices v_0, v_k such that $d(v_0, v_k) = k = diam(G)$ and let $P = (v_0, v_1, ..., v_k)$ be a shortest path from v_0 to v_k. Moreover, let V_P denote the vertices on the path. We show that any dominating set D satisfies $|D| \geq \left\lceil \frac{k+1}{3} \right\rceil$.

In earlier comments we remarked that any vertex on a shortest path has at most two neighbors that lie on the path and any vertex not on a shortest path has at most three neighbors that lie on the path. In particular, we have that any vertex in D dominates at most 3 vertices on the path P. That is, for any $v \in D$ we have $|N[v] \cap V_P| \leq 3$. Since D is a dominating set we therefore have that

$N[D] = V \implies N[D] \cap V_P = V_P$. Therefore we have,

$$|V_P| = |N[D] \cap V_P| = |(\bigcup_{v \in D} N[v]) \cap V_P| = |\bigcup_{v \in D} (N[v] \cap V_P)| \leq 3|D|.$$

That is, $k + 1 = |V_P| \leq 3|D| \implies \frac{k+1}{3} \leq |D|$ and the statement follows because $|D|$ is an integer. □

Finally, we should mention that in the literature there are many variations of a dominating set which impose conditions on the induced subgraph of the dominating set itself. For example, a subset $D \subseteq V$ is called an *independent dominating set* if $\langle D \rangle$ is a graph with only isolated vertices, a *total dominating set* if $\langle D \rangle$ is a graph with no isolated vertices, and a *connected dominating set* if $\langle D \rangle$ is a connected graph.

2.2 Results on Common Graph Families

In this section we define some well-known graph families and give their domination number as a closed form expression, e.g. as a function of n where n indexes the family.

Definition 2.2.1.

- $\mathbf{K_n}$ *: The complete graph on n vertices, denoted by K_n, is the graph in which every pair of vertices are adjacent. That is, given a positive integer n, $|V(K_n)| = n$ and $E(K_n) = \{\{u, v\} \mid u, v \in V(K_n), u \neq v\}$.*

- $\mathbf{K_{m,n}}$ *: For $1 \leq m \leq n$, the complete bipartite graph with parts of size m and n, denoted by $K_{m,n}$, is the bipartite graph in which each vertex of one part is adjacent to every vertex in the other part. That is, $V(K_{m,n}) = X \cup Y$ where $|X| = m$, $|Y| = n$ and $E(K_{m,n}) = \{\{x, y\} \mid x \in X, y \in Y\}$. Whenever $|X| = m = 1$ we call this a star with n leaves and the only vertex in X is called the center of the star.*

- $\mathbf{P_n}$ *: For $n \geq 1$, we let P_n denote the path on n vertices where $V(P_n) = \{v_i \mid 1 \leq i \leq n\}$ and $E(P_n) = \{\{v_i, v_{i+1}\} \mid 1 \leq i \leq n - 1\}$.*

- $\mathbf{F_{n,k}}$ *: For $n \geq 2$ and $k \geq 1$, we let $F_{n,k}$ denote the fork on n vertices with k leaves which is a graph containing a path on $n - k$ vertices, where the $(n-k)$'th vertex is the center of a star*

with k leaves. That is, $V(F_{n,k}) = \{v_i \mid 1 \leq i \leq n\}$ and $E(F_{n,k}) = E(P_{n-k}) \cup \{\{v_{n-k}, v_j\} \mid n-k+1 \leq j \leq n\}$.

- $\mathbf{C_n}$ *: For $n \geq 3$, we let C_n denote the cycle on n vertices where $V(C_n) = \{v_i \mid 1 \leq i \leq n\}$ and $E(C_n) = E(P_n) \cup \{\{v_n, v_1\}\}$.*

- $\mathbf{W_n}$ *: For $n \geq 3$, we let W_n denote the wheel with n spokes which is a graph that consists of a cycle on n vertices with an additional vertex that is adjacent to every vertex along the cycle. That is, $V(W_n) = \{v_i \mid 1 \leq i \leq n+1\}$ and $E(W_n) = E(C_n) \cup \{\{v_i, v_{n+1}\} \mid 1 \leq i \leq n\}$.*

- $\mathbf{T_{m,h}}$ *: For $m \geq 2$ and $h \geq 0$, we let $T_{m,h}$ denote the* perfect *m-ary tree of height h. We provide a recursive definition for $T_{m,h}$. A single vertex is a perfect m-ary tree of height $h = 0$ (where this vertex is the root of the tree). For $h \geq 1$, the complete m-ary tree of height h has a specified vertex r as the root, where r has exactly m children and each child is the root of a perfect m-ary tree of height $h-1$. Equivalently, a perfect m-ary tree of height h is a rooted tree in which each internal vertex has exactly m children and every leaf is at distance h from the root.*

Observation 2.2.1.

- *For $1 = m \leq n$, $\gamma(W_n) = \gamma(K_n) = \gamma(K_{m,n}) = 1$.*

- *For $2 \leq m \leq n$, $\gamma(K_{m,n}) = 2$.*

- *For $n \geq 1$, $\gamma(P_n) = \gamma(C_n) = \lceil \frac{n}{3} \rceil$.*

- *For $n \geq 2$ and $k \geq 1$, $\gamma(F_{n,k}) = 1 + \gamma(P_{n-k-2}) = 1 + \lceil \frac{n-2-k}{3} \rceil$.*

- *Let $m \geq 2$, $\gamma(T_{m,0}) = \gamma(T_{m,1}) = 1$ and $\gamma(T_{m,2}) = m$.*

  For $h \geq 3$, $\gamma(T_{m,h}) = m^{h-1} + \gamma(T_{m,h-3}) = \sum_{i=0}^{\lfloor \frac{h}{3} \rfloor - 1} m^{h-3i-1} + m^{\alpha}$, where $\alpha = 0$ for $h \equiv 0, 1 \pmod 3$ and $\alpha = 1$ for $h \equiv 2 \pmod 3$.

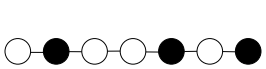

Figure 2.1: P_7 : A dominating set is shaded in black.

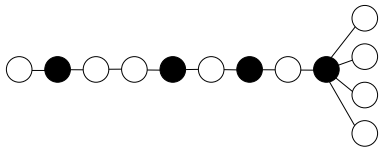

Figure 2.2: $F_{13,4}$: A dominating set is shaded in black.

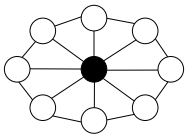

Figure 2.3: W_8 : Single dominating vertex is shaded in black.

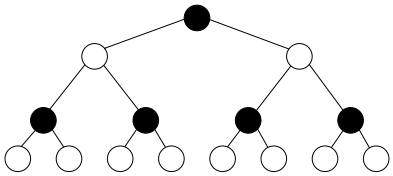

Figure 2.4: $T_{2,3}$: A dominating set is shaded in black.

Chapter 3

Offline Domination: A Brief Survey

In this chapter we survey some results on domination from an algorithmic perspective. More specifically, we consider the problem from the point of view of offline algorithms. In the first section, we define the decision problem DOMINATING SET and show that it is NP-complete. Based on the extensive research literature, we note that, for many restricted classes of graphs, the problem remains NP-complete. In the section that follows we consider the problem of finding a minimum dominating set (for any input graph G) as an optimization problem. Given that the respective decision problem is NP-complete, a polynomial time algorithm that determines the exact value of $\gamma(G)$ for general graphs does not exist unless $P = NP$.

We therefore survey some restricted classes that admit polynomial time algorithms. The third section introduces the notion of a polynomial-time approximation algorithm and describes some known approximation results for the minimum dominating set problem. The final section concludes the chapter by discussing some exact exponential time algorithms.

3.1 NP-Completeness

From the viewpoint of an algorithm designer for the dominating set problem, the primary challenge is to design an algorithm that takes a graph $G = (V, E)$ as input and returns as output an integer that corresponds to $\gamma(G)$. After some time struggling to find an efficient algorithm for this task, one might suspect that the problem is NP-hard. Although the theory of NP-completeness

is reserved for *decision problems*, e.g., those problems whose answer is a simple "yes" or "no", translating an optimization problem into a decision problem is usually straightforward. For the dominating set problem, the algorithm designer instead designs an algorithm that takes a graph G and integer k as input and returns as output a "yes" if $\gamma(G) \leq k$ and a "no" otherwise (i.e. when $\gamma(G) > k$). It is clear that an optimization algorithm can be used to solve this decision problem and conversely, an algorithm for this decision problem can be used to find $\gamma(G)$ by successively applying the decision algorithm at most $|V|$ times. That is, since we have $1 \leq \gamma(G) \leq |V|$ for any graph G, using an algorithm that decides whether $\gamma(G) \leq k$ one simply asks $\gamma(G) \leq i$ for each $1 \leq i \leq |V|$ and finds the smallest j such that $\gamma(G) \leq j$ [1]. In short, restricting our attention to decision problems is appropriate, even for the algorithm designer concerned with the optimization problem. The interested reader can refer to Garey and Johnson (1979) for a thorough treatment of the theory of NP-completeness. We consider the dominating set problem as a decision problem, using the format introduced therein.

DOMINATING SET

INSTANCE : A graph $G = (V, E)$ and a positive integer k.

QUESTION : Does G have a dominating set of size $\leq k$?

We start by stating a fundamental complexity result originally attributed to Garey and Johnson (1979).

Theorem 3.1.1. *Garey and Johnson (1979) DOMINATING SET is NP-complete.*

The authors propose a reduction from the well-known NP-complete problem VERTEX COVER. Given a graph $G = (V, E)$, a subset $C \subseteq V$ is called a *vertex cover* if every edge has at least one endpoint in C. That is, for all $\{u, v\} \in E$ we have that $u \in C$ or $v \in C$ (or both). In the VERTEX COVER problem you are given a graph G and an integer k and have to decide whether there is a vertex cover of size $\leq k$. The reduction from VERTEX COVER works as follows; Consider an instance (G, k) of VERTEX COVER (assume without loss of generality that G has no isolated vertices) and construct in polynomial time, a supergraph of G which has all the vertices of G along

[1] In fact, with the decision algorithm that answers $\gamma(G) \leq k$, one can use a "binary search" type algorithm here with a runtime overhead of a factor of $O(\log |V|)$

with all the edges of G. In addition, for each edge $\{u, v\}$ in G, the supergraph has a new vertex w which is adjacent to both u and v (and only u and v). This is not to be confused with an elementary subdivision of an edge (see Figure 3.1 for an example). One can show that a vertex cover of size $\leq k$ exists in G if and only if a dominating set of size $\leq k$ exists in the resulting supergraph of G.

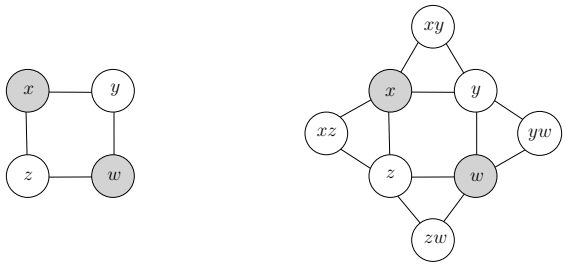

Figure 3.1: An example of the reduction described in Theorem 3.1.1. The vertices shaded gray are respectively, a vertex cover in the left picture and a dominating set in the right picture.

Karp (1972) introduced the so-called SET COVER problem as one of his 21 NP-complete problems. An instance to SET COVER is a finite set $\mathcal{U}$ (sometimes referred to as the *ground set*), a finite family $\mathcal{S}$ of subsets of $\mathcal{U}$ whose union equals $\mathcal{U}$, and an integer k. One must decide whether there is a *cover* of U with size $\leq k$, that is, whether there is a subfamily $C \subseteq \mathcal{S}$ with $\mathcal{U} = \bigcup_{X \in C} X$ and $|C| \leq k$. There is a tight relationship between SET COVER and DOMINATING SET and many of the results surrounding the two are interchangeable.

A reduction from DOMINATING SET to SET COVER is straightforward. Consider an instance (G, k) and set $\mathcal{U} = V$ and $\mathcal{S} = \{N[v] \mid v \in V\}$. That is, the vertices are the elements to be covered and the closed neighborhoods of vertices are those sets which can cover them. It is clear that D is a dominating set of G if and only if $S_D = \{N[v] \mid v \in D\}$ is a cover of $\mathcal{U}$.

Reducing SET COVER to DOMINATING SET requires slightly more care. Given an instance $(\mathcal{U}, \mathcal{S})$ for SET COVER, one constructs a graph $G_{\mathcal{U},\mathcal{S}} = (\mathcal{U} \cup \mathcal{S}, E)$ where $E = \{\{x, S\} \mid x \in S\} \bigcup \{\{X, X'\} \mid X \neq X' \in \mathcal{S}\}$. That is, a graph which has all the elements of $\mathcal{U}$ and all the sets in $\mathcal{S}$ as vertices, an edge between a set and all the elements that belong to that set, and an edge between each pair of sets. One can show that there is a cover of $\mathcal{U}$ with size $\leq k$ if and only if there is a dominating set of $G_{\mathcal{U},\mathcal{S}}$ with size $\leq k$. If $n = |\mathcal{U}|$ and $m = |\mathcal{S}|$ then $G_{\mathcal{U},\mathcal{S}}$ is a graph on

$n + m$ vertices. In some instances, m can be exponential in n therefore we caution the reader when translating results between the two problems.

When confronted with an NP-hard graph problem, one might attempt to simplify the problem by restricting the inputs to a particular class of graphs. Depending on the class, it often happens that the problem is no longer NP-hard. Unfortunately, this is not always the case as there are restricted classes where the problem remains NP-hard. The remainder of the section is dedicated to surveying the graph classes for which the DOMINATING SET problem remains NP-hard, and hence NP-complete. For the classes mentioned below, 'hardness' is here to stay.

A subset of vertices is called an *independent set* if no two vertices of the set are adjacent and it is called a *clique* if every pair of vertices are adjacent. A graph $G = (V, E)$ is called *bipartite* if V can be partitioned into two independent sets and is called *split* if V can be partitioned into a clique and an independent set. Although the class of bipartite graphs and that of split graphs are only superficially similar they are still both popular classes to investigate when studying NP-complete graph problems. A proof that DOMINATING SET remains NP-complete for bipartite graphs simultaneously appeared in Chang and Nemhauser (1984) and Bertossi (1984) and one for split graphs appears in both Corneil and Perl (1984) and Bertossi (1984).

A natural approach to dissecting hard graph problems is to impose degree bounds on the inputs, e.g., consider graphs with maximum degree bounded by a constant. A graph G with maximum degree 2 has a rather simple structure; It consists of a disjoint union of path graphs and cycle graphs. A straightforward algorithm for determining $\gamma(G)$ is to find the connected components of G, which are guaranteed to be either P_k or C_k for some $k \geq 1$, and then apply the results of Section 2.2 for each component. Unfortunately for those who love linear algorithms and graphs with small degree, Kikuno, Yoshida, and Kakuda (1980) show that DOMINATING SET is NP-complete on planar graphs of maximum degree 3. The DOMINATING SET problem has all but a couple degrees of freedom, so to speak.

A graph is said to be *chordal* if every cycle of length at least 4 has a chord (an edge that does not lie on the cycle but is incident on two vertices of the cycle). Booth and Johnson (1982) were the first to show that the problem is NP-complete for chordal graphs. In fact, they prove a stronger result,

that the problem is NP-complete even when inputs are restricted to undirected path graphs which are a subclass of the chordal graphs. A simple proof of the result for chordal graphs is provided in McRae (1996).

For some integer $k \geq 1$, the class of k-*trees* is defined recursively as follows : K_k is a k-tree, and if G is a k-tree then the graph H obtained by adding a new vertex v to G such that v has k neighbours and $N[v]$ forms a clique, is a k-tree. When $k = 1$ this is the familiar class of trees and when $k = 2$, this is a proper subclass of series-parallel graphs. If we consider the class of graphs $\{G \mid G\ is\ a\ k\text{-tree}\ for\ some\ k\}$ then we obtain a proper subclass of all graphs. Restricted to this class (i.e. when k is not fixed) the problem still remains NP-complete as shown in Corneil and Keil (1987). A straightforward result is that, for any k, a k-tree is also a chordal graph. Therefore, this can also be seen as strengthening the result on chordal graphs.

3.2 Exact Polynomial Time Algorithms for Restricted Inputs

To know why a problem is difficult, it is also useful to know when it is easy. Moreover, finding the easy version of your problem is just the beginning. Once a version is established as easy, one can meticulously add back difficulty, in hopes of discovering the exact point at which it becomes too difficult to deal with. For our purposes, this means finding a class of graphs which admits a polynomial time algorithm (i.e. the easy versions of the problem) as well as consider superclasses of these so-called easy classes until the problem becomes NP-complete again (i.e. the versions that have become too difficult to deal with). In this section we attempt to do exactly this. As a small caveat, when we discuss polynomial time algorithms we are now referencing the optimization version of the problem, that is, a polynomial time algorithm that determines the exact value of $\gamma(G)$ when given an input graph G. Although, as previously mentioned this immediately implies a polynomial time algorithm for the decision variant.

Periodically, we provide a chain of class inclusions with the intent of pointing out the area where the problem becomes NP-complete again. A [$\mathbf{P}$] or [$\mathbf{NPc}$] following the class is taken to mean that the class admits a polynomial-time algorithm or it remains NP-complete, respectively. If we assume that $P \neq NP$ then this disjunction of [$\mathbf{P}$] or [$\mathbf{NPc}$] is an exclusive one, e.g., a class cannot both be

NP-complete and admit a polynomial time algorithm.

Since most problems that are NP-complete on general graphs happen to be solvable in polynomial time on trees, the class of trees are considered the "training wheels" for a graph problem. An algorithm designer can start to understand the problem by designing an efficient algorithm that works for trees, which usually exists, and from there they can slowly add structure back and find the problem's breaking point. Unsurprisingly, a linear algorithm to find a minimum dominating set in a tree exists as shown by Cockayne, Goodman, and Hedetniemi (1975). This algorithm is generalized in Hedetniemi, Laskar, and Pfaff (1986) so that it applies to *cactus graphs*, that is, those graphs in which every edge belongs to at most one cycle. A graph $G = (V, E)$ is said to be *biconnected* if it is connected and the removal of any vertex (and all of its incident edges) results in a graph that is still connected. A *block* (*biconnected component*) of G is a maximal biconnected subgraph of G. Given an integer $k \geq 0$, a graph G is said to be *almost-tree*(k) if it is connected and every block of G can be made into a tree by removing no more than k edges. For $k = 0$ and $k = 1$, we obtain the class of trees and cactus graphs, respectively. An $O(m4^{k/2})$ algorithm for almost-tree(k) graphs is given in Gurevich, Stockmeyer, and Vishkin (1984), therefore this algorithm runs in polynomial time for fixed k. Remarking that any connected graph is almost-tree(k) for some k then the problem clearly remains NP-complete when k is not fixed. We recap the contents thus far in this chain of class inclusions.

- trees[**P**] $\subset$ cactus graphs [**P**] $\subset$ almost-tree(k), fixed k [**P**] $\subset$ almost-tree(k), arbitrary k [**NPc**]

As we saw earlier, when k is unbounded, DOMINATING SET remains NP-complete for the class of k-trees. Although, Corneil and Keil (1987) provide an algorithm, which takes a k-tree G as input, and finds a minimum dominating set of G in $O(n^{k+3}2^{2k+2})$. Thus, when k is fixed this class admits an exact polynomial time algorithm. Recalling that, for any k, a k-tree is a chordal graph and that the problem is still NP-complete for chordal graphs we have the following chain.

- trees[**P**] $\subset$ k-trees, fixed k [**P**] $\subset$ k-trees, arbitrary k [**NPc**] $\subset$ chordal [**NPc**]

A graph is called a *block* graph (or *clique tree*) if every block of the graph is a clique. An equivalent characterization for block graphs is that they are the vertex intersection graphs of blocks in a graph. That is, G is a block graph if there is some graph H such that the set of blocks in H is the vertex set of G and two vertices in G are adjacent if the corresponding blocks in G share a common vertex. A graph is called an *undirected path graph* if it is the vertex intersection graph of paths in a tree (i.e. there is some tree T such that the set of paths in T are the vertices and two vertices are adjacent if the corresponding paths in T share a common vertex). A graph is called a *directed path graph* if it is the vertex intersection graph of directed paths in a directed tree.[2] Within Booth and Johnson (1982), where the problem is shown to be NP-complete for undirected path graphs, a polynomial time algorithm is given for the class of directed path graphs giving us the following.

- trees[$\mathbf{P}$] $\subset$ block graphs [$\mathbf{P}$] $\subset$ directed path graphs [$\mathbf{P}$] $\subset$ undirected path graphs [$\mathbf{NPc}$] $\subset$ chordal graphs [$\mathbf{NPc}$]

A two-terminal graph (G, s, t) is a graph G with two distinguished vertices s, called a source, and t, called a sink. For a pair of two-terminal graphs (G_1, s_1, t_1) and (G_2, s_2, t_2), there are two composition operations:

- *Parallel composition*: take a disjoint union of G_1 with G_2 and merge s_1 with s_2 to get the new source, as well as t_1 with t_2 to get the new sink.
- *Series composition*: take a disjoint union of G_1 with G_2 and merge t_1 with s_2, which now becomes an inner vertex of the resulting two-terminal graph; s_1 becomes the new source and t_2 becomes the new sink.

A two-terminal series-parallel graph is a two-terminal graph that can be obtained by starting with several copies of the K_2 graph and applying a sequence of parallel and series compositions. Lastly, a graph is called *series-parallel* if it is a two-terminal series-parallel graph for some choice of source and sink vertices. The series-parallel graphs are a popular graph class when studying graph problems. A linear algorithm for series-parallel graphs first appeared in Kikuno, Yoshida, and

[2]There is a notion of a directed graph $G = (V, E)$ for which E consists of ordered pairs (u, v) of vertices. Many of the definitions for undirected graphs are analogous for directed graphs. A directed graph is called a directed tree if the underlying undirected graph (i.e. the graph resulting from relaxing the direction on each edge) is a tree.

Kakuda (1983). A graph is said to be a partial k-tree if it is a subgraph of a k-tree. In Arnborg and Proskurowski (1989), an algorithm that is polynomial in the size of the graph but superexponential in k is given. At the time of publication, the authors considered the algorithm only feasible for values of $k \leq 8$. Nonetheless, this yields a polynomial time algorithm for fixed values of k. Although the series-parallel graphs do not generalize trees, the *generalized series-parallel* graphs, which are a subclass of the partial 2-trees, contain both the series-parallel graphs and trees among other notable classes. Hence, this result is more general than both the results on trees and series-parallel graphs. Another chain of inclusions is in order.

- trees[**P**] $\subset$ generalized series-parallel graphs [**P**] $\subset$ partial 2-trees [**P**] $\subset$ partial k-trees, fixed k [**P**]

The class of *cographs* can be defined recursively as follows: K_1 is a cograph, and (1) if G is a cograph then the graph complement $\overline{G}$ is a cograph and (2) if G and H are cographs then the disjoint union of G and H is a cograph. It is straightforward to show that any connected cograph has a dominating set of size 1 or 2, hence a polynomial time algorithm for cographs is obvious. This simple observation is originally stated in Corneil and Perl (1984). The cographs, like trees, make for a nice starting class of investigation. The class of k-CUBs are an interesting superclass of cographs which arise by extending the recursive definition given above. Corneil and Stewart (1990) provide a polynomial time algorithm for the class of 1-CUBs, as well as show that the problem is NP-complete for k-CUBs when $k \geq 2$.

- cographs [**P**] $\subset$ 1-CUBs [**P**] $\subset$ 2-CUBs [**NPc**] $\subset$ CUBs [**NPc**]

3.3 Approximation Algorithms

An approximation algorithm yields a solution that is not necessarily optimal but can be guaranteed to lie within some range of an optimal solution. For our purposes, this is an algorithm that returns a dominating set that is not necessarily of minimum size but is guaranteed to be no larger than some function of the minimum size. Typically, the sacrifice of solution quality is traded for

time (as well as simplicity). Under the assumption that $P \neq NP$, this trade-off is a necessary evil as there can be no polynomial time algorithms that return exact solutions for our problem.

Let ALG be an algorithm for the dominating set problem. That is, on any graph G, ALG returns a dominating set for G and we denote this dominating set by $ALG(G)$. We say that ALG achieves *approximation ratio* c if for every input graph G, we have $|ALG(G)| \leq c \cdot \gamma(G)$. The value c need not be a constant and can often be a function of the input size.

For general graphs, a greedy-like algorithm, which at each iteration selects the vertex v with the maximum number of undominated vertices in $N[v]$, achieves an approximation ratio of $H_{\Delta+1}$, where Δ is the maximum degree for the input graph and $H_n = \sum_{i=1}^{n} \frac{1}{i}$ is the n'th *harmonic number*. A straightforward result is that $\ln(n+1) < H_n \leq \ln n + 1$ and therefore the greedy algorithm achieves approximation ratio $\Theta(\ln \Delta)$. This result is originally attributed to Johnson (1974), although the algorithm and analysis is given with respect to the set cover optimization problem. Duh and Fürer (1997) consider the set cover problem where each set in the family has at most k elements and provide an algorithm that achieves approximation ratio $H_{k+1} - \frac{1}{2}$. If we recall the reduction from DOMINATING SET to SET COVER we see that this implies an approximation ratio of $H_{\Delta+1} - \frac{1}{2}$ for the dominating set problem.

With respect to the set cover optimization problem Lund and Yannakakis (1994) show that if there is a polynomial-time approximation algorithm that achieves approximation ratio $c \log n$ for $c < 1/4$ then $NP \subset DTIME(n^{O(poly \log n)})$ (where n denotes the number of elements in the ground set). Feige (1998) strengthen these results by showing that if there is some $\epsilon > 0$ such that a polynomial time algorithm achieves approximation ratio $(1-\epsilon)\ln n$ then $NP \subset DTIME(n^{O(\log \log n)})$. Using the SET COVER to DOMINATING SET reduction we described earlier, it is not immediately obvious that this hardness of approximation results carry over to the dominating set problem. For example, if an instance $(\mathcal{U}, \mathcal{S})$ of SET COVER with $n = |\mathcal{U}|$ and $m = |\mathcal{S}|$ satisfies $m = \omega(n)$ then the construction yields a graph on $|N| = n + m = \omega(n)$ vertices. Within Lund and Yannakakis (1994), the authors show that the hardness result applies even for set cover instances where m is linear in n. Therefore the hardness results do in fact apply for the dominating set problem. Since $H_{\Delta+1} = \Theta(\ln \Delta)$, and Δ can be $\Theta(n)$ for arbitrary graphs, it follows that

the greedy-like algorithm achieves the best possible approximation ratio (ignoring low-order terms) under plausible complexity assumptions.

When inputs have maximum degree bounded by a constant then the approximation ratio $H_{\Delta+1} - \frac{1}{2}$ is a constant. Hence, the problem is in APX. Alimonti and Kann (1997) show that the problem is in fact APX-complete even for inputs with maximum degree 3.

3.4 Exact Exponential Time Algorithms

Certain applications call for exact solutions, regardless of the cost to be payed in running time. The area of designing exact, but exponential, algorithms for the minimum dominating set problem has seen some growth in recent years and here we give a brief overview of some of the work done in this area.

Given a graph $G = (V, E)$ on n vertices, a naive brute force approach is to determine whether S is a dominating set for each of the $2^n - 1$ non-empty subsets $S \subseteq V$, and selects the smallest such S. This gives rise to an $\Omega(2^n)$ algorithm. The first exponential algorithm with running time c^n for $c < 2$ appeared in Fomin, Kratsch, and Woeginger (2004). In said paper, an algorithm for arbitrary graphs with complexity $O(1.93782^n)$ and when inputs are restricted to split graphs, bipartite graphs, and graphs with maximum degree 3, algorithms are given with time complexities of $O(1.41422^n)$, $O(1.73206^n)$, and $O(1.51433^n)$, respectively. Shortly thereafter, Grandoni (2006) independently discovered an $O(1.8021^n)$ algorithm for arbitrary graphs and Fomin, Grandoni, and Kratsch (2005) gave one with running time of $O(1.5263^n)$. To date, the fastest known algorithm for arbitrary graphs is given in Van Rooij and Bodlaender (2011) with a running time of $O(1.4969^n)$

Chapter 4

Online Domination

In the offline setting, we often interpret hardness to mean that the problem inherently requires an obstructive amount of time. In light of the preceding chapter, we see that the minimum dominating set problem is hard in this sense. In an online setting, where information is withheld from an algorithm but other computational resources such as time and space are not of interest, hardness is interpreted to mean that the problem inherently requires an obstructive amount of information. In this chapter we study the dominating set problem in the online setting, showing that it is also a hard problem in this sense. The contents of this chapter is a collaborative work of the author which can be found in the following; Harutyunyan, Pankratov, and Racicot (2021).

Consider a setting where a graph is revealed one vertex at a time. When a vertex is revealed its entire neighborhood is revealed as well. An algorithm is required to make an irrevocable decision on whether to include the newly revealed vertex into the dominating set the algorithm is constructing or not. This decision must be made before the next vertex is revealed. Settings may differ on how strict the condition of irrevocability is and in the amount of neighborhood information provided, e.g., a vertex might be revealed with all of its neighbors or a restricted subset. The performance of an online algorithm is measured against an optimal offline algorithm, i.e., an algorithm that knows the entire input in advance and has infinite computational resources. This measure is captured by the notion of competitive ratio and analysis.

We provide a brief overview of competitive analysis framework. For more details, an interested reader should consult excellent books Borodin and El-Yaniv (1998); Komm (2016) and references

therein. Let ALG be an algorithm for the online dominating set problem. Let $ALG(G, \sigma)$ denote the set of vertices that are selected by ALG on the input graph G with its vertices revealed according to the order σ. We sometimes abuse the notation and omit G or σ (or both) when they are clear from the context. Abusing notation even more, we sometimes write $ALG(G, \sigma)$ to mean $|ALG(G, \sigma)|$. Similar conventions apply to an offline optimal solution denoted by OPT. We say that ALG achieves *strict competitive ratio* c if $ALG \leq c \cdot OPT$ on all inputs. We say that ALG achieves *asymptotic competitive ratio* c (or, alternatively, that ALG is c-competitive) if $\limsup_{OPT \to \infty} \frac{ALG}{OPT} \leq c$. The *competitive ratio* of ALG is the infimum over all c such that ALG is c-competitive. When we simply write "competitive ratio" we typically mean "asymptotic competitive ratio" unless stated otherwise.

Online dominating set problem has been studied in the vertex arrival model by Boyar et al. (2019). In that model, when a vertex is revealed only a restricted neighborhood of that vertex is revealed as well, namely, those neighbors that appear among previously revealed vertices. Moreover, in the model considered by Boyar et al. decisions are only partially irrevocable, i.e., when a vertex arrives an algorithm may add this vertex together with *any of its neighbors from the restricted neighborhood* to the dominating set. Thus, the decision to include a vertex is irrevocable, while the decision not to include a vertex is only partially irrevocable – an algorithm has a chance to reconsider when any yet unrevealed neighbors arrive. The catch is that the algorithm does not know the input size and must maintain a dominating set of the revealed part at all times. In the model considered in this chapter, all decisions (to include or exclude a vertex from a dominating set) are irrevocable. Boyar et al. considered the online dominating set problem in two settings, namely, with the restriction of an adversary being forced to maintain an always connected graph and without this restriction. For the fairness of comparison, when we talk about Boyar et al. results we refer to their results for the always-connected setting[1].On one hand, this makes our model stronger for the adversary. On another hand, our model is weaker for the adversary than the model of Boyar et al. in the aspect of the adversary being forced to reveal all neighbors of a newly revealed vertex at once.

[1]In our model, two natural definitions of always-connected restriction are possible: (i) with respect to all vertices that the algorithm is aware of at any particular moment (this includes vertices that have arrived and their neighbors that have not yet arrived), and (ii) with respect to only those vertices that have arrived. Our work is in setting (ii). This distinction is absent in the vertex arrival model.

Thus, our results when compared to those of the vertex arrival model can be viewed as quantifying the value of getting to know all neighbors of a vertex at the time of its revelation.

As a brief aside, we remark that a vertex arrival model (with restricted neighborhoods) and strict irrevocability is rather hopeless for an online algorithm. Indeed, consider revealing each vertex adjacent to all those previously revealed vertices as long as an algorithm continues to select vertices. If the algorithm selects these vertices indefinitely we obtain an input with which all n vertices are selected. If the algorithm decides not to select a vertex, then all vertices revealed thereafter are adjacent only to this "excluded" vertex and an algorithm is forced to select each of these vertice revealed after ultimately yielding an output of $n-1$ vertices. In either of these cases, only 1 vertex needs to be selected for an optimal solution. Similar remarks were mentioned in King and Tzeng (1997) and this restrictive model was further explored in Böckenhauer et al. (2021), although with advice.

Perhaps somewhat surprisingly, we discover in several results that the benefit of knowing all neighbors outweighs the drawbacks of fully irrevocable decisions. Our results are summarized below, but in particular we show that in our model Δ-bounded degree graphs admit $O(\sqrt{\Delta})$ online algorithms, while Boyar et al. show that $\Omega(\Delta)$ is necessary in their model. Similarly, we demonstrate and analyze a 2-competitive algorithm for trees, while Kobayashi (2017) shows a lower bound of 3 in the vertex arrival model. Our degree upper bound implies that $O(\sqrt{n})$ competitive ratio is tight for general graphs, whereas Boyar et al. showed the lower bound of $\Omega(n)$ in the vertex arrival model. This paints a picture that knowing all the neighbors improves not only precise constants, when graph classes allow for small competitive ratio algorithms, but also give asymptotic improvements for more "challenging" graph classes for algorithms.

We shall consider performance of algorithms with respect to restricted inputs, specified by various graph classes, such as trees, cactus graphs, series-parallel, etc. The above definitions of competitive ratios can be modified by restricting them to inputs coming from certain graph classes. We denote the competitive ratio of an algorithm ALG with respect to the restricted graph class CLASS by $\rho(ALG,\text{CLASS})$.

The following is a summary of our contributions with the section numbers where the results appear.

- tight competitive ratio 2 on trees (Section 4.2.1);
- tight competitive ratio $\frac{5}{2}$ on cactus graphs (Section 4.2.2);
- tight competitive ratio $\Theta(\sqrt{\Delta})$ on maximum degree Δ graphs (Section 4.2.3);
- tight competitive ratio $t-1$ on $K_{1,t}$-free graphs (Section 4.2.4);
- tight competitive ratio $\Theta(\sqrt{n})$ for threshold graphs (Section 4.3.1), planar bipartite graphs (Section 4.3.2), and series-parallel graphs (Section 4.3.3).

We note that all our upper bounds are in terms of strict competitive ratios, and all our lower bounds, with the exception of $K_{1,t}$-free graphs, are in terms of asymptotic competitive ratios.[2]

The remainder of the chapter is split into three main sections. Section 4.1 describes definitions and establish notations that is used frequently throughout the chapter. Section 4.2 consists of graph classes which admit algorithms with competitive ratio that is independent of the input size whereas Section 4.3 consists of those which do not.

4.1 Preliminaries

Let $G = (V, E)$ be a connected graph on $n = |V| \geq 1$ vertices. The vertices of V are revealed online in order $(v_1, ..., v_n)$. Since we consider the online input model where vertices are revealed alongside their neighbors, we distinguish between two notions: those vertices that are revealed by a certain time and those that are visible. More precisely, we have the following:

Definition 4.1.1.
- v_i *is* revealed by time j *if* $i \leq j$.
- v_j *is* visible at time i *if it is either revealed by time* i *or it is adjacent to some vertex revealed by time* i.
- R_i *denotes the set of all vertices revealed by time* i.
- $V_i = N[R_i]$ *denote the vertices visible at time* i.

[2]With the small caveat that the performance ratio for threshold graphs is measured as a function of input size for reasons provided later.

The adversary chooses the graph G as well as the revelation order of vertices; however, the adversary is restricted to those revelation orders that guarantee that $\langle R_i \rangle$ is connected for all i. Thus, we observe that the process of revelation of a graph by the adversary is a natural generalization of the breadth-first search (BFS) and depth-first search (DFS) explorations of the graph. Thus, we can define the *revelation tree* analogous to BFS and DFS trees. We need the following observation first:

Observation 4.1.1. *If $v_j \in V_i \setminus V_{i-1}$ with $i \geq 2$ then v_i is the unique neighbour of v_j at time i.*

In the preceding observation, we say that v_j is a *child* of v_i and that v_i is the *parent* of v_j. The edge $\{v_i, v_j\}$ is called a *tree edge*. The subgraph induced on the tree edges is the revelation tree. Any edge $\{u, v\}$ where u is not the parent of v nor v the parent of u is called a *cross edge*.

After the vertex v_i is revealed together with its closed neighborhood $N[v_i]$, an online algorithm ALG must make a decision $d_i \in \{0, 1\}$, which indicates whether the algorithm takes this vertex to be in the dominating set or not.

Definition 4.1.2. *Given an online algorithm ALG we let:*

- $S_i = \{v_k \mid d_k = 1, 1 \leq k \leq i\}$ *denote the set of revealed vertices selected by ALG after decision d_i where $S_0 = \emptyset$.*
- $D_i = N[S_i]$ *denote the set of vertices that are dominated after decision i.*
- $U_i = V_i \setminus D_{i-1}$ *denote the set of visible vertices undominated immediately before decision d_i where $U_0 = \emptyset$.*

A series of figures are provided below which illustrate the preceding definitions. For these figures, and all others in this chapter, the convention is that vertices that are shaded in gray are those selected by ALG, vertices with thicker boundaries belong to OPT, an edge that is dashed is a cross edge, and all the solid edges are tree edges.

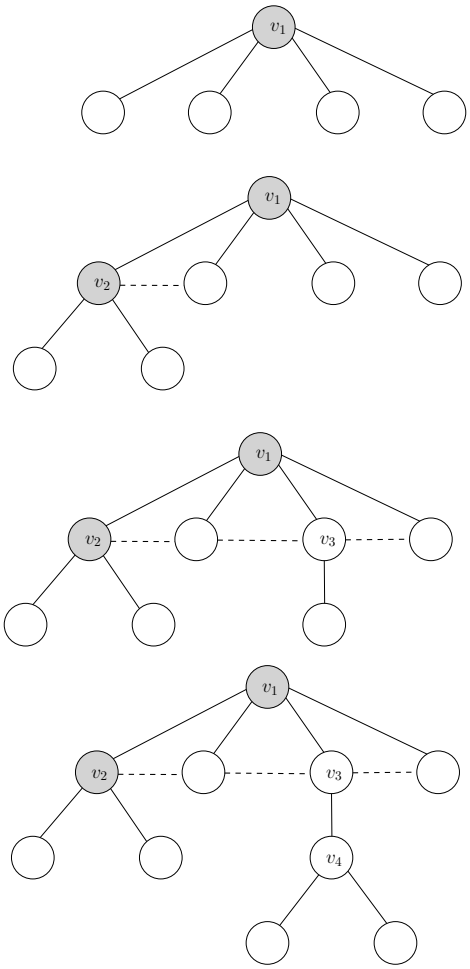

Figure 4.1: A series of prefixes of the graph depicted in Figure 4.2

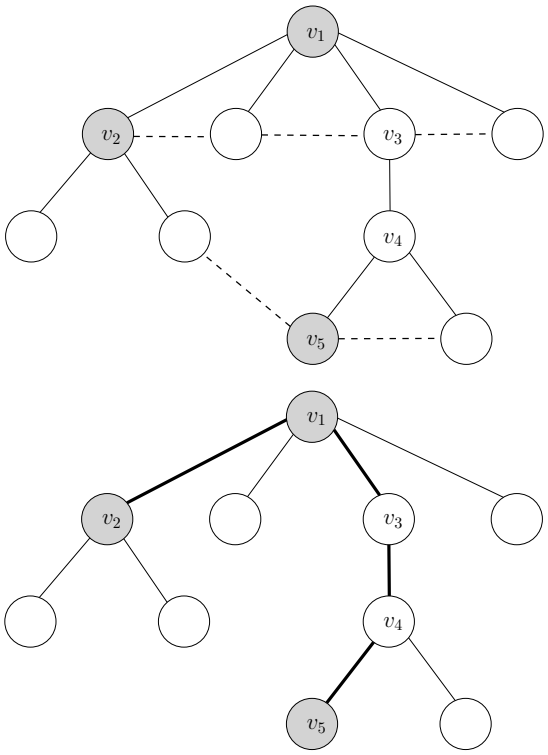

Figure 4.2: Top picture depicts the prefix of a graph with 5 revealed vertices and 10 visible vertices. Bottom picture depicts its revelation tree. The thickened edges illustrate that the induced subgraph on the revealed vertices is guaranteed to be connected.

Since an online algorithm makes irrevocable decisions and it must produce a feasible solution, there may be situations where an algorithm is forced to select a vertex v_j to be in the dominating set. This happens because v_j is the "last chance" to dominate some other vertex v_i. In this case, we say that v_j *saves* v_i or that v_j is the *savior* of v_i. Note that it is possible for a vertex v_j to save itself. The following definition makes the notion of "saving" precise.

Definition 4.1.3. *A vertex* $v_j, j \geq 1$ saves *a vertex* v_i *if* $j = \max\{k \mid v_k \in N[v_i]\}$ *and* $N[v_i] \setminus \{v_j\}$ *contains no vertices from* S_{j-1}*. Let* $s(v_j)$ *denote the set of vertices that* v_j *saves.*

Observe that if a vertex is saved then it must be that every one of its neighbours (itself included) had a chance to dominate the said vertex.

Observation 4.1.2. *If* v_i *is saved then* $v_i \in N[v_j] \cap U_j$ *for any* $v_j \in N[v_i]$.

All our upper bounds are established by either a GREEDY algorithm or a k-DOMINATE algorithm for some fixed integer value of parameter k:

- The algorithm GREEDY selects a newly revealed vertex if and only if the vertex is not currently dominated. Using the notation introduced above, GREEDY selects $v_i, i \geq 1$ if and only if $v_i \in U_i$.
- The algorithm k-DOMINATE (for some fixed integer parameter k) selects a newly revealed vertex if and only if either (1) the vertex has at least k undominated neighbors, or (2) the vertex saves at least one other vertex. Using the notation introduced above, $v_i, i \geq 1$ is selected if and only if either (1) $|N(v_i) \cap U_i| \geq k$, or (2) $|s(v_i)| \geq 1$.

Both GREEDY and k-DOMINATE give rise to rather efficient offline algorithms so that any of the positive results given in this paper may be realized as efficient offline approximation algorithms.

4.2 Competitive Graph Classes

4.2.1 Trees

In this section we establish the tight bound of 2 on the best competitive ratio when the input graph is restricted to be a tree. The upper bound is achieved by the 2-DOMINATE algorithm and is proved in Theorem 4.2.2 below. The lower bound on all online algorithms is established in Theorem 4.2.1. We begin this section with the lower bound.

Theorem 4.2.1. *$\rho(ALG, TREE) \geq 2$ for any algorithm ALG.*

Proof. Consider an arbitrary small $\epsilon > 0$. We will give an adversarial input that guarantees that $ALG \geq (2 - \epsilon)OPT$. Let $k = \lceil \frac{3}{\epsilon} \rceil \geq 4$. At the start, the adversary reveals v_1 with k children $\{c_1, \ldots, c_k\}$. Then we start the process described in the next paragraph at c_1. The process can terminate in two ways: (i) ALG stops selecting vertices to be in the dominating set, or (ii) ALG selects k vertices revealed after c_1 (inclusive). If the process terminates because of (i), then the adversary restarts the process at child c_2 of v_1. The process again terminates either with (i) or (ii) with respect to c_2. If it is due to (i), then the adversary restarts the process at c_3, and so on. If the process terminates with (ii) with respect to c_i then we reveal c_j for $j > i$ as leaves of v_1.

Next, we describe the process with respect to c_i. The adversary reveals c_i with 2 children and if ALG selects c_i then exactly one child of c_i is revealed with two additional children. If ALG selects the child then one of its children is revealed with two additional children, and so on. Let j_i be the number of these vertices that are selected by ALG. This process terminates only if ALG stops selecting these vertices with two children ($j_i < k$) or when ALG selects k of them ($j_i = k$). At this point the subtree grown at c_i has some revealed vertices as well as visible, but not yet revealed vertices. To finish revealing the entire subtree, the adversary proceeds as follows.

If $j_i < k$ then the two children on the $(j_i + 1)$'st vertex are revealed to be leaves. Moreover, each of the j_i selected vertices have exactly one visible child that is not yet revealed. Reveal those j_i children, called *support vertices*, with an additional leaf child (i.e. the child is revealed to be a leaf after its parent is revealed). Including the 2 children of the $(j_i + 1)$'st vertex ALG must select at least $j_i + 2$ additional vertices to dominate these leaves for a total of $j_i + (j_i + 2) = 2(j_i + 1)$ selected vertices in this subtree. In this case, OPT can select the support vertices together with the $(j_i + 1)$'st vertex for a total $j_i + 1$ vertices to dominate the entire subtree.

If $j_i = k$ the procedure to finish revealing the entire subtree at c_i is similar: the k'th vertex children are both revealed to be leaves and each of the other $k - 1$ selected vertices has the other child become a support vertex, i.e., revealed with an additional leaf child. The performance is similar here but ALG is not forced to select the two children of the k'th vertex so ALG selects at least $k + (k - 1) = 2k - 1$. In this case, OPT needs only select the k'th vertex together with the support vertices for a total of k vertices to dominate the subtree.

To finish the analysis, we consider the following two cases:

Case 1 : for all i we have $j_i < k$. Then $ALG \geq 2(j_i+1)$ on each subtree whereas $OPT \leq j_i+1$ on each subtree. Summing over all subtrees and remarking that OPT might select v_1 we obtain that

$$ALG/OPT \geq \Big(\sum 2(j_i + 1)\Big) \,/\, \Big(1 + \sum (j_i + 1)\Big) \geq 2 - 2/k \geq 2 - \epsilon.$$

Case 2 : there exists ℓ such that $j_\ell = k$. Then OPT selects $j_i + 1$ vertices for $i < \ell$, k vertices for $i = \ell$, 0 vertices for $i > \ell$ per subtree, plus v_1. Whereas ALG selects at least $2(j_i + 1)$ for $i < \ell$, $2k - 1$ for $i = \ell$, and 0 for $i > \ell$. By a similar calculation to **Case 1**, we obtain that

$ALG/OPT \geq 2 - 3/k \geq 2 - \epsilon.$

□

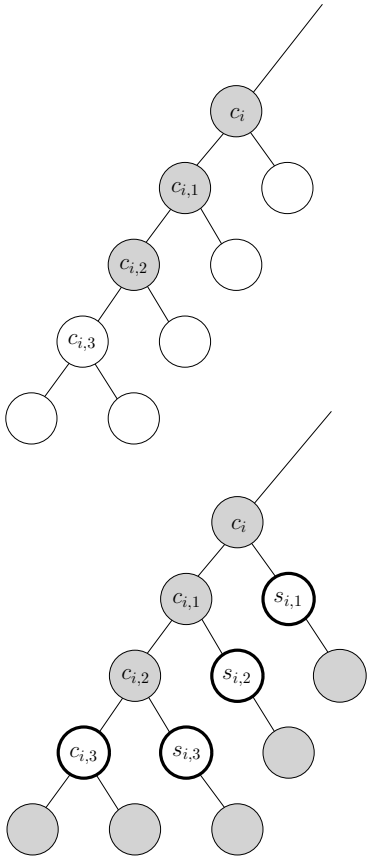

Figure 4.3: An example of the process described in Theorem 4.2.1 where ALG selects $j_i = 3$ vertices on the subtree rooted at c_i. The top depicts the subtree immediately after revealing $c_{i,3}$ whereas the bottom shows the entirely revealed subtree.

Now that we have established an asymptotic lower bound of 2 for any algorithm we show that 2-DOMINATE is 2-competitive.

Theorem 4.2.2. $\rho(\textit{2-DOMINATE, TREE}) = 2$.

High level overview of the proof. Consider an arbitrary input $T = (V, E)$ on $n \geq 3$ vertices and let OPT denote a minimum dominating set of T which contains no vertices of degree 1 (i.e. any such vertex can be exchanged for its only neighbor). Recall that S is the set of vertices selected by 2-DOMINATE. Initially, we assign charge 1 to each vertex v in S and charge 0 to each vertex

v not in S. Thus, $|S| = \sum_{v \in S} ch(v)$ where $ch(v)$ denotes the charge of v. With a charging scheme described shortly, we spread the charge from the vertices in S to the vertices of V. Let $ch^*(v)$ denote the new charge associated with vertex v. We extend the functions ch and ch^* to subsets of vertices linearly, e.g., for $W \subseteq V$ we have $ch(W) = \sum_{v \in W} ch(v)$. We shall demonstrate that the procedure of spreading the charge satisfies two properties:

(1) conservation property: $\sum_v ch(v) = \sum_v ch^*(v)$ meaning that the total charge is preserved; and

(2) OPT-concentration property: for each $v \in OPT$ we have $ch^*(N[v]) \leq 2$.

With these two properties it follows that,

$$|S| \leq \sum_v ch(v) = \sum_v ch^*(v) \leq \sum_{v \in OPT} ch^*(N[v]) \leq 2OPT,$$

so 2-DOMINATE is strictly 2-competitive.

Before we proceed with this plan, we make a couple of useful observations:

Lemma 4.2.1. *If input is a tree, there are no cross edges incident on any v_i. In particular, v_i has at most one neighbour before it is revealed.*

Corollary 4.2.1. *If $deg(v_i) \geq 3$ then $v_i \in S$.*

Now, we are ready to present formal details of the above plan. We spread the charges according to the following rule:

Consider any $v_i \in S$ with $X_i = N[v_i] \cap U_i$. Remarking that $X_i \neq \emptyset$ we then give each vertex in X_i an equal charge of $\frac{1}{|X_i|}$. That is, a vertex selected by 2-DOMINATE spreads its charge evenly to all the newly dominated vertices in its closed neighbourhood. We say that each vertex in X_i is charged by v_i.

Observation 4.2.1. *Every vertex is charged by exactly one vertex.*

The preceding observation immediately implies that any vertex has charge at most 1. This observation is tight in the sense that, on certain inputs, there are vertices with charge equal to 1. A

vertex with charge 1 is a rather special case though. In particular, if v_i has charge 1 then it must be saved by some vertex v_j where $X_j = \{v_i\}$ (this does not exclude the possibility that $v_i = v_j$). If v_i does not meet this condition then it must have charge at most $\frac{1}{2}$.

Lemma 4.2.2. *If v_i and v_j both have charge equal to 1 then they share no common neighbours.*

Proof. Suppose for the sake of deriving a contradiction that $v_{i'}$ were a common neighbour of v_i and v_j. Since v_i is saved, by Observation 4.1.2 it must be that $v_i \in N(v_i') \cap U_{i'}$. Similarly, we have that $v_j \in N(v_i') \cap U_{i'}$. That is, $|N(v_i') \cap U_{i'}| \geq 2$ and thus $v_i' \in S$. Moreover, $X_{i'} = N[v_{i'}] \cap U_{i'}$ contains v_i and v_j. In particular, we have that $|X_{i'}| \geq 2$ with $v_i, v_j \in X_{i'}$ and therefore v_i and v_j receive charge no larger than $\frac{1}{2}$, a contradiction. □

Lemma 4.2.3. *If v_i and v_j both have charge equal to 1 then they are not adjacent.*

Proof. It is easy to see that v_1 cannot have charge 1 on any input with at least 2 vertices. Therefore we safely assume that $1 < i < j$ such that both v_i and v_j have a parent. We assume for the sake of deriving a contradiction that v_i and v_j are adjacent.

Now, since both v_i and v_j have charge 1 it follows that they are both saved vertices. First we show that both $v_i, v_j \notin S$. Notice that any saved vertex v_k has the property that $|N[v_k] \cap S| = 1$. Therefore, if we assume by way of contradiction that $v_i \in S$ we obtain that $N[v_i] \cap S = N[v_j] \cap S = \{v_i\}$ and therefore v_i saves itself and v_j. This yields that $X_i = N[v_i] \cap U_i$ contains v_i and v_j. In particular, we have that $|X_i| \geq 2$ with $v_i, v_j \in X_i$ and therefore v_i and v_j receive charge no larger than $\frac{1}{2}$, a contradiction. An identical argument will yield that $v_j \notin S$.

Therefore it must be that v_i is saved by some vertex $v_{i'}$ with $i' \notin \{i, j\}$. Moreover, we must have $i < j < i'$ since $i < j$ by assumption and $i' = \max\{k \mid v_k \in N[v_i]\}$. This implies that both $v_j, v_{i'}$ are children of v_i by Observation 4.2.1 yielding that $|N(v_i) \cap U_i| \geq 2$ but v_i cannot be in S. □

From the two preceding lemmas we have the immediate corollary.

Corollary 4.2.2. *For any v_i, at most one vertex in $N[v_i]$ has charge 1.*

Now, we finish the proof of 2-competitiveness of 2-DOMINATE on trees.

(Proof of Theorem 4.2.2). The lower bound follows from Theorem 4.2.1. Let $v_i \in OPT$ be an arbitrary vertex in OPT. We consider two cases **(1)** $deg(v_i) = 2$ or **(2)** $deg(v_i) \geq 3$.

Case 1 : Suppose that $deg(v_i) = 2$ and hence $|N[v_i]| = 3$. By Corollary 4.2.2 it follows that at most one vertex in $N[v_i]$ has charge 1. If no vertices in $N[v_i]$ have charge 1 then $ch(x) \leq \frac{1}{2}$ for each $x \in N[v_i]$ and we obtain that $\sum_{x \in N[v_i]} ch(x) \leq 3\left(\frac{1}{2}\right) < 2$. If there is exactly one vertex $x' \in N[v_i]$ with charge 1 we therefore obtain that $\sum_{x \in N[v_i]} ch(x) = \sum_{x \in N[v_i] \setminus \{x'\}} ch(x) + ch(x') \leq \frac{2}{2} + 1 = 2$.

Case 2 : Suppose that $deg(v_i) \geq 3$. By Corollary 4.2.1 it follows that $v_i \in S$ with at least 2 children. Let $C_i = V_i \setminus V_{i-1}$ denote the children of v_i and remark that $C_i \subseteq X_i$. That is, each child of v_i is charged by v_i and only v_i. Therefore the children of v_i can receive at most the full initial charge on v_i and thus attribute a charge of at most 1.

Now we claim that any vertex in $N[v_i] \setminus C_i$ has a charge of at most $\frac{1}{2}$. Indeed, suppose a vertex $v_{i'} \in N[v_i] \setminus C_i$ has charge 1 then it must be saved by v_i since $|N[v_{i'}] \cap S| = 1$ for any saved vertex $v_{i'}$. That is, there is exactly one vertex in its closed neighbourhood that is selected and since v_i is selected it must be v_i. Thus, we must have that $v_{i'} \in X_i$ but since $C_i \subseteq X_i$ we know that $|X_i| \geq 2$ and thus $v_{i'}$ receives a charge of no more than $\frac{1}{2} < 1$, contradicting our assumption that $v_{i'}$ has charge 1.

Thus, by remarking that $|N[v_i] \setminus C_i| \leq 2$ we obtain that $\sum_{x \in N[v_i]} ch(x) = \sum_{v_j \in C_i} ch(v_j) + \sum_{v_{i'} \in N[v_i] \setminus C_i} ch(v_{i'}) \leq 1 + 2\left(\frac{1}{2}\right) = 2$ as desired. □

4.2.2 Cactus Graphs

A graph G is said to be a *cactus graph* if it is connected and every edge belongs to at most one cycle. Hedetniemi et al. (1986) provide an exact offline algorithm that runs in linear time for finding a minimum dominating set of a cactus graph. Of course, an efficient offline algorithm does not guarantee that an online algorithm can perform well but fortunately, cactus graphs are a class of graphs for which an online algorithm can achieve constant competitive ratio. In this section, we show that 2-DOMINATE is $\frac{5}{2}$-competitive when inputs are restricted to cactus graphs, and that this is as well as any algorithm can perform.

Before presenting a lower bound of $\frac{5}{2}$ on all online algorithms we describe a gadget that is used

in the proof. The gadget itself is a cactus graph on $3 \leq n \leq 4$ vertices with the property that OPT selects exactly 1 vertex and any algorithm ALG selects at least 2 vertices. Consider revealing a root vertex r with 2 children c and c'. If ALG does not select r then both c, c' are revealed as only adjacent to r and ALG must select both whereas OPT selects only r. If ALG does select r then c is revealed as adjacent to c', and c' is revealed with an additional child x. The vertex x is adjacent only to c' and thus ALG must select at least one of c', x whereas OPT selects only c' (both cases are depicted in figure 4.4). Given any input cactus graph with a visible vertex r not yet revealed this gadget can be constructed with r as the root. Within the proof of the lower bound we call this a 2-gadget.

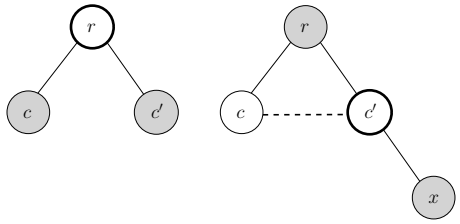

Figure 4.4: The cactus 2-gadget : The leftmost figure depicts the case where ALG does not select the root r and rightmost depicts the case where ALG selects r.

Theorem 4.2.3. *$\rho(ALG, CACTUS) \geq \frac{5}{2}$ for any algorithm ALG.*

Proof. Consider an arbitrary small $\epsilon > 0$ and let $k = \lceil \frac{4}{\epsilon} \rceil \geq 5$. We will give an adversarial input that guarantees that $OPT \geq k$ and $ALG \geq (\frac{5}{2} - \epsilon)OPT$. To begin the input, the adversary reveals v_1 with k children $\{c_1, \ldots, c_k\}$. Then we run an adversarial process starting with the child c_1 of v_1. The process consists of rounds, where each round increases OPT by 2 while increasing ALG by 5. The process might terminate for one of two reasons: either (i) we guarantee strict competitive ratio at least $5/2$ on the subcactus rooted at c_1, or (ii) k rounds starting at c_1 elapse. If the process terminates because of (i), then the adversary restarts the process at child c_2 of v_1. The process again terminates either with (i) or (ii) with respect to c_2. If it is due to (i), then the adversary restarts the process at c_3, and so on. If the process terminates with (ii) with respect to c_i then we reveal c_j for $j > i$ as leaves. Below we describe the process starting at a child of v_1 although the first round of the process differs from the others that follow.

We now describe the first round starting at a child c of v_1. Initially, we reveal c with 3 children. If ALG does not select c then each child of c is revealed as leaf and ALG must select all 3 children whereas OPT selects c. Suppose then that ALG selects c and let $c_{1,1}, c_{1,2}, c_{1,3}$ be the three children of c. Reveal $c_{1,1}$ as adjacent to $c_{1,2}$ along with 2 additional children. If ALG does not select $c_{1,1}$ then the children of $c_{1,1}$ are revealed as leaves, forcing ALG to select them and $c_{1,3}$ is revealed as the root of a 2-gadget ($c_{1,2}$ is revealed with no additional neighbours). Thus, $\frac{ALG}{OPT} \geq \frac{5}{2}$ in this case (see Figure 4.5). If instead ALG selects $c_{1,1}$ then $c_{1,2}$ and $c_{1,3}$ are revealed as the roots of two distinct 2-gadgets and since c is dominated by v_1 (we assume that $v_1 \in OPT$) we have that $\frac{ALG}{OPT} \geq \frac{5}{2}$ on this subcactus (excluding $c_{1,1}$) thus far (see Figure 4.6). At this point, $c_{1,1}$ is selected by ALG and we start the second round (which is described below) with $c_{1,1}$ as the root. Every round that follows will be the same as the second and requires a root selected by ALG which has two children.

The second round starts at a selected root $c_{1,1}$ and we let $c_{2,1}, c_{2,2}$ be the 2 children of $c_{1,1}$. We reveal $c_{2,1}$ as adjacent to $c_{2,2}$ with 2 children $c_{3,1}, c_{3,2}$. If ALG does not select $c_{2,1}$ then $c_{3,1}, c_{3,2}$ are revealed as leaves and ALG selects $c_{1,1}, c_{3,1}, c_{3,2}$ and OPT can select $c_{2,1}$ for a performance of 3 along with the running performance of $\frac{5}{2}$ (see Figure 4.7). If ALG does select $c_{2,1}$ then $c_{3,1}$ is revealed as adjacent to $c_{3,2}$ with two children $c_{4,1}, c_{4,2}$. If ALG does not select $c_{3,1}$ then $c_{4,1}, c_{4,2}$ are revealed as leaves and $c_{2,2}$ is revealed with an additional leaf neighbour $l_{2,2}$ so that ALG must select at least one of $c_{2,2}, l_{2,2}$. Thus, ALG here selects $c_{1,1}, c_{2,1}, c_{4,1}, c_{4,2}$ and at least one of $c_{2,2}, l_{2,2}$ whereas OPT can select $c_{3,1}$ and $c_{2,2}$ for a performance of $\frac{5}{2}$ (see Figure 4.8). If instead ALG selects $c_{3,1}$ (thus far $c_{1,1}, c_{2,1}$ and $c_{3,1}$ are all selected) then $c_{2,2}$ is revealed with an additional leaf neighbour $l_{2,2}$ so that ALG must select at least one of $c_{2,2}, l_{2,2}$, and $c_{3,2}$ is revealed as the root of a 2-gadget so that $\frac{ALG}{OPT} \geq \frac{5}{2}$ on the subcactus thus far (excluding $c_{3,1}$) and we repeat the trap with $c_{3,1}$ as the selected root (see Figure 4.9).

Let $j_i \geq 1$ denote the number of rounds that passed in the adversarial process starting at the child c_i. To finish the analysis, we consider the following two cases:

Case 1 : for all i we have that $j_i < k$. Then $ALG \geq 5j_i$ on each subcactus whereas $OPT \leq 2j_i$ on each subcactus[3]. Summing over all subcacti and remarking that OPT selects v_1 we obtain that

[3]We have omitted the cases where ALG does not select the root c_i. These cases result in ALG selecting 3 vertices

$ALG/OPT \geq (\sum 5j_i) \,/\, (1 + \sum 2j_i) \geq \frac{5}{2} - \frac{5}{2k} \geq \frac{5}{2} - \epsilon$.

Case 2 : there exists ℓ such that $j_\ell = k$. In this case, there is an additional vertex $c_{j,1}$ with $j = 3(k-1)$ that was selected by ALG and must also be selected by OPT. (i.e. c_j is the root where a $(k+1)$'st round could start). Therefore, OPT selects $2j_i$ vertices for each process on child c_i with $i < \ell$, $2k+1$ vertices for $i = \ell$, 0 vertices for $i > \ell$ plus v_1. Whereas ALG selects at least $5j_i$ for $i < \ell$, $5k+1$ for $i = \ell$, and 0 for $i > \ell$. Ultimately, we obtain that $ALG/OPT \geq (\sum 5j_i + 5k + 1) \,/\, (\sum 2j_i + 2k + 2) \geq \frac{5}{2} - 4/k \geq \frac{5}{2} - \epsilon$. □

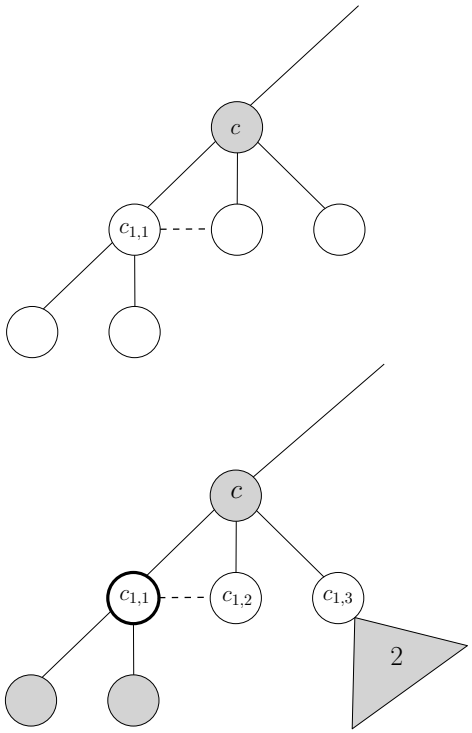

Figure 4.5: The case described in Theorem 4.2.3 where ALG does not select $c_{1,1}$.

on the subcacti with OPT selecting only 1 and the result clearly still holds in this case.

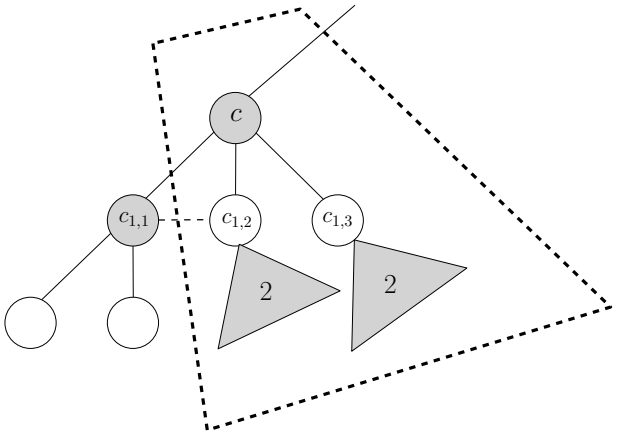

Figure 4.6: The case described in Theorem 4.2.3 where ALG does select $c_{1,1}$. The enclosed region contributes a performance of $\frac{5}{2}$. A trap is continued in this case with the root $c_{1,1}$.

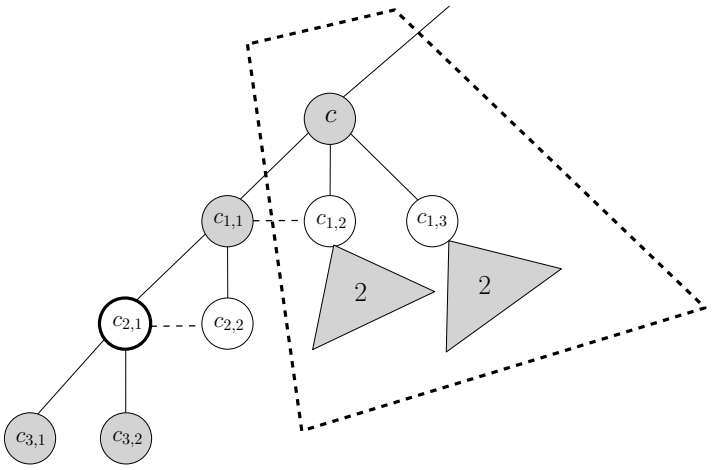

Figure 4.7: The case described in Theorem 4.2.3 where ALG does not select $c_{2,1}$.

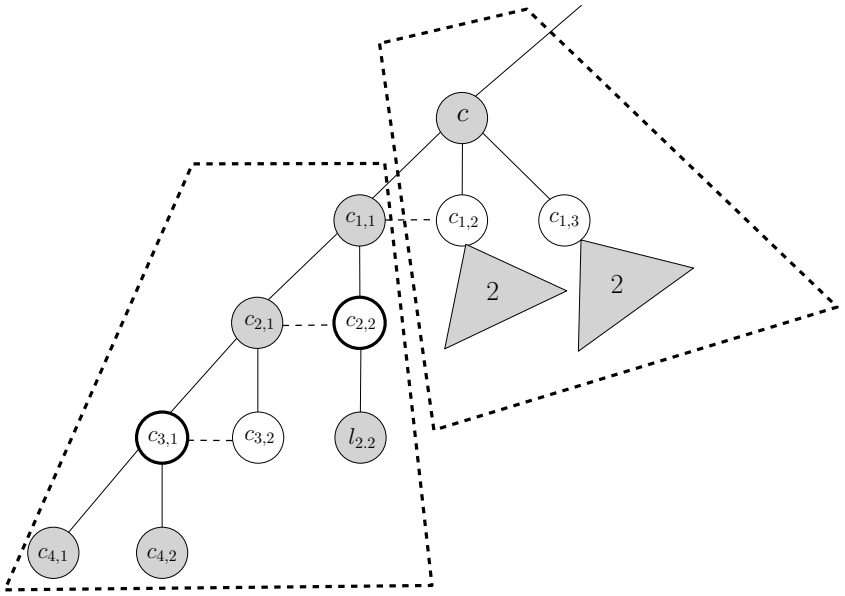

Figure 4.8: The case described in Theorem 4.2.3 where ALG does not select $c_{3,1}$. The enclosed regions each contribute a performance of $\frac{5}{2}$.

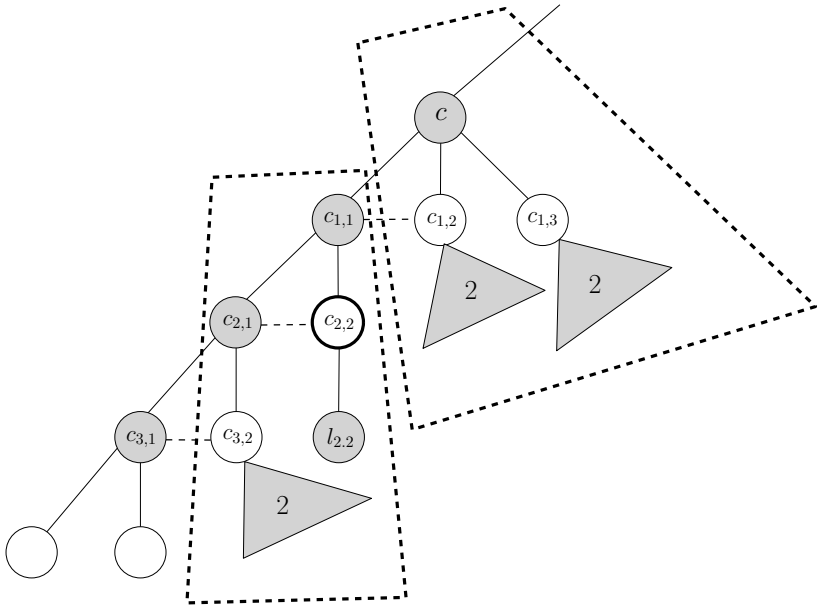

Figure 4.9: The case described in Theorem 4.2.3 where ALG does select $c_{3,1}$. The enclosed regions each contribute a performance of $\frac{5}{2}$. The trap used on a selected root $c_{1,1}$ is repeated on the root $c_{3,1}$.

Theorem 4.2.4. *ρ(2-DOMINATE, CACTUS)* $= \frac{5}{2}$.

The proof can be viewed as an adaptation of our proof for trees to cactus graphs. We use a charging argument similar to the one given in the section on trees. Initially, a charge of 1 is given for each $v \in S$, the charge on each vertex is then spread to certain neighbours, and we then show that $\sum_{x \in N[v_i]} ch(x) \leq \frac{5}{2}$ for each $v_i \in OPT$. We spread the charge according to the same rule given in the preceding section and recall that Observation 4.2.1 (each vertex receives a new charge from one other vertex) still holds. In the analysis of how the charge gets reallocated, the structure of the underlying graph is of paramount importance. We begin with an analogue to Lemma 4.2.1.

Lemma 4.2.4. *In cactus graphs, there is at most one cross edge incident on any v_i. In particular, v_i has at most 2 neighbours before it is revealed.*

Proof. The statement is clearly true for $v_i = v_1$ so we assume that $v_i \neq v_1$ and let v_h be the parent of v_i. Suppose for the sake of deriving a contradiction that there are two cross edges $\{v_i, v_{i_1}\}, \{v_i, v_{i_2}\}$ incident on v_i. Notice that v_{i_1} is visible at time $i-1$ since otherwise would imply that $\{v_i, v_{i_1}\}$ were a tree edge. Thus, at time $i-1$, v_{i_1} is visible and there is only one tree edge incident on v_i. In particular, this implies that there is a path consisting entirely of tree edges from v_{i_1} to v_h where said path does not contain the edge $\{v_h, v_i\}$ since it does not pass through v_i nor does it contain the edges $\{v_i, v_{i_1}\}, \{v_i, v_{i_2}\}$ since they are cross edges. Thus, by adding edges $\{v_h, v_i\}, \{v_i, v_{i_1}\}$ to this path we obtain a cycle (in the completely revealed input graph) that contains the edge $\{v_h, v_i\}$ but does not contain the edge $\{v_i, v_{i_2}\}$. A similar argument yields that there is a path consisting of tree edges from v_{i_2} to v_h that does not contain the edges $\{v_h, v_i\}, \{v_i, v_{i_1}\}, \{v_i, v_{i_2}\}$ and hence by adding edges $\{v_h, v_i\}, \{v_i, v_{i_1}\}$ we obtain a cycle which contains the edge $\{v_h, v_i\}$ but does not contain the edge $\{v_i, v_{i_1}\}$. That is, two distinct cycles that share the common edge $\{v_h, v_i\}$, a contradiction. □

Since v_i has at most 2 neighbours before it is revealed then it has at least $deg(v_i) - 2$ children. The following is analogous to Corollary 4.2.1 for trees.

Corollary 4.2.3. *If $deg(v_i) \geq 4$ then $v_i \in S$.*

Lemma 4.2.5. *(1) If v_i and v_j both have charge equal to 1 then they share no common neighbours.*

(2) If v_i and v_j both have charge equal to 1 then they are not adjacent.

(3) For any v_i, at most one vertex in $N[v_i]$ has charge 1.

Proof. (1) Follows identically to the proof of Lemma 4.2.2.

(2) First, note that v_1 cannot have charge 1 on any input with at least 2 vertices. Therefore we safely assume that $1 < i < j$ such that both v_i and v_j have a parent. We assume for the sake of deriving a contradiction that v_i and v_j are adjacent.

Now, since both v_i and v_j have charge 1 it follows that they are both saved vertices. We first argue that both $v_i, v_j \notin S$. Notice that any saved vertex v_k has the property that $|N[v_k] \cap S| = 1$. Therefore, if we assume by way of contradiction that $v_i \in S$ we obtain that $N[v_i] \cap S = N[v_j] \cap S = \{v_i\}$ and therefore v_i saves itself and v_j. This yields that $X_i = N[v_i] \cap U_i$ contains v_i and v_j. In particular, we have that $|X_i| \geq 2$ with $v_i, v_j \in X_i$ and therefore v_i and v_j receive charge no larger than $\frac{1}{2}$, a contradiction. An identical argument will yield that $v_j \notin S$.

Thus, we assume that v_i is saved by a neighbour $v_{i'}$ and v_j is saved by a neighbour $v_{j'}$ where $i', j' \notin \{i, j\}$. Moreover, $i' \neq j'$ since v_i and v_j can share no common neighbours by part 1. Thus, we have that i, j, i', j' are all distinct with $i < j < i'$ and $i < j < j'$ since $i' = \max\{k \mid v_k \in N[v_i]\}$ and $j' = \max\{k \mid v_k \in N[v_j]\}$. As mentioned above v_i must have a parent v_h where $h < i < j < i'$. Therefore, $deg(v_i) \geq 3$ and since $v_i \notin S$ it follows by Corollary 4.2.3 that $deg(v_i) = 3$.

We are now in the situation where $v_i, v_j \notin S$ and v_i is incident on exactly 3 edges $\{v_h, v_i\}$, $\{v_i, v_j\}$, $\{v_i, v_{i'}\}$ where exactly one of the edges $\{v_i, v_j\}, \{v_i, v_{i'}\}$ is a tree edge (and the other a cross edge). We finish the proof by examining the two cases where **(1) :** $\{v_i, v_{i'}\}$ is a tree edge or **(2) :** $\{v_i, v_j\}$ is a tree edge.

Case 1 : Suppose $\{v_i, v_{i'}\}$ is a tree edge so that $v_{i'}$ is a child of v_i. Therefore, $v_{i'} \in C_i \subseteq N(v_i) \cap U_i$, that is, $v_{i'}$ is an undominated neighbour of v_i when v_i is revealed. Since v_j

is saved then by Observation 4.1.2 it follows that $v_j \in N(v_i) \cap U_i$, that is, v_j is also an undominated neighbour of v_i when v_i is revealed. That is, both $v_{i'}, v_j \in N(v_i) \cap U_i$ implying that $|N(v_i) \cap U_i| \geq 2$ but $v_i \notin S$, a contradiction.

Case 2 : Suppose $\{v_i, v_j\}$ is a tree edge so that v_j is a child of v_i. First notice that $\{v_i, v_j\}$ is the only tree edge incident on v_j. Indeed, if there were a tree edge $\{v_j, v_l\}$ then v_l would be the child of v_j. Since v_i is saved we have $v_i \in N(v_j) \cap U_j$ by Observation 4.1.2 implying that $|N(v_j) \cap U_j| \geq 2$ but $v_j \notin S$. Thus, we are in the situation depicted in Figure 4.10 where $\{v_i, v_j\}$ is the only tree edge incident on v_j and by assumption $\{v_h, v_i\}, \{v_i, v_j\}$ are the only two tree edges incident on v_i. Therefore we have a path from $v_{i'}$ to v_h consisting of tree edges where said path does not contain the edges $\{v_h, v_i\}, \{v_i, v_{i'}\}, \{v_i, v_j\}, \{v_j, v_{j'}\}$. Thus, by adding edges $\{v_h, v_i\}, \{v_i, v_i'\}$ to this path we obtain a cycle (in the completely revealed input) that contains the edge $\{v_h, v_i\}$ but does not contain the edge $\{v_j, v_{j'}\}$. Similarly, there is a path from $v_{j'}$ to v_h consisting of tree edges where said path does not contain the edges $\{v_h, v_i\}, \{v_i, v_{i'}\}, \{v_i, v_j\}, \{v_j, v_{j'}\}$ and by adding edges $\{v_h, v_i\}, \{v_i, v_j\}, \{v_j, v_{j'}\}$ we obtain a cycle (in the completely revealed input) that contains the edge $\{v_h, v_i\}$ but does not contain the edge $\{v_i, v_{i'}\}$. That is, two distinct cycles that share the common edge $\{v_h, v_i\}$, a contradiction.

(3) Follows immediately from the previous parts.

□

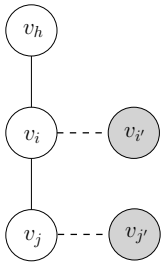

Figure 4.10: Case 2 of the second part of Lemma 4.2.5.

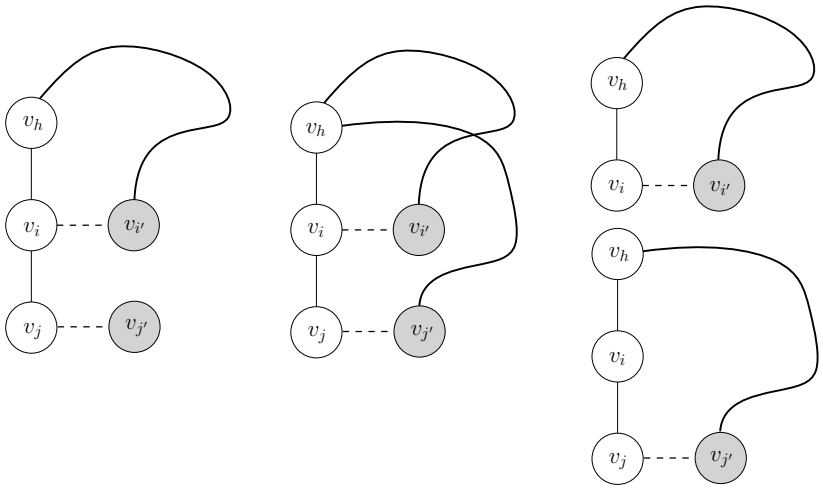

Figure 4.11: Resolution of the preceding case in Figure 4.10. Two cycles sharing the common edge $\{v_h, v_i\}$.

Now, we are ready to prove the upper bound for Theorem 4.2.4.

Proof of Theorem 4.2.4. The lower bound follows from Theorem 4.2.3. Let $v_i \in OPT$ be an arbitrary vertex in OPT. We consider two cases **(1)** $deg(v_i) \leq 3$ or **(2)** $deg(v_i) \geq 4$.

Case 1 : Suppose that $deg(v_i) \leq 3$ and hence $|N[v_i]| \leq 4$. By Lemma 4.2.5 part 3 it follows that at most one vertex in $N[v_i]$ has charge 1. If no vertices in $N[v_i]$ have charge 1 then $ch(x) \leq \frac{1}{2}$ for each $x \in N[v_i]$ and we obtain that $\sum_{x \in N[v_i]} ch(x) \leq 4\left(\frac{1}{2}\right) = 2 < \frac{5}{2}$. If there is exactly one vertex $x' \in N[v_i]$ with charge 1 we therefore obtain that $\sum_{x \in N[v_i]} ch(x) = \sum_{x \in N[v_i] \setminus \{x'\}} ch(x) + ch(x') \leq \frac{3}{2} + 1 = \frac{5}{2}$.

Case 2 : Suppose that $deg(v_i) \geq 4$. By Corollary 4.2.3 it follows that $v_i \in S$ with at least 2 children. Let $C_i = V_i \setminus V_{i-1}$ denote the children of v_i and remark that $C_i \subseteq X_i$. That is, each child of v_i is charged by v_i and only v_i. Therefore the children of v_i can receive at most the full initial charge on v_i and thus attribute a charge of at most 1.

Now we claim that any vertex in $N[v_i] \setminus C_i$ has a charge of at most $\frac{1}{2}$. Indeed, suppose a vertex $v_{i'} \in N[v_i] \setminus C_i$ has charge 1 then it must be saved by v_i since $|N[v_{i'}] \cap S| = 1$ for any saved vertex $v_{i'}$. That is, there is exactly one vertex in its closed neighbourhood that is selected and since v_i is selected it must be v_i. Thus, we must have that $v_{i'} \in X_i$ but since $C_i \subseteq X_i$

we know that $|X_i| \geq 2$ and thus $v_{i'}$ receives a charge of no more than $\frac{1}{2} < 1$, contradicting our assumption that $v_{i'}$ has charge 1. Thus, by remarking that $|N[v_i] \setminus C_i| \leq 3$ we obtain that $\sum_{x \in N[v_i]} ch(x) = \sum_{v_j \in C_i} ch(v_j) + \sum_{v_{i'} \in N[v_i] \setminus C_i} ch(v_{i'}) \leq 1 + 3\left(\frac{1}{2}\right) = \frac{5}{2}$ as desired.

□

4.2.3 Graphs of Bounded Degree

We study the problem when the inputs are restricted to graphs of bounded degree. That is, a positive integer $\Delta \geq 2$ is provided to the algorithm beforehand and the adversary is restricted to presenting graphs where every vertex has degree no larger than Δ. The problem of bounded degree graphs was explored by Boyar et al. (2019) although within the vertex arrival model described earlier. The authors show that a greedy strategy obtains a competitive ratio no larger than Δ and, when inputs are further restricted to be "always-connected" (i.e. each prefix of the input is connected) they provide a lower bound of $\Delta - 2$ for any algorithm.

By definition, any input belonging to our setting is "always-connected" yet the lower bound of $\Delta - 2$ does not apply. In particular, we show that $\lceil\sqrt{\Delta}\rceil$-DOMINATE is $3\sqrt{\Delta}$-competitive along with a lower bound of $\Omega(\sqrt{\Delta})$ for any online algorithm, essentially closing the problem in our setting. As previously mentioned, King and Tzeng (1997) consider a setting similar to ours where their adversary is not required to reveal visible vertices and they assume that an algorithm has additional knowledge of input size n. In this setting they provide an algorithm that achieves competitive ratio of $\Theta(\sqrt{n})$ for arbitrary graphs. For the upper bound below we follow a proof nearly identical to theirs modulo some minor details and definitions.

Definition 4.2.1. *A vertex $v_i \in S$ is said to be heavy if $|N(v_i) \cap U_i| \geq \lceil\sqrt{\Delta}\rceil$ and light otherwise. We let H and L denote the set of heavy and light vertices in S so that $|S| = |H| + |L|$.*

To establish that $\lceil\sqrt{\Delta}\rceil$-DOMINATE is $3\sqrt{\Delta}$-competitive we use a charging argument different from the two given in Sections 4.2.1 and 4.2.2. Initially, let $ch(v) = 1$ for each $v \in S$ so that $|S| = \sum_{v \in S} ch(v)$. Then spread the charge from S strictly to vertices in OPT so that $\sum_{v \in S} ch(v) = \sum_{v \in OPT} ch^*(v)$ where $ch^*(v)$ is the new charge on a vertex in OPT. We then show that $ch^*(v) \leq 3\sqrt{\Delta}$ for all $v \in OPT$ and thus $|S| = \sum_{v \in S} ch(v) = \sum_{v \in OPT} ch^*(v) \leq (2\sqrt{\Delta})|OPT|$ and the result

then follows. We spread the charge from S to OPT according to the following rules:

(1) If $v_i \in S \cap OPT$ then v_i keeps its full initial charge.

(2) If $v_i \in H \setminus OPT$ then its spread its initial charge evenly over all vertices in OPT. That is, each $v \in OPT$ obtains an additional charge of $\frac{1}{|OPT|}$ from v_i.

(3) For each $v_i \in L \setminus OPT$, let $s(v_i)$ denote the set of vertices saved by v_i. Given a vertex $v_{i'} \in s(v_i)$ let $opt(v_{i'}) = v_{i'}$ if $v_{i'} \in OPT$ and $opt(v_{i'}) = \min\{k \mid v_k \in N(v_{i'}) \cap OPT\}$ otherwise. For each $v_{i'} \in s(v_i)$, v_i spreads $\frac{1}{|s(v_i)|}$ to $opt(v_{i'})$.

Lemma 4.2.6. *If $v_i \in OPT$ then it receives charge from at most $\lceil\sqrt{\Delta}\rceil$ light vertices.*

Proof. We consider two cases; **(1)** $v_i \in S$ or **(2)** $v_i \notin S$.

Case 1 : Suppose that $v_i \in S$, we show that v_i then it receives no charge from a distinct light vertex (therefore it receives charge from at most one light vertex, itself). Since $v_i \in S$ this implies that it is not saved by any $v_j, j \neq i$. Thus, if v_i were to receive charge from a light vertex it must be that $v_i = opt(v_{i'})$ for some $v_{i'}$ that is saved by some $v_k \in L$ different from v_i. More precisely, v_i must be adjacent to some $v_{i'}$ that is saved by some v_k with $k \neq i$. Yet, if $v_{i'} \in N(v_i)$ is saved then $N[v_{i'}] \cap S = \{v_i\}$ so this cannot be the case.

Case 2 : Assume that $v_i \notin S$ and first remark that v_i is saved by at most one vertex so that it receives at most one charge from a light vertex in this way. If v_i receives charge from any other light vertex $v_k \in L$, it must be that v_i is adjacent to some vertex $v_{i'}$ that is saved by v_k. By Observation 4.1.2 it must be that $v_{i'} \in N(v_i) \cap U_i$, that is, is undominated when v_i is revealed. All this to say, that any light vertex that charges v_i determines at least one neighbor of v_i that is undominated at time i. Since $v_i \notin S$ we have $|N(v_i \cap U_i| \leq \lceil\sqrt{\Delta}\rceil - 1$ and thus accounting for possibly one light vertex that charges v_i there are at most $\lceil\sqrt{\Delta}\rceil$ light vertices that charge v_i. □

Lemma 4.2.7. $\frac{|H|}{|OPT|} \leq \sqrt{\Delta} + \frac{1}{\sqrt{\Delta}}$.

Proof. Since every vertex in H is selected because it dominated at least $\lceil\sqrt{\Delta}\rceil$ undominated vertices it follows that $|H| \leq \left\lfloor \frac{n}{\lceil\sqrt{\Delta}\rceil} \right\rfloor$. Moreover, by Theorem 2.1.2 we have that $|OPT| \geq \left\lceil \frac{n}{\Delta+1} \right\rceil$.

Ultimately this yields that

$$\frac{|H|}{|OPT|} \leq \frac{\left\lfloor \frac{n}{\lceil \sqrt{\Delta} \rceil} \right\rfloor}{\left\lceil \frac{n}{\Delta+1} \right\rceil} \leq \frac{\frac{n}{\lceil \sqrt{\Delta} \rceil}}{\frac{n}{\Delta+1}} \leq \frac{\frac{n}{\sqrt{\Delta}}}{\frac{n}{\Delta+1}} = \frac{\Delta+1}{\sqrt{\Delta}} = \sqrt{\Delta} + \frac{1}{\sqrt{\Delta}}.$$

□

Theorem 4.2.5. $\rho(\lceil \sqrt{\Delta} \rceil$*-DOMINATE*$, \Delta$*-BOUNDED*$) \leq 3\sqrt{\Delta}$.

Proof. Consider an arbitrary vertex $v_i \in OPT$. In light of Lemma 4.2.6 we see that it receives charge from at most $\lceil \sqrt{\Delta} \rceil$ light vertices, where each charge is no larger than 1. Moreover, by Lemma 4.2.7 the charge received by the heavy vertices is at most $\sqrt{\Delta} + \frac{1}{\sqrt{\Delta}}$ and v_i possibly receives charge from itself (it may be a heavy or light vertex). In particular we obtain that

$$ch(v_i) \leq \frac{|H|}{|OPT|} + \lceil \sqrt{\Delta} \rceil + 1 \leq \left(\sqrt{\Delta} + \frac{1}{\sqrt{\Delta}}\right) + \lceil \sqrt{\Delta} \rceil + 1 \leq 3\sqrt{\Delta}.$$

□

We now prove a lower bound $\Omega(\sqrt{\Delta})$ for any online algorithm. We should note that the adversarial input is bounded in size by a function of Δ. Although we have omitted the details, it is straightforward to extend the input so that the lower bound is in fact an asymptotic one.

Theorem 4.2.6. $\rho(ALG, \Delta$*-BOUNDED*$) = \Omega(\sqrt{\Delta})$.

Proof. For simplicity we assume that Δ is a perfect square. Reveal v_1 with Δ children and reveal each child of v_1 with an additional $\sqrt{\Delta}$ children. Of the Δ children of v_1, suppose that ALG selects exactly j where $0 \leq j \leq \Delta$. For the $\Delta - j$ vertices not selected, their $\sqrt{\Delta}$ neighbours are revealed to have degree 1 and ALG is forced to select each of these $(\Delta - j)(\sqrt{\Delta})$ vertices of degree 1.

Let S_j denote the set of the j selected vertices in $N(v_1)$ and $X = \bigcup_{v_i \in S_j} N(v_i)$. Since each vertex in S_j has $\sqrt{\Delta}$ children, it follows that $|X| = j\sqrt{\Delta}$. Partition the vertices of X into $\lceil \frac{j\sqrt{\Delta}}{\Delta} \rceil = \lceil \frac{j}{\sqrt{\Delta}} \rceil$ parts of size Δ (with at most one part having size $< \Delta$). Letting the parts be $X_1, X_2, ..., X_{\lceil \frac{j}{\sqrt{\Delta}} \rceil}$ we reveal each vertex in a given part to a common vertex y_i (see figure 4.12 for an example). ALG must select at least one vertex for each part to dominate y_i and therefore at least an additional $\lceil \frac{j}{\sqrt{\Delta}} \rceil$ vertices are selected.

In total, ALG selects at least $j + (\Delta - j)(\sqrt{\Delta}) + \frac{j}{\sqrt{\Delta}}$ whereas OPT simply selects v_1, the $\Delta - j$ vertices in $N(v_1) \setminus S_j$ and the $\frac{j}{\sqrt{\Delta}}$ vertices with labels y_i. Ultimately we have

$$\frac{ALG}{OPT} \geq \frac{j + (\Delta - j)(\sqrt{\Delta}) + \frac{j}{\sqrt{\Delta}}}{1 + (\Delta - j) + \frac{j}{\sqrt{\Delta}}} = \frac{j + j\sqrt{\Delta} + (\Delta - j)\Delta}{j + \sqrt{\Delta} + (\Delta - j)\sqrt{\Delta}}$$
$$= \frac{\sqrt{\Delta}(j/\sqrt{\Delta} + j + (\Delta - j)\sqrt{\Delta})}{2(j/2 + \sqrt{\Delta}/2 + (\Delta - j)\sqrt{\Delta}/2)} \geq \frac{\sqrt{\Delta}}{2},$$

where the last inequality follows from the fact that $j/2 + \sqrt{\Delta}/2 + (\Delta - j)\sqrt{\Delta}/2 \leq j/\sqrt{\Delta} + j + (\Delta - j)\sqrt{\Delta}$, since $\sqrt{\Delta}/2 \leq j/\sqrt{\Delta} + j/2 + (\Delta - j)\sqrt{\Delta}/2$, which can be seen since when $j < \Delta$ then the last term on the right hand side already is at least as large as the left hand side and when $j = \Delta$ then the middle term on the right hand side is at least the left hand side.

□

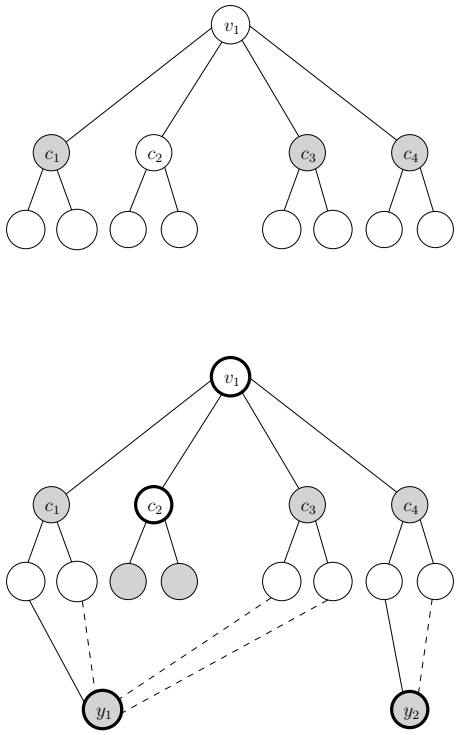

Figure 4.12: An instance described in the proof of Theorem 4.2.6 with $\Delta = 4$. The top depicts the graph after the children of v_1 have been revealed. Assuming that ALG selects $\{c_1, c_3, c_4\}$ above, the bottom depicts the completely revealed graph.

4.2.4 Graphs with Bounded Claws

Let $t \geq 3$, a graph G is said to be $K_{1,t}$-*free* if it contains no induced subgraph isomorphic to $K_{1,t}$. When $t = 3$, this is the well-studied class of *claw-free* graphs. In this section we study $K_{1,t}$-free graphs, which we also refer to as graphs with bounded "claws".

From the preceding sections one might notice that the existence of an induced subgraph $K_{1,t}$ poses challenges for an algorithm. This section suggests that this intuition holds more than just a grain of truth. We show that, when inputs are restricted to $K_{1,t}$-free graphs, the competitive ratio of every algorithm is bounded below by $t-1$ and there is an algorithm that achieves competitive ratio $t-1$. The upper bounds that we have demonstrated so far were all based on the k-DOMINATE algorithm for a suitable choice of parameter k. Interestingly, our upper bound on $K_{1,t}$-free graphs is based on a conceptually simpler GREEDY algorithm. The analysis is no longer based on a charging scheme, but follows from combinatorial properties of graphs with bounded claws.

Theorem 4.2.7. $\rho(ALG, K_{1,t}\text{-}FREE) \geq t-1$.

Proof. Reveal v_1 with $t-1$ children. If ALG does not select v_1 then the input terminates as a star on t vertices (i.e. the $t-1$ neighbours of v_1 are revealed with no additional neighbours). Any feasible algorithm must select the $t-1$ neighbours of v_1 whereas OPT selects v_1 and the statement then follows. Suppose that ALG selects v_1 and let $c_i, 1 \leq i \leq t-1$ be the children of v_1. Reveal c_1 as adjacent to each child of v_1 and with an additional $t-2$ children. If ALG does not select c_1 then the children of c_1 are revealed as leaves whereas the rest of the input is revealed to be a clique. That is, $N[v_1]$ is a clique and only c_1 has children. ALG selected v_1 and is forced to select the $t-2$ children of c_1 whereas OPT selects only c_1 as a single dominating vertex. It is not hard to see that this input is $K_{1,t}$-free and the result then follows (see Figure 4.13 for an example).

Suppose that ALG selects c_1, the input then continues in the following way; For each $2 \leq j \leq t-2$, (as long as ALG is accepting c_j) we reveal c_j as adjacent to every visible vertex and with an additional $t-3$ children. That is, c_j is adjacent to each child $c_i, i \neq j$ of v_1 and the grandchildren of v_1 (i.e. the children of all the c_i with $1 \leq i \leq j$) so that c_j is a single dominating vertex of this prefix.

Case 1 : If there is some $2 \leq j \leq t-2$ such that ALG does not select c_j then the $t-3$ children

of c_j are revealed as leaves, $N[v_i]$ is revealed as a clique, and the $(t-2)+\sum_{i=2}^{j}(t-3) = j(t-3)+1$ grandchildren of v_1 are revealed to form a clique. At this point, ALG has selected $\{v_1, c_1, ..., c_{j-1}\}$ and is now forced to select the $t-3$ children of c_j for an output of at least $j+(t-3) \geq 2+(t-3) = t-1$ whereas OPT selects only c_j so that $\frac{ALG}{OPT} \geq \frac{t-1}{1}$.

We now argue that this input is $K_{1,t}$-free. Notice that for all v in this input we have $N(v) \subseteq N(c_j)$ so that if there is a an induced $K_{1,t}$ with central vertex v then there is a claw with central vertex c_j. Therefore it is sufficient to show that is no claw with central vertex c_j to finish the claim. Suppose for contradiction's sake that there were an induced $K_{1,t}$ where c_j is the central vertex and the t neighbors of c_j are all pairwise non-adjacent. Let G denote the grandchildren of v_1 and remark that any neighbor of c_j is either a child of c_j, a grandchild of v_1, or a vertex from $N[v_1] \setminus \{c_j\}$. Since there are t vertices and c_j only has $t-3$ children by the pigeonhole principle we must have at least two vertices u, v that both are grandchildren of v_1 or both belong $N[v_1] \setminus \{c_j\}$. Yet, both the set of grandchildren of v_1 and $N[v_1] \setminus \{c_j\}$ are cliques. Therefore we have that u and v are adjacent, contradicting our assumption.

Case 2 : If ALG selects each $c_i, 1 \leq i \leq t-2$ then the $(t-2)(t-3)+1$ grandchildren of v_1 are then revealed to form a clique ($N[v_1]$ has already been revealed as a clique). ALG has already selected $\{v_1, c_1, ..., c_{t-2}\}$ and therefore has an output of at least $t-1$ whereas OPT selects only c_{t-2}. An argument similar to the one above will yield that this input is $K_{1,t}$-free and the result then follows. □

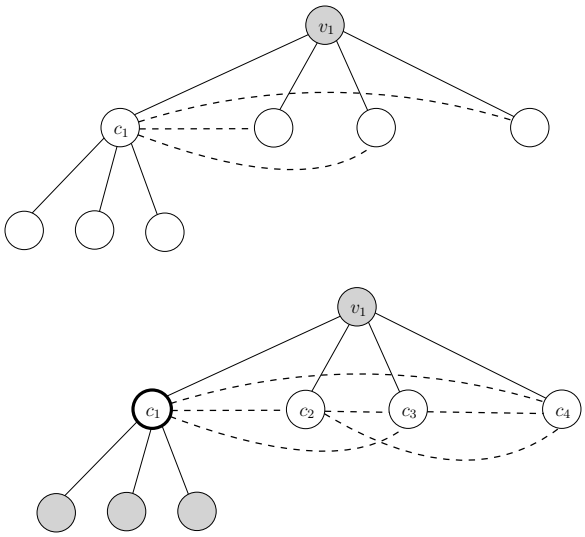

Figure 4.13: An instance described in Theorem 4.2.7 with $t = 5$ where ALG does not select c_1. The top depicts the graph at the moment c_1 was revealed and the bottom depicts the completely revealed graph.

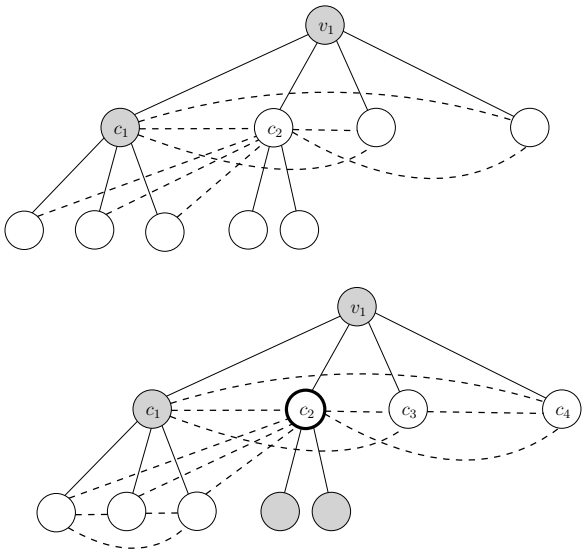

Figure 4.14: An instance described in Theorem 4.2.7 with $t = 5$ where ALG does not select c_2. The top depicts the graph at the moment c_1 was revealed and the bottom depicts the completely revealed graph.

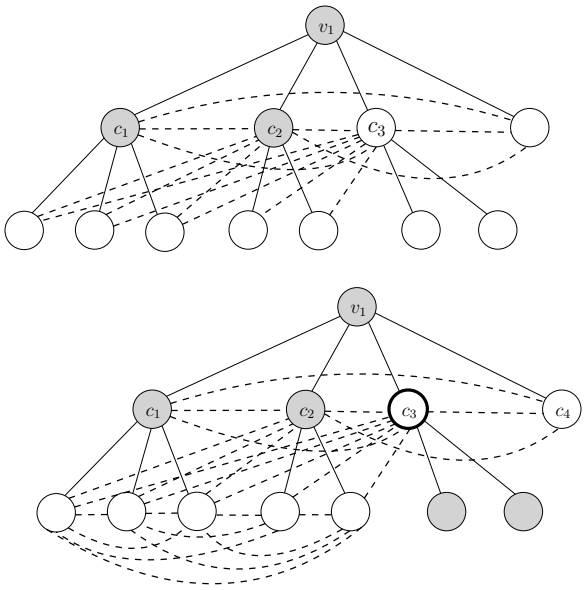

Figure 4.15: An instance described in Theorem 4.2.7 with $t = 5$ where ALG does not select c_3. The top depicts the graph at the moment c_3 was revealed and the bottom depicts the completely revealed graph.

When inputs are restricted to $K_{1,t}$-free graphs, we show that the online algorithm GREEDY is $(t-1)$-competitive. The crucial observation to make here is that the output of GREEDY is an independent set. We provide a result below that is a straightforward generalization of one given in Cygan, Philip, Pilipczuk, Pilipczuk, and Wojtaszczyk (2011). The simplicity of the result suggests that it may have appeared in earlier work.

Lemma 4.2.8. *Let $t \geq 3$, $G = (V, E)$ be a $K_{1,t}$-free graph and I be any independent set in G. Then $|D| \geq \frac{|I|}{t-1}$ for any dominating set D in G.*

Proof. Suppose for the sake of deriving a contradiction that there is some dominating set D in G with $|D| < \frac{|I|}{t-1}$. Remarking that the vertices of D dominate the vertices of I as D is a dominating set we notice that there is some vertex $v \in D$ that dominates at least t vertices of I (i.e. if every vertex of D dominated at most $t-1$ vertices then D would dominate at most $(t-1)|D| < |I|$ vertices). Moreover, since v is adjacent to at least one of the $t \geq 3$ vertices of I it dominates, it cannot belong to I as I is independent. Therefore, the vertices of I dominated by $v \notin I$ are adjacent to v. In particular, at least t vertices of I, all pairwise non-adjacent, are neighbors of v and this

induces $K_{1,t}$ in G. □

The preceding lemma shows that for any independent set I in a $K_{1,t}$-free graph G, $|I| \leq (t-1)\gamma(G)$. Given that GREEDY outputs an independent set we obtain the following result which is of interest to us.

Theorem 4.2.8. *$\rho(GREEDY, K_{1,t}\text{-}FREE) = t - 1$.*

Proof. The upper bound is a consequence of Proposition 4.2.8 and the remarks that follow. The lower bound follows from Theorem 4.2.7. □

4.3 Noncompetitive Graph Classes

Recall that the setting defined in King and Tzeng (1997) is nearly identical to ours except that an algorithm knows the input size n beforehand and the induced subgraph on the revealed vertices is not necessarily connected. Within this setting the authors establish a lower bound of $\Omega(\sqrt{n})$ for arbitrary graphs. Their proof can be augmented to show a lower bound of $\Omega(\sqrt{n})$ in our model, which is tight by our upper bound of $O(\sqrt{\Delta})$ on degree at most Δ graphs (applied to $\Delta = n - 1$). Instead, we strengthen such a result in several ways by showing that the lower bound of $\Omega(\sqrt{n})$ applies to several restricted classes such as threshold graphs[4], planar bipartite graphs, and series-parallel graphs.

4.3.1 Threshold Graphs

The graph join operation applied to two graphs G_1 and G_2 takes the disjoint union of the two graphs and adds all possible edges between the two graphs to the result (in addition to retaining the edges of G_1 and G_2). The class of threshold graphs can be described recursively as follows:

(1) K_1 (i.e. a single isolated vertex) is a threshold graph.

(2) If G is a threshold graph then the disjoint union $G \cup K_1$ is a threshold graph.

(3) If G is a threshold graph then the graph join $G \oplus K_1$ is a threshold graph.

[4]With the caveat that, for threshold graphs, we instead consider the performance ratio as a function of input size.

It is not hard to see that any connected threshold graph has a dominating set of size 1. Since our setting only allows for connected graphs we instead measure ALG as a function of input size n since $OPT \leq 1$ on every input. In particular, we show that for any algorithm there is an infinite family of threshold graphs for which this algorithm selects $\Omega(\sqrt{n})$ vertices (where the input has n vertices). Although OPT does not tend towards infinity, we consider this to be an asymptotic lower bound, but with input size n tending to infinity. In a sense, this is a stronger lower bound since the algorithm is guaranteed an input graph with a single dominating vertex, yet it still selects more than $\Omega(\sqrt{n})$ vertices in the input.

Observation 4.3.1. *The star on $n \geq 1$ vertices, that is, $K_{1,n-1}$, is a threshold graph.*

Now we describe a slightly more complicated graph belonging to the class of threshold graphs. Let $k \geq 2$ and $j_1, ..., j_k$ be non-negative integers. Let $n = k + 1 + \sum_{i=1}^{k} j_i$ and consider the following graph G on n vertices; $V(G) = \{u\} \cup C_k \cup I$, where $C_k = \{v_1, ..., v_k\}$ and $I = I_{j_1} \cup I_{j_2} \cup ... \cup I_{j_k}$, with each I_{j_i} having exactly j_i vertices (each I_{j_i} is possibly empty). The set $\{u\} \cup C_k$ is a clique on $k + 1$ vertices, and for each i, I_{j_i} is an independent set where each $v \in I_{j_i}$ is adjacent only to vertices $v_i, ..., v_k$.

Lemma 4.3.1. *The graph described above is a threshold graph.*

Proof. We describe a construction using the recursive definition given above. Initially, start with the single isolated vertex u. For each $1 \leq i \leq k$, take the resulting graph from the previous step, disjoint union said graph with an independent set I_{j_i} (i.e. repeatedly perform j_i disjoint unions of with a single vertex) and then join the vertex v_i. That is, let $G_0 = (\{u\}, \emptyset)$ and for $1 \leq i \leq k$, $G_i = (G_{i-1} \cup I_{j_i}) \oplus v_i$. □

We are now ready to prove a strong lower bound for any online algorithm. Although we do not mention this explicitly in the proof, the adversarial inputs given are either $K_{1,k-1}$ for some $k \geq 3$ or one that can be obtained by appropriately applying the recursive construction in Lemma 4.3.1.

Theorem 4.3.1. *For infinitely many values of n there is a threshold graph G_n such that*

$$ALG(G_n) = \Omega(\sqrt{n}).$$

Proof. Let $k \geq 3$ be an integer and reveal v_1 with $k-1$ children. If ALG does not select v_1 then the input terminates as a star on k vertices (i.e. the $k-1$ neighbours of v_1 are revealed with no additional neighbours). ALG is forced to select the $k-1$ neighbours of v_1 (OPT selects only v_1). In this case, the statement follows since $ALG = k-1 \geq \sqrt{k} = \sqrt{n}$.

Suppose that ALG selects v_1 and let $c_i, 1 \leq i \leq k-1$ be the children of v_1. Reveal c_1 as adjacent to each child of v_1 and with an additional k children. If ALG does not select c_1 then the children of c_1 are revealed as leaves whereas the rest of the input is revealed to be a clique. That is, $N[v_1]$ is a clique and only c_1 has children. In this case, ALG must select the k children of v_2 yielding an output of $k+1$ (see Figure 4.16 for an example). Therefore, in this case the statement follows since $ALG \geq k+1 = \frac{n}{2} + 1 \geq \sqrt{n}$.

Suppose that ALG selects c_1, the input then continues in the following way; For each $2 \leq j \leq k-1$, (as long as ALG is accepting c_j) we reveal c_j as adjacent to every visible vertex and with an additional k children. That is, c_j is adjacent to each child $c_i, i \neq j$ of v_1 and the grandchildren of v_1 (i.e. the children of all the c_i with $1 \leq i \leq j$) so that c_j is a single dominating vertex of this prefix.

Case 1 : If there is some $2 \leq j \leq k-1$ such that ALG does not select c_j then the k children of c_j are revealed as leaves and $N[v_1]$ is revealed as a clique (see Figure 4.17 for an example). The input has $n = 1 + (k-1) + \sum_{i=2}^{j} ik = k + (j-1)k = jk$ vertices. At this point, ALG has selected $\{v_1, c_1, ..., c_{j-1}\}$ and is now forced to select the k children of c_j (OPT selects only c_j) for an output of at least $j + k \geq 2 + k \geq \sqrt{n}$ since $j < k$ and $n = jk$.

Case 2 : If ALG selects each $c_i, 1 \leq i \leq k-1$ then the input is terminated with $n = 1 + (k-1) + \sum_{i=2}^{j} ik = k + (k-1)k = k^2$ vertices after revealing c_{k-1}. ALG has already selected $\{v_1, c_1, ..., c_{k-1}\}$ and therefore $ALG \geq k = \sqrt{n}$.

□

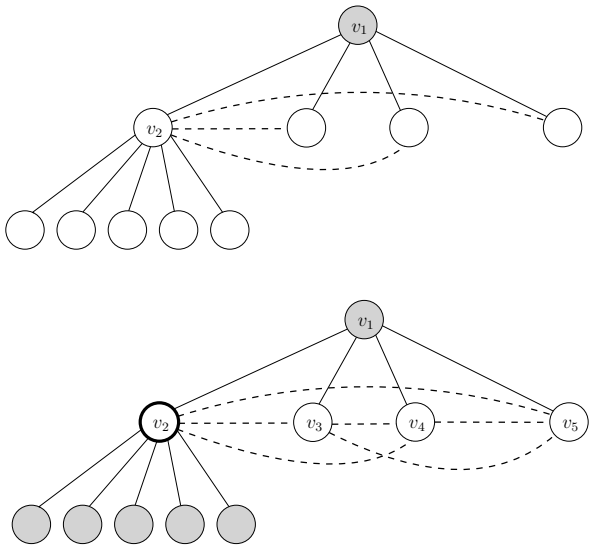

Figure 4.16: An instance described in Theorem 4.3.1 with $k = 5$ where ALG does not select v_2. The top depicts the graph at the moment v_2 was revealed and the bottom depicts the completely revealed graph.

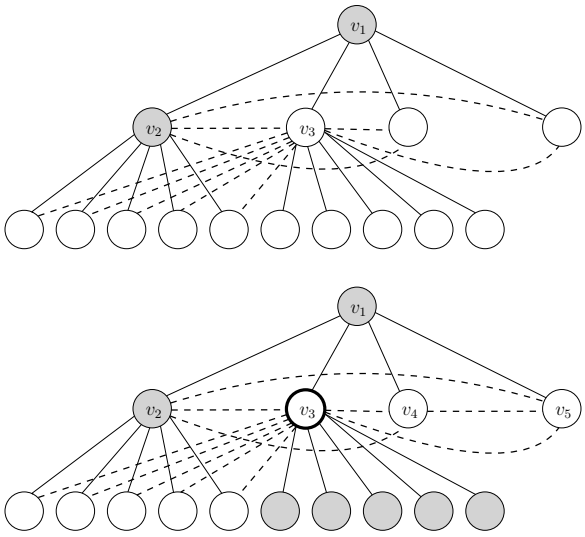

Figure 4.17: An instance described in Theorem 4.3.1 with $k = 5$ where ALG does not select v_3. The top depicts the graph at the moment v_3 was revealed and the bottom depicts the completely revealed graph.

4.3.2 Planar Bipartite Graphs

Below is a lower bound of $\Omega(\sqrt{n})$ for planar bipartite graphs. We should mention that is strikingly similar to the lower bound on general graphs given by King and Tzeng (1997). We provide a simple augmentation of their lower bound so that it not only consists of inputs that are revealed according to our model but inputs that are also planar bipartite graphs.

Theorem 4.3.2. $\rho(ALG,$*PLANAR BIPARTITE*$) = \Omega(\sqrt{n})$.

Proof. Let $k \geq 2$ and consider a path on k vertices with the vertices ordered $v_i, 1 \leq i \leq k$. Each vertex along the path is adjacent to k neighbors appearing as leaves. Every odd labeled vertex is adjacent to a common vertex o and every even labeled vertex is adjacent to a common vertex e where both e, o do not lie on the path (and have not yet been revealed). The ordering of the path is the order in which these vertices were revealed to ALG (see Figure 4.18). Of the k vertices along the path we suppose that ALG selects $k - i$ where $0 \leq i \leq k$. For each of the $k - (k - i) = i$ vertices not selected by ALG, the k leaves adjacent are revealed to remain leaves and ALG must select them. For each of the $k - i$ vertices selected by ALG, the leaves adjacent to said vertices are revealed as adjacent to e if their neighbour had an odd label and o if their neighbour had an even label. Thus, ALG selects at least $(k - i) + (i)k = k + i(k - 1)$ vertices whereas OPT need only select o, e and the i vertices not selected by ALG.

Thus, we have that $\frac{ALG}{OPT} \geq \frac{k+i(k-1)}{i+2}$. Noting that $k - 1 \geq \frac{k}{2}$ since $k \geq 2$ we obtain that

$$i(k-1) \geq i(k/2) \iff k + i(k-1) \geq i(k/2) + k = \frac{k}{2}(i+2) \iff \frac{k + i(k-1)}{i+2} \geq \frac{k}{2}.$$

Since the input consists of $n = k^2 + k + 2$ vertices the result would then follow. To finish we provide a justification that the input is planar and bipartite. To see that it is bipartite let one part X consist of the vertices along the path with odd labels and the neighbors of the vertices with even labels (this includes e). The other part Y consists of the vertices along the path with even labels and the neighbors of the vertices with odd labels (this includes o). To see that it is planar, consider a drawing with the vertices along the path drawn in a line from left to right, e placed above this path and o placed below. For any odd labeled vertex v_i, the k neighbors of v_i that do not lie on the

path (and are different from o) are placed immediately above v_i but below e (i.e. v_i along with said neighbors are depicted as a star on $k+1$ vertices with v_i as the center). Similarly, for any even labeled vertex v_i, the k neighbors of v_i that do not lie on the path and are different from e are placed immediately below v_i and above o. □

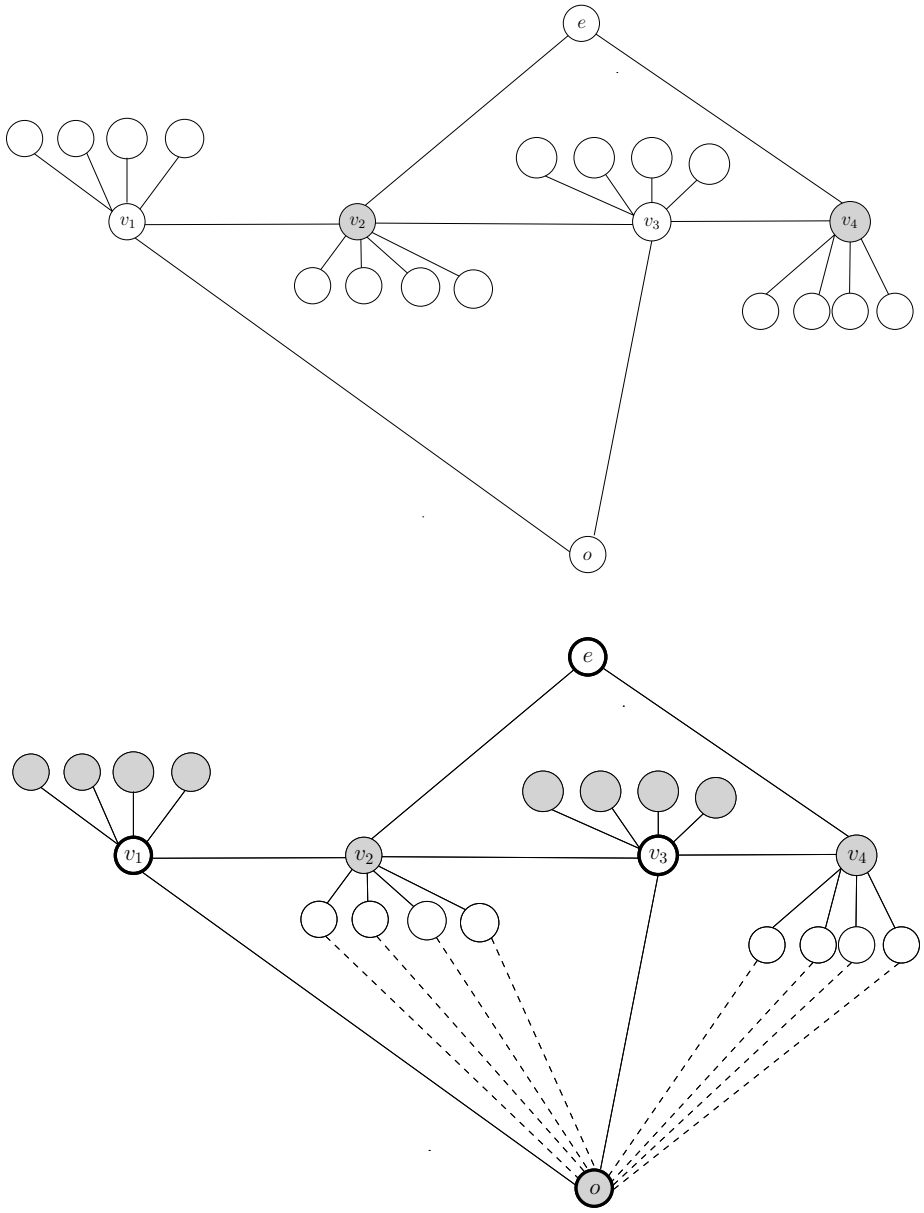

Figure 4.18: An instance described in Theorem 4.3.2 with $k = 4$. The top depicts the prefix where only vertices along the path have been revealed. Assuming the vertices on the path that ALG selects are v_2 and v_4, the bottom depicts the completely revealed graph.

We remark that in Theorem 4.3.2 there are cases when $OPT \leq \alpha$ for some constant $\alpha \geq 2$. For example, when ALG selects all k vertices along the path OPT selects only $\{e, o\}$. In this case, we extend the input by revealing o with an additional neighbor u_1, and repeat a similar trap with u_1 as the first vertex along the path.

4.3.3 Series-Parallel Graphs

In light of our 2-competitive algorithm for trees, it is natural to suppose that some class of graphs generalizing trees might admit competitive algorithms, that is, algorithms with bounded competitive ratio independent of the input size. One such generalization is graphs of bounded treewidth. Trees have treewidth 1, so the next step is to consider graphs of treewidth 2. Unfortunately, in this section we show that by increasing treewidth parameter from 1 to 2, the online dominating set problem becomes extremely hard for online algorithms. More specifically, we show that series-parallel graphs do not admit online algorithms with competitive ratio better than $\Omega(\sqrt{n})$. We remark that series-parallel graphs have treewidth at most 2.

We begin by recalling the definition of a series-parallel graph. It is defined with the help of the notion of a two-terminal graph (G, s, t), which is a graph G with two distinguished vertices s, called a source, and t, called a sink. For a pair of two-terminal graphs (G_1, s_1, t_1) and (G_2, s_2, t_2), there are two composition operations:

- *Parallel composition*: take a disjoint union of G_1 with G_2 and merge s_1 with s_2 to get the new source, as well as t_1 with t_2 to get the new sink.
- *Series composition*: take a disjoint union of G_1 with G_2 and merge t_1 with s_2, which now becomes an inner vertex of the resulting two-terminal graph; s_1 becomes the new source and t_2 becomes the new sink.

A two-terminal series-parallel graph is a two-terminal graph that can be obtained by starting with several copies of the K_2 graph and applying a sequence of parallel and series compositions. Lastly, a graph is called series-parallel if it is a two-terminal series-parallel graph for some choice of source and sink vertices. Observe that intermediate graphs resulting in the construction of a series-parallel

graph may have multiple parallel edges, so they are multigraphs. This is permitted, as long as the resulting overall graph is a simple undirected graph at the end.

Now, we are ready to prove the main result of this section.

Theorem 4.3.3. $\rho(ALG,$*SERIES-PARALLEL*$) = \Omega(\sqrt{n})$.

Proof. Let $k \geq 2$ be an integer. The adversary reveals s with k neighbors $c_1, \ldots, c_k$. Then $c_1, \ldots, c_k$ are revealed in this order with k new neighbors each. Let neighbors of c_i be $d_{i1}, \ldots, d_{ik}$. Let $S \subseteq \{c_1, \ldots, c_k\}$ be those vertices selected by ALG. For those $i \notin S$ we reveal their new neighbors in order $d_{i1}, \ldots, d_{ik}$. Each such d_{ij} is revealed with a single new neighbor f_{ij}. For $i \in S$ we reveal their new neighbors in order $d_{i1}, \ldots, d_{ik}$. Each such d_{ij} is revealed with a new neighbor t that is common to all these vertices. Then f_{ij} are revealed in arbitrary order with t as a new neighbor. Lastly t is revealed without any new neighbors.

Let $p = |S|$. Observe that in addition to these p vertices ALG must select at least one vertex from each of $\{d_{ij}, f_{ij}\}$ pairs for those $i \notin S$; otherwise, vertex d_{ij} would be undominated. Thus, $ALG \geq p + k(k-p)$. Also, observe that $\{s, t\} \cup \{c_i \mid i \notin S\}$ is a dominating set, so $OPT \leq k - p + 2$. The bound on the competitive ratio is

$$\frac{ALG}{OPT} \geq \frac{p + k(k-p)}{k-p+2} = k - \frac{2k-p}{k-p+2} \geq \frac{k}{2},$$

where the last inequality is obtained as follows. For $k \geq 2$ we have $k^2 - kp \geq 2k - 2p$, which implies $k^2 - kp + 2k \geq 4k - 2p$. This in turn implies that $k(k-p+2) \geq 2(2k-p)$, hence $(2k-p)/(k-p+2) \leq k/2$. The quantitative part of the statement of this theorem follows from the fact that the total number of vertices is at most $2 + k + k^2 + k(k-p) = \Theta(k^2)$.

Lastly, we note that the adversarial graph thus constructed is, indeed, series-parallel. For each $i \notin S$ and $j \in \{1, \ldots, k\}$ the path $c_i \to d_{ij} \to f_{ij} \to t$ is a series-composition of 3 copies of K_2. These paths can be merged by a parallel composition to obtain the subgraph induced on $\{c_i, t\} \cup \{d_{ij}, f_{ij} \mid j \in \{1, \ldots, k\}\}$ for each $i \notin S$. Each of these subgraphs is composed at c_i with another copy of K_2 with the new vertex playing the role of s. Similar argument holds to show that the subgraph induced on $\{s, c_i, t\} \cup \{d_{ij} \mid j \in \{1, \ldots, k\}\}$ for $i \in S$ is a two-terminal series-parallel graph. Lastly, all these subgraphs are merged by a sequence of parallel compositions at s and t. □

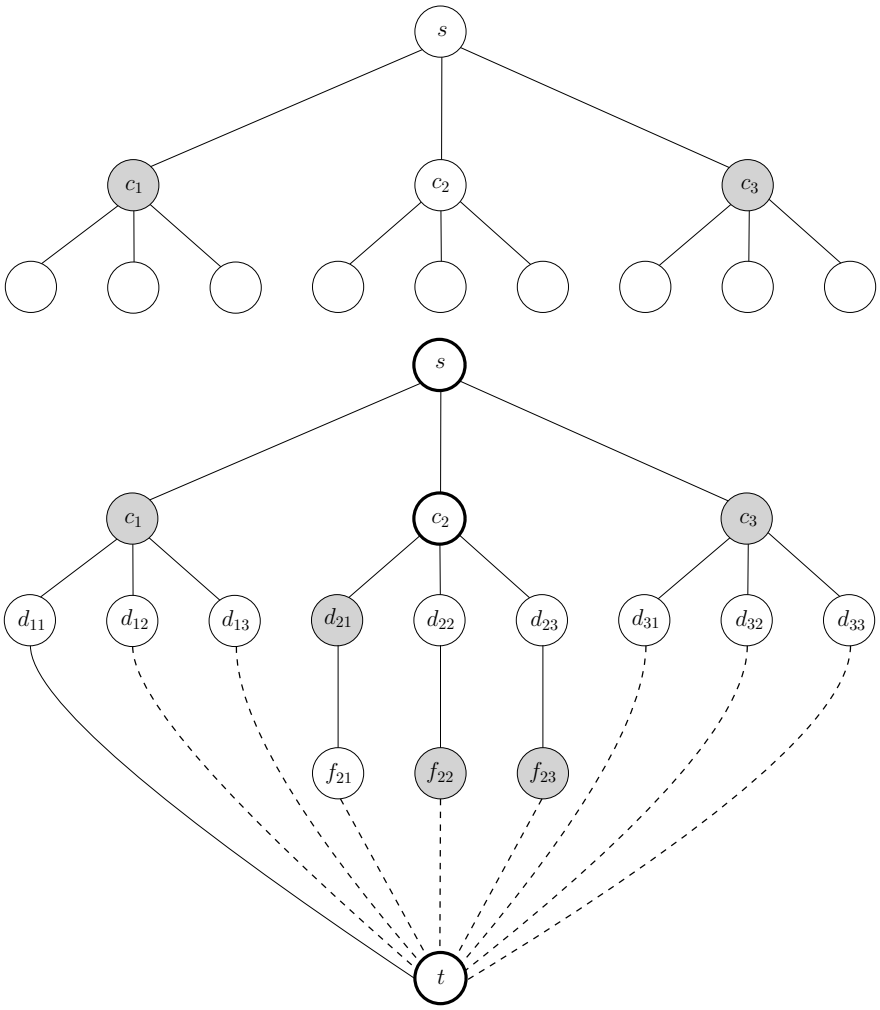

Figure 4.19: An instance described in Theorem 4.3.3 with $k = 3$. The top depicts a prefix where ALG selects $S = \{c_1, c_3\}$ whereas the bottom depicts the completely revealed graph.

Chapter 5

Domination in Knödel graphs

Our investigation thus far has shown that determining $\gamma(G)$ for an arbitrary graph G, is a computationally difficult optimization problem. In many cases, restricting the inputs to belong to certain classes allows the problem to be solved rather efficiently, e.g., with linear or polynomial time algorithms that output $\gamma(G)$. Taking this idea of restricted inputs to its extreme, one may discover special classes with an exploitable combinatorial or algebraic structure for which a closed form mathematical expression can be provided. For example, recall Section 2.2 where we gave mathematical expressions, usually as a function of the number of vertices, for the domination number for several common graph families.

Roughly speaking, we might regard an algorithm to be a sequence of instructions based on mathematical insights, and a closed form expression to be one big mathematical insight (possibly supported by lemmas, observations, etc). Yet, the distinction between an algorithm and a closed form formula is somewhat artificial since the latter still possibly requires adding, subtracting and multiplying many terms. Thus, a closed form expression can often give rise to a linear algorithm, if we are actually interested in the output on a given instance. Perhaps, this distinction is often made because a closed form formula is typically expressed in primitive mathematical terms that consist only of summations, products, binomial coefficients, and so on.

In light of all this, there is still a looming sense that there is a qualitative difference between an algorithm and a closed form expression, with the common notion being that a closed form expression is especially beautiful. Is there a reason as to why it is beautiful? To quote the late, great Paul

Erdős, when he was asked why it is that numbers are beautiful, "It's like asking why is Ludwig van Beethoven's Ninth Symphony beautiful. If you don't see why, someone can't tell you."

The following chapter explores domination, from the "superficially different", mathematical perspective. In particular, we study the domination number of a family of graphs known as the Knödel graphs, which have an algebraic definition exploitable by techniques from elementary number theory. The contents of this chapter is the joint work of the author with Rosso in the following; Racicot and Rosso (2021).

5.1 The Knödel graph

The Knödel graph was implicitly defined in Knödel (1975). Therein, Walter Knödel answers the following question; Given n people, each with a unique message they wish to share with the others where sharing the information with a peer requires one discrete time unit, what is the minimum number of time units required so that every person knows every message. Knödel describes a protocol, referred to as *broadcasting* in the literature, which gives rise to the structure of the Knödel graph, e.g., the people are vertices and for any given person, the list of people they inform throughout the protocol are their neighbors in the graph. The Knödel graph has been a topic of interest since and the interested reader can see Fertin and Raspaud (2004) for an in-depth survey. Here we present the definition of the Knödel graph used by Bermond, Harutyunyan, Liestman, and Perennes (1997) which is equivalent to the original definition of the Knödel graph. We should mention that this particular definition originally appeared in Khachatrian and Haroutunian (1990).

Definition 5.1.1. *Let $n \geq 6$ be even and let $KG_n = (V, E)$ denote the Knödel graph on n vertices where*

$$V = \{0, 1, 2, ..., n-1\}$$

and

$$E = \{\{x, y\} \mid x + y \equiv 2^t - 1 (\text{mod } n)\}$$

$$where\ t = 1, 2, ..., \lfloor \log n \rfloor.$$

Observe that every vertex in KG_n has degree $\lfloor \log n \rfloor$ although it is worth mentioning that there is a more general definition of the Knödel graph. By taking some integer $1 \leq k \leq \lfloor \log n \rfloor$, possibly dependent on n, we can allow the value t (given in the preceding definition) to range from $1, 2, ..., k$ so that you obtain a graph where every vertex has degree k.

Domination in Knödel graphs has been studied in Mojdeh, Musawi, and Nazari (2018); Xueliang, Xu, Yuansheng, and Feng (2009) for the special case where the graph has small constant degree. The study of different variants of domination have also appeared in Jafari Rad, Mojdeh, Musawi, and Nazari (2021); Mojdeh, Musawi, and Nazari (2019); Varghese, Vijayakumar, and Hinz (2018). The Knödel graphs are of particular interest in the area of broadcasting since KG_n is known to be a broadcast graph Bermond et al. (1997). Harutyunyan and Liestman (2012) provide an explicit application of dominating sets to broadcasting. In particular, for a given value of n they construct a sparse broadcast graph on $n + 1$ vertices by finding a dominating set of KG_n satisfying certain properties and joining an additional vertex to every vertex in this dominating set. In fact, the results of Section 5.2 are heavily inspired by said paper.

The main results are Theorems 5.2.3 and 5.2.4. We fix a prime number p dividing n and we suppose that 2 is a primitive root modulo p; in the first theorem we prove that if $p \leq \lceil \log n \rceil$ then $\gamma(KG_n) \leq \frac{n}{p}$. In the second theorem we suppose that 2 is a primitive root modulo p^k, where p^k divides n and Euler's totient function $\phi(p^k) < \lceil \log n \rceil$, and then we show that $\gamma(KG_n) \leq \frac{2n}{p^k}$. Both results are constructive, as we exhibit an explicit dominating set. In summary, our main results provide an upper bound on $\gamma(KG_n)$ for a very general class of even values of n. We remark that it is important to have both results, as it is not expected that 2 is a primitive root modulo p^2 any time that 2 is a primitive root modulo p.

The remainder of the chapter consists of two sections. The first section is the core of the chapter and presents both the state-of-the-art results on $\gamma(KG_n)$ and our main results. The second section will discuss necessary conditions to achieve the best theoretical lower bound on $\gamma(KG_n)$.

5.2 Upper bounds on $\gamma(KG_n)$

The following section presents some upper bounds on $\gamma(KG_n)$ that apply to a large class of even integers. We state a few definitions and preliminary results from elementary number theory as they will be heavily used in the arguments that follow.

Let $n > 1$ be a positive integer. The number of positive integers less than n that are relatively prime to n is given by *Euler's totient function*, denoted by $\phi(n)$. If $\gcd(a, n) = 1$, it is well known that $a^{\phi(n)} \equiv 1 \pmod{n}$ but $\phi(n)$ is not necessarily the smallest integer for which this congruence holds. Thus, we define the *order* of a modulo n as the smallest positive integer k such that $a^k \equiv 1 \pmod{n}$. This prompts the following definition.

Definition 5.2.1. *Let $n > 1$ be a positive integer and let a be an integer such that $\gcd(a, n) = 1$. If a has order $\phi(n)$ modulo n, then a is said to be a* primitive root *modulo n.*

Note that most integers don't have primitive roots; a primitive root exists only when $n = p^k$ or $n = 2p^k$, with p an odd prime, or $n = 2, 4$.

If we let $a_1, a_2, ..., a_{\phi(n)}$ be the positive integers less than n and relatively prime to n then whenever a is a primitive root of n, we have that $a, a^2, ..., a^{\phi(n)}$ are congruent modulo n to $a_1, a_2, ..., a_\phi(n)$ in some order. As well, of particular interest in this paper is the fact that $\phi(p) = p - 1$ and more generally, $\phi(p^k) = p^k - p^{k-1}$ for prime p and $k \geq 1$.

With the preliminaries out of the way we are ready to investigate $\gamma(KG_n)$. We state the following result proved in Harutyunyan and Liestman (2012).

Theorem 5.2.1. *Harutyunyan and Liestman (2012) Let n be even such that $p = \lceil \log n \rceil$ where p is an odd prime. Moreover, suppose that p divides n and that 2 is a primitive root modulo p. It then follows that*

$$\gamma(KG_n) = \frac{n}{p}.$$

Given the conditions in the hypothesis, the authors construct a dominating set of size $\frac{n}{p}$ which yields that $\gamma(KG_n) \leq \frac{n}{p}$. By remarking that $\Delta(KG_n) = \lfloor \log n \rfloor$ and recalling Theorem 2.1.2 which states that $\gamma(G) \geq \left\lceil \frac{n}{\Delta(G)+1} \right\rceil$ for any graph G on n vertices, they obtain that $\gamma(KG_n) \geq$

$\frac{n}{\lfloor \log n \rfloor + 1} = \frac{n}{\lceil \log n \rceil} = \frac{n}{q}$ (implicitly here, we have that $\lfloor \log n \rfloor + 1 = \lceil \log n \rceil = p$ which follows because n is not a power of two since an odd prime p divides n). Thereby establishing $\gamma(KG_n)$ exactly. Thus, the conclusion is as strong as one can hope for, although the conditions in the hypothesis are rather restrictive. In fact, the best known upper bound on $\gamma(KG_n)$ for arbitrary even n, also given in Harutyunyan and Liestman (2012) is stated in the following theorem.

Theorem 5.2.2. *Harutyunyan and Liestman (2012) For arbitrary even n, $\gamma(KG_n) \leq \frac{n}{4}$.*

We will generalize the results of Theorem 5.2.1. In particular, we will relax some of the conditions on the value of n and obtain positive results for $\gamma(KG_n)$. Our first main result of the section is given below. It establishes an upper bound of $\gamma(KG_n)$ whenever n has an odd prime factor $p \leq \lceil \log n \rceil$ such that 2 is a primitive root modulo p. Notice that we have relaxed the condition in Theorem 5.2.1 that the prime p be equal to $\lceil \log n \rceil$. Although this result does not apply to all even values of n, it applies to a rather general class of even integers.

Theorem 5.2.3. *Let n be even and suppose that n has an odd prime factor $p \leq \lceil \log n \rceil$ such that 2 is a primitive root modulo p. It then follows that*

$$\gamma(KG_n) \leq \frac{n}{p}.$$

Proof. Notice that $\frac{n}{2p}$ is indeed an integer because p, an odd prime, is assumed to be a factor of n, where n is also an even number. Thus, we consider the following set $S = \{2pl \mid 0 \leq l \leq \frac{n}{2p} - 1\} \cup \{2pl - 1 \mid 1 \leq l \leq \frac{n}{2p}\}$. We will argue that S is a dominating set in KG_n and given that the size of S is $\frac{n}{2p} + \frac{n}{2p} = \frac{n}{p}$, the result will then follow.

To show that S is a dominating set we will show that any vertex $x \notin S$ is adjacent to a vertex in S with a constructive argument. That is, we will give a closed form expression for the neighbour in S which depends on the value of x.

Let $x \in V \setminus S$ be an arbitrary vertex. Since $x \notin S$, it follows that x takes the form $x = 2pl_0 + m_0$, where $0 \leq l_0 \leq \frac{n}{2p} - 1$ and $1 \leq m_0 \leq 2p - 2$. We break the proof into two cases, based on the parity of x.

First consider the case where x is odd. Since x is odd we must have that m_0 is odd which implies that $m_0 \neq p - 1$. That is, $1 \leq m_0 \leq 2p - 2$ and $m_0 \neq p - 1$. Therefore, we have that

$\gcd(m_0+1, p) = 1$. Using the fact that 2 is a primitive root modulo p we obtain that $m_0 + 1 \equiv 2^i \pmod{p}$ for some $1 \leq i \leq p-1$.

That is, $2^i - 1 = m_0 + jp$ where j is even because $2^i - 1$, m_0 and p are all odd. Thus, consider $l_1 = \frac{j}{2} - l_0$ and $l_2 = \frac{n}{2p} + \frac{j}{2} - l_0$. Notice that both l_1 and l_2 are integers because both $\frac{n}{2p}$ and $\frac{j}{2}$ are integers by previous remarks made. Also, notice that at least one of l_1 or l_2 is between 0 and $\frac{n}{2p} - 1$, as $i < p \leq \lceil \log n \rceil$.

If $0 \leq l_1 \leq \frac{n}{2p} - 1$ then we take $s = 2pl_1 \in S$. We have that $x + s = (2pl_0 + m_0) + 2pl_1 = (2pl_0 + m_0) + 2p(\frac{j}{2} - l_0) = (m_0 + jp) = 2^i - 1$. That is, $x + s = 2^i - 1$ and s is therefore adjacent to x.

Similarly, if $0 \leq l_2 \leq \frac{n}{2p} - 1$ then we take $s = 2pl_2 \in S$ and obtain that $x + s = (2pl_0 + m_0) + 2pl_2 = (2pl_0 + m_0) + 2p(\frac{n}{2p} + \frac{j}{2} - l_o) = (m_0 + jp) + n = 2^i - 1 + n$. That is, $x + s \equiv 2^i \pmod{n}$ and s is therefore adjacent to x.

Thus, in the case that x is odd we have shown that there is a vertex in S that is adjacent to x.

Now, consider the case where x is even. We have that m_0 must be even and therefore $\gcd(m_0, p) = 1$. Similarly we obtain that $m_0 \equiv 2^i \pmod{p}$ for some $1 \leq i \leq p-1$.

That is, $2^i = m_0 + jp$ where j must be even. Following the argument given above we consider $l_1 = \frac{j}{2} - l_0$ and $l_2 = \frac{n}{2p} + \frac{j}{2} - l_0$ and select $s = 2pl_1 - 1$ or $s = 2pl_2 - 1$ accordingly. One of which must be a vertex in S adjacent to x. □

We informally state an immediate consequence of this theorem. Consider any even n which has an odd prime factor that satisfies the aforementioned conditions. One can select the largest such prime factor of n to achieve the strongest result. When a prime greater than 3 with the desired properties is found we have established a better upper bound than $\frac{n}{4}$ for a rather general class of even integers. The formal statement is given explicitly in the following corollary.

Corollary 5.2.1. *Let n be even and suppose that n has a prime factor $3 < p \leq \lceil \log n \rceil$ such that 2 is a primitive root modulo p. It then follows that*

$$\gamma(KG_n) \leq \frac{n}{p} < \frac{n}{4}.$$

We now turn to the second main result in this section. We present an upper bound on a slightly

more restricted class of even integers. Although it should be noted that, in certain cases, this upper bound can be quite strong in comparison.

Theorem 5.2.4. *Let n be even, p be an odd prime and $k \geq 2$ be an integer. Suppose that $\phi(p^k) < \lceil \log n \rceil$, that p^k divides n, and that 2 is a primitive root modulo p^k. It then follows that*

$$\gamma(KG_n) \leq \frac{2n}{p^k}.$$

Proof. Let $S = \{lp^k \mid 0 \leq l \leq \frac{n}{p^k} - 1\} \cup \{lp^k - 1 \mid 1 \leq l \leq \frac{n}{p^k}\}$. We will show that S is a dominating set of KG_n by following a proof similar to 5.2.3 which now considers a couple more cases.

Let $x \in V \setminus S$ be an arbitrary vertex and remark that x takes the form $x = l_0p^k + m_0$, where $0 \leq l_0 \leq \frac{n}{p^k} - 1$ and $1 \leq m_0 \leq p^k - 2$. We consider the case where x is odd and the details for the case where x is even follow similarly.

Suppose that $x = p^k l_0 + m_0$ is odd. We therefore have that either **(1)** l_0 is even and m_0 is odd or **(2)** l_0 is odd and m_0 is even.

Case 1 : If we suppose that l_0 is even and m_0 is odd then we have that $m_0 + 1$ is even and therefore $\gcd(m_0 + 1, p^k) = 1$ or $\gcd(m_0 + 1, p^k) = p^a$ for some $a > 0$.

If $\gcd(m_0 + 1, p^k) = 1$ we therefore obtain that $m_0 + 1 \equiv 2^i \pmod{p^k}$ for some $1 \leq i \leq p^k - p^{k-1}$. That is to say, $m_0 = 2^i - 1 + jp^k$.

Thus, consider $l_1 = -(j + l_0)$ and $l_2 = \frac{n}{p^k} - (j + l_0)$. Depending on the size of l_1 or l_2 we select $s = l_1p^k$ or $s = l_2p^k$ as a vertex in S and obtain that s is adjacent to x.

If $\gcd(m_0 + 1, p^k) = p^a$ with $a > 0$ then $\gcd(m_0, p^k) = 1$ and $m_0 = 2^i + jp^k$. Similarly, we consider $l_1 = -(j + l_0)$ or $l_2 = \frac{n}{p^k} - (j + l_0)$ and pick $s = l_1p^k - 1$ or $s = l_2p^k - 1$ accordingly and we are done.

Case 2 : In the case that l_0 is odd and m_0 is even we have that $\gcd(m_0, p^k) = 1$ or $\gcd(m_0, p^k) = p^a$ for some $a > 0$.

If $\gcd(m_0, p^k) = 1$ then $m_0 = 2^i + jp^k$. Consider $l_1 = -(j + l_0)$ or $l_2 = \frac{n}{p^k} - (j + l_0)$ and pick $s = l_1p^k - 1$ or $s = l_2p^k - 1$.

If $\gcd(m_0, p^k) = p^a$ with $a > 0$ then $\gcd(m_0 + 1, p^k) = 1$ and $m_0 = 2^i - 1 + jp^k$. Consider

$l_1 = -(j + l_0)$ or $l_2 = \frac{n}{p^k} - (j + l_0)$ and pick $s = l_1 p^k$ or $s = l_2 p^k$.

□

Informally, we can discuss the power of the two previous theorems. Take n to be even and consider $n = p_1^{k_1} p_2^{k_2} ... p_j^{k_j}$. Find the largest prime p_l that meets the conditions in Theorem 5.2.3 and the largest prime p_h that meets the conditions in Theorem 5.2.4. These two primes, p_l and p_h, may coincide. We then have that $\gamma(KG_n) \leq \min\left\{\frac{2n}{p_h^{k_h}}, \frac{n}{p_l}\right\}$.

It is worth comparing our results with the results of Harutyunyan and Liestman (2012). For the values of n for which their results apply, the authors achieved the strongest possible bound on $\gamma(KG_n)$. Yet, one should mention that they have not necessarily provided results for an infinite family of values. Indeed, a condition on n is that $\lceil \log n \rceil$ is a prime with 2 as a primitive root although it is not known whether there are infinitely many such primes (i.e. this is a special case of Artin's Conjecture Li and Pomerance (2002)). Thus whether their results apply to infinitely many even values of n is conditional on the conjecture. Although not necessarily the strongest bounds in some cases, the results that we have presented unconditionally apply to an infinite family of even values.

5.3 Necessary Conditions for $\gamma(KG_n) = \left\lceil \frac{n}{\lfloor \log n \rfloor + 1} \right\rceil$

As previously mentioned we have a lower bound of $\left\lceil \frac{n}{\lfloor \log n \rfloor + 1} \right\rceil$ on $\gamma(KG_n)$. As shown in Harutyunyan and Liestman (2012), there are some sufficient conditions which allow KG_n to meet this lower bound. Although this section could have been appropriately titled lower bounds on $\gamma(KG_n)$ we will see that the results are not much better than the preceding general bound. In this section we investigate necessary conditions on the value of n if $\gamma(KG_n) = \left\lceil \frac{n}{\lfloor \log n \rfloor + 1} \right\rceil$ which can very well be interpreted as lower bounds on the domination number.

To begin we present a few definitions and well known results which will be of use. A graph $G = (V, E)$ is said to *k-regular* for some integer $k \geq 0$ if every vertex has degree k. A dominating set $S \subseteq V$ is called a *perfect* if every vertex in $V \setminus S$ has exactly one neighbour in S. A dominating set $S \subseteq V$ is called *efficient* if it is a perfect dominating set that is also independent (i.e. for any pair

$u, v \in S$ we have $\{u, v\} \notin E$). The following result is attributed to Haynes et al. (1998).

Theorem 5.3.1. *Haynes et al. (1998) Let G be a graph on n vertices and suppose that $\frac{n}{\Delta+1}$ is an integer. If $\gamma(G) = \frac{n}{\Delta+1}$ then every γ-set is an efficient dominating set.*

This result has some interesting implications which are illustrated in the following proposition. We believe that this proposition may possibly already be known yet not stated in the literature explicitly.

Proposition 5.3.1. *Let $G = (V, E)$ be a $k \geq 2$-regular bipartite graph on n vertices and suppose that $\frac{n}{k+1}$ is an integer. If $\gamma(G) = \frac{n}{k+1}$ then $\frac{n}{k+1}$ is an even integer. Moreover, every γ-set S of G can be partitioned into two equal sized independent sets.*

Proof. Let X and Y be a partition of V with $|X| = x$ and $|Y| = y$. First, notice that $x = y$. One can realize this by counting the edges of G in two ways. That is, $|E| = k|X|$ and $|E| = k|Y|$ since G is a k-regular bipartite graph. Hence, $kx = ky$ which implies that $x = y$.

Now let S be any γ-set of G and let $S_X = X \cap S$ and $S_Y = Y \cap S$. That is, S_X and S_Y partition the vertices in S with respect to the aforementioned partition of V. Denoting $x_0 = |S_X|$ and $y_0 = |S_Y|$ we will show that $x_0 = y_0$ and the result will then follow.

Remark that S is an efficient dominating set by Theorem 5.3.1. Thus, for the kx_0 edges incident on the x_0 vertices in S_X we count precisely kx_0 distinct vertices belonging to Y that are not in S_Y. Now, since every vertex in Y is either in S_Y or counted by these incident edges, we obtain that $y = y_0 + kx_0$. Similar remarks yield that $x = x_0 + ky_0$.

Noting that $x = y$ we obtain that $y_0 + kx_0 = x_0 + ky_0$. Standard algebraic manipulation yields that $x_0 = y_0$ and the result then follows. □

One obtains the immediate corollary by remarking that KG_n is a $\lfloor \log n \rfloor$-regular bipartite graph that is partitioned based on the parity of the vertices (i.e. even vertices belong to one partition and odd vertices belong to the other).

Corollary 5.3.1. *Let n be even and suppose that $\frac{n}{\lfloor \log n \rfloor + 1}$ is an integer. If $\gamma(KG_n) = \frac{n}{\lfloor \log n \rfloor + 1}$ then $\frac{n}{\lfloor \log n \rfloor + 1}$ is an even integer. Moreover, every γ-set S can be partitioned into two equal sized independent sets S_E and S_O consisting of even and odd vertices, respectively.*

In some sense, this next proposition generalizes the last one. It is perhaps better understood by considering the contrapositive. Roughly put, it states that if one were to meet this lower bound of $\lceil \frac{n}{k+1} \rceil$ and this value is an odd integer, then division of n by $k+1$ must leave a small remainder.

Proposition 5.3.2. *Let $G = (V, E)$ be a $k \geq 2$-regular bipartite graph on n vertices and suppose that $n = 2j(k+1) + r$ with $4 \leq r < k+1$. It then follows that $\gamma(G) > 2j + 1 = \lceil \frac{n}{k+1} \rceil$.*

Proof. Suppose for the sake of deriving a contradiction that $\gamma(G) = 2j + 1$ and let S be a γ-set of G. Consider $S_X = S \cap X$ and $S_Y = S \cap X$ where X and Y are partitions of V.

Since $|S| = 2j + 1$ it follows that at least one of S_X or S_Y has no more than j vertices. Without loss of generality, assume that $|S_X| \leq j$. We will show that S cannot possibly dominate all the vertices of Y.

If $|S_X| = j$ then S_X dominates at most $k|S_X| = kj$ vertices of Y. Since there are $j+1$ vertices in S_Y we see that the number of vertices in Y that S dominates is at most $k|S_X| + (j+1) = kj + (j+1) = j(k+1) + 1 < \frac{n}{2}$. But we know that $|X| = |Y| = \frac{n}{2}$ from previous remarks and thus the vertices of Y are not dominated by S contradicting the fact that S is a dominating set.

It is clear from the argument that that S cannot be a dominating set whenever $|S_X| < j$. Therefore, it must be that $\gamma(G) > 2j + 1 = \lceil \frac{n}{k+1} \rceil$. □

Corollary 5.3.2. *Let n be even and let $k = \lfloor \log n \rfloor$. If $n = 2j(k+1) + r$ with $4 \leq r < k+1$. It then follows that $\gamma(KG_n) > 2j + 1 = \lceil \frac{n}{\lfloor \log n \rfloor + 1} \rceil$.*

Chapter 6

Conclusions and Further Research

This thesis explored the topic of domination from a computational perspective in Chapters 3 and 4 and from a pure mathematical perspective in Chapter 5. The area of offline domination is very rich and Chapter 3 surveys some of the known computational complexity results for the dominating set problem, and although no original contributions were made in this area, future research in this area is still of interest. The rest of this section consists of concluding remarks for both Online Domination and Domination in Knödel Graphs.

6.1 Online Domination

In Chapter 4 we studied the minimum dominating set problem in an online setting where a vertex is revealed alongside all its neighbors. We also contrasted our results with those obtained by Boyar et al. (2019) and Kobayashi (2017) in a related vertex-arrival model. In our setting, the best achievable competitive ratio on general graphs is $\Theta(\sqrt{n})$. This observation prompted us to study this problem with respect to more restrictive graph classes. Trees provide a natural graph class that usually allows for non-trivial competitive ratios. Indeed, we showed that in our model trees admit 2-competitive algorithms. There are several ways to try to extend this result to larger graph classes. We considered cactus graphs and showed that the optimal competitive ratio is 2.5 on them. Another way of generalizing trees is to consider graphs of higher treewidth. Unfortunately, once treewidth goes up to 2, competitive ratio jumps to $\Omega(\sqrt{n})$ (which is trivial in our setting due to $O(\sqrt{n})$ upper

bound), as witnessed by series-parallel graphs. We also established non-trivial upper bounds on graphs of bounded degree, as well as graphs with bounded claws. When one moves to planar (even bipartite planar) graphs and threshold graphs, the competitive ratio jumps to $\Omega(\sqrt{n})$ again.

The above can be viewed as the beginnings of a larger program of developing a deeper understanding of the dominating set problem in an online setting. What are the main structural obstacles in graphs that prohibit online algorithms with small competitive ratios? In particular, we have the following question.

Question 6.1.1. *(Informal) Can one discover a family of graphs parameterized by some parameter t, which include cactus graphs, claw-free graphs, and bounded-degree graphs, such that the competitive ratio scales gracefully with t?*

If we restrict our attention to the results on trees and cactus graphs, a natural generalization of these results suggests itself. A graph G is said to be almost-tree(k) if it is connected and every block (biconnected component) of G can be made into a tree by removing no more than k edges. For $k = 0$ and $k = 1$, we obtain the class of trees and cactus graphs, respectively. We suspect the following conjecture to be true.

Conjecture 6.1.1. *Let $k \geq 0$ be an integer and $f(k)$ be a linear function of k. Then there is an algorithm ALG_k such that $\rho(ALG_k$,ALMOST-TREE$(k)) = 2+f(k)$. More specifically, we suspect that ρ(2-DOMINATE, ALMOST-TREE$(k)) = 2 + \frac{k}{2}$.*

The unanswered question still remains,"What exactly is the value of getting to know all your neighbors?" In the context of online domination, we would like to know if a quantitative comparison could be made between the setting given in Boyar et al. (2019) and the setting explored in Chapter 4. We believe that regardless of the graph class, there is always utility in getting to know the entire neighborhood of a vertex, even if decisions cannot be delayed. The results of Chapter 4 can be seen as evidence for this claim. In the case that this belief is false, we would like to know what are the exact classes for which the best possible online algorithm in a (partially irrevocable) vertex-arrival setting is better than the best possible algorithm in our "full-neighborhood" setting where decisions are completely irrevocable. An answer to this question would let us know exactly when those "neighborhood block parties" can be a bad idea.

We are under the impression that getting to know all your neighbors is a good idea, what can be said about getting to know your neighbors' neighbors? More generally, we could consider a setting where a vertex v is revealed with all the vertices found at distance at most k from v for some $k \geq 1$.

A *hypergraph* is a pair $H = (V, E)$ where V is a finite set of vertices and E is a collection of subsets of V called *hyperedges*. Thus, a graph is a hypergraph where every hyperedge has size two. There is an analogous definition of a dominating set in a hypergraph; A subset $D \subseteq V$ is called a dominating set if for all $v \in V \setminus D$ there is some $u \in D$ such that $\{u, v\} \subseteq e$ for some $e \in E$. That is, both vertices u, v belong to a common hyperedge. It would be interesting to explore online domination on hypergraphs.

Lastly, in another research direction, we mention that we have only considered the deterministic setting, so it would be of interest to extend our results to the randomized setting, as well as the setting of online algorithms with advice.

6.2 Domination in Knödel Graphs

In Chapter 5 we explored the topic of domination in Knödel Graphs. Our main contribution is that we use techniques from elementary number theory in a novel way to establish an upper bound on $\gamma(KG_n)$. In particular, in Theorem 5.2.3 we show that whenever we find a prime p dividing n, where 2 is a primitive root modulo p and $p \leq \lceil \log n \rceil$ then $\gamma(KG_n) \leq \frac{n}{p}$. In Theorem 5.2.4 we suppose that 2 is a primitive root modulo p^k, where p^k divides n and $\phi(p^k) < \lceil \log n \rceil$ to ultimately show that $\gamma(KG_n) \leq \frac{2n}{p^k}$. In comparing our results with that of Harutyunyan and Liestman (2012) we remark that their results apply to infinitely many even values of n only if Artin's conjecture holds whereas the results we have presented unconditionally apply to an infinite family of even values.

We propose a few conjectures in order from most likely to least likely. The first two conjectures seem quite likely to be true although the details may be slightly hairier and we therefore decided not to pursue them. In particular, these conjectures would allow one to relax the restrictions in Theorems 5.2.3 and 5.2.4 which require that p be a factor of n. A possible dominating set could be similar to the dominating sets provided in Theorems 5.2.3 and 5.2.4 but instead each vertex would be translated by the remainder that n leaves upon division by said prime p.

Conjecture 6.2.1. *Let n be even and let $p \leq \lceil \log n \rceil$ be an odd prime such that 2 is a primitive root modulo p. It then follows that*

$$\gamma(KG_n) \leq \left\lceil \frac{n}{p} \right\rceil.$$

Conjecture 6.2.2. *Let n be even, p be an odd prime, $k \geq 2$ be an integer. If $\phi(p^k) < \lceil \log n \rceil$ and 2 is a primitive root modulo p^k then*

$$\gamma(KG_n) \leq \left\lceil \frac{2n}{p^k} \right\rceil.$$

Lastly, the final conjecture is that $\gamma(KG_n)$ is in fact very close to the trivial lower bound in Theorem 2.1.2. Perhaps, with some small additive constant. Roughly put, due to the symmetry of KG_n we would imagine this to be true, yet believe that the dominating set would not be as neatly described as it were in Theorems 5.2.3 and 5.2.4.

Conjecture 6.2.3. *Let n be even and not a power of 2 with $\lceil \log n \rceil = m$. It then follows that*

$$\left\lceil \frac{n}{m} \right\rceil \leq \gamma(KG_n) \leq \left\lceil \frac{n}{m} \right\rceil + \alpha.$$

for some small constant $\alpha > 0$.

Finally, we leave some questions whose answer may be insightful in resolving these conjectures and ultimately determining $\gamma(KG_n)$.

Question 6.2.1. *The upper bounds presented in Section 5.2 impose the restriction that an odd prime factor of $\lceil \log n \rceil$ must have 2 as a primitive root. Can we relax this constraint? An idea may be to construct a dominating set based on the order of 2 modulo some chosen prime p.*

Question 6.2.2. *The necessary conditions presented in Section 5.3 are based on general arguments for k-regular bipartite graphs. Although these arguments are appreciated for their own sake they do not incorporate the algebraic structure of KG_n. Can we strengthen the necessary conditions presented or simply find better lower bounds on $\gamma(KG_n)$?*

Question 6.2.3. *Can some of the techniques used here be applied to other classes of graphs that exhibit a similar algebraic structure?*

References

Alimonti, P., & Kann, V. (1997). Hardness of approximating problems on cubic graphs. In *Italian conference on algorithms and complexity* (pp. 288–298).

Arnborg, S., & Proskurowski, A. (1989). Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete applied mathematics*, *23*(1), 11–24.

Berge, C. (1962). The Theory of Graphs and its Applications, Methuen & Co. *Ltd., London*.

Bermond, J.-C., Harutyunyan, H. A., Liestman, A. L., & Perennes, S. (1997). A note on the dimensionality of modified Knödel graphs. *International Journal of Foundations of Computer Science*, *8*(02), 109–116.

Bertossi, A. A. (1984). Dominating sets for split and bipartite graphs. *Information processing letters*, *19*(1), 37–40.

Böckenhauer, H.-J., Hromkovič, J., Krug, S., & Unger, W. (2021). On the advice complexity of the online dominating set problem. *Theoretical Computer Science*.

Bollobás, B. (1998). *Modern Graph Theory*. Springer.

Booth, K. S., & Johnson, J. H. (1982). Dominating sets in chordal graphs. *SIAM Journal on Computing*, *11*(1), 191–199.

Borodin, A., & El-Yaniv, R. (1998). *Online computation and competitive analysis*. Cambridge University Press.

Boyar, J., Eidenbenz, S. J., Favrholdt, L. M., Kotrbcík, M., & Larsen, K. S. (2019). Online dominating set. *Algorithmica*, *81*(5), 1938–1964.

Chang, G. J., & Nemhauser, G. L. (1984). The k-domination and k-stability problems on sun-free chordal graphs. *SIAM Journal on Algebraic Discrete Methods*, *5*(3), 332–345.

Cockayne, E. J., Goodman, S., & Hedetniemi, S. (1975). A linear algorithm for the domination number of a tree. *Information Processing Letters*, *4*(2), 41–44.

Cockayne, E. J., & Hedetniemi, S. T. (1977). Towards a theory of domination in graphs. *Networks*, *7*(3), 247–261.

Corneil, D. G., & Keil, J. (1987). A dynamic programming approach to the dominating set problem on k-trees. *SIAM Journal on Algebraic Discrete Methods*, *8*(4), 535–543.

Corneil, D. G., & Perl, Y. (1984). Clustering and domination in perfect graphs. *Discrete Applied Mathematics*, *9*(1), 27–39.

Corneil, D. G., & Stewart, L. K. (1990). Dominating sets in perfect graphs. *Discrete Mathematics*, *86*(1-3), 145–164.

Cygan, M., Philip, G., Pilipczuk, M., Pilipczuk, M., & Wojtaszczyk, J. O. (2011). Dominating set is fixed parameter tractable in claw-free graphs. *Theoretical Computer Science*, *412*(50), 6982–7000.

Das, B., & Bharghavan, V. (1997). Routing in ad-hoc networks using minimum connected dominating sets. In *1997 IEEE international conference on communications: Towards the Knowledge Millennium, ICC 1997, Montréal, Québec, Canada, June 8-12, 1997* (pp. 376–380). IEEE.

de Jaenisch, C.-F. (1862). *Traité des applications de l'analyse Mathématique au Jeu des échecs*. l'Académie impériale des sciences.

De Meo, P., Ferrara, E., Fiumara, G., & Provetti, A. (2014). Mixing local and global information for community detection in large networks. *Journal of Computer and System Sciences*, *80*(1), 72–87.

Duh, R.-c., & Fürer, M. (1997). Approximation of k-set cover by semi-local optimization. In *Proceedings of the twenty-ninth annual acm symposium on theory of computing* (pp. 256–264).

Eidenbenz, S. J. (2002). *Online dominating set and variations on restricted graph classes* (Tech. Rep. No. 380). Department of Computer Science, ETH Zurich.

Feige, U. (1998). A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, *45*(4), 634–652.

Fertin, G., & Raspaud, A. (2004). A survey on Knödel graphs. *Discrete Applied Mathematics*,

137(2), 173–195.

Fomin, F. V., Grandoni, F., & Kratsch, D. (2005). Some new techniques in design and analysis of exact (exponential) algorithms. *Bulletin of the EATCS*, *87*(47-77), 0–288.

Fomin, F. V., Kratsch, D., & Woeginger, G. J. (2004). Exact (exponential) algorithms for the dominating set problem. In *International workshop on graph-theoretic concepts in computer science* (pp. 245–256).

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability : A Guide to the Theory of NP-Completeness.* Freeman, New York.

Graham, R. L. (1966). Bounds for certain multiprocessing anomalies. *Bell system technical journal*, *45*(9), 1563–1581.

Grandoni, F. (2006). A note on the complexity of minimum dominating set. *Journal of Discrete Algorithms*, *4*(2), 209–214.

Gurevich, Y., Stockmeyer, L., & Vishkin, U. (1984). Solving NP-hard problems on graphs that are almost trees and an application to facility location problems. *Journal of the ACM (JACM)*, *31*(3), 459–473.

Harary, F. (1969). *Graph Theory*. Addison-Wesley Publishing Company.

Harutyunyan, H. A. (2008). An efficient vertex addition method for broadcast networks. *Internet Mathematics*, *5*(3), 211–225.

Harutyunyan, H. A., & Liestman, A. L. (2012). Upper bounds on the broadcast function using minimum dominating sets. *Discrete Mathematics*, *312*(20), 2992–2996.

Harutyunyan, H. A., Pankratov, D., & Racicot, J. (2021). *Online domination: The value of getting to know all your neighbors.* arXiv preprint arXiv: 2105.00299.

Haynes, T. W., Hedetniemi, S., & Slater, P. (1998). *Fundamentals of domination in graphs*. CRC press.

Hedetniemi, S. T., Laskar, R., & Pfaff, J. (1986). A linear algorithm for finding a minimum dominating set in a cactus. *Discrete Applied Mathematics*, *13*(2-3), 287–292.

Jafari Rad, N., Mojdeh, D. A., Musawi, R., & Nazari, E. (2021). Total domination in cubic Knödel graphs. *Communications in Combinatorics and Optimization*, *6*(2), 221–230.

Johnson, D. S. (1974). Approximation algorithms for combinatorial problems. *Journal of computer*

and system sciences, *9*(3), 256–278.

Karlin, A. R., Manasse, M. S., Rudolph, L., & Sleator, D. D. (1988). Competitive snoopy caching. *Algorithmica*, *3*(1), 79–119.

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of computer computations* (pp. 85–103). Springer.

Khachatrian, L., & Haroutunian, O. (1990). Construction of new classes of minimal broadcast networks. In *Conference on Coding Theory, Armenia* (pp. 69–77).

Kikuno, T., Yoshida, N., & Kakuda, Y. (1980). The NP-completeness of the dominating set problem in cubic planar graphs. *IEICE TRANSACTIONS (1976-1990)*, *63*(6), 443–444.

Kikuno, T., Yoshida, N., & Kakuda, Y. (1983). A linear algorithm for the domination number of a series-parallel graph. *Discrete Applied Mathematics*, *5*(3), 299–311.

King, G., & Tzeng, W. (1997). On-line algorithms for the dominating set problem. *Inf. Process. Lett.*, *61*(1), 11–14.

Knödel, W. (1975). New gossips and telephones. *Discrete Mathematics*, *13*, 95.

Kobayashi, K. M. (2017). Improved bounds for online dominating sets of trees. In Y. Okamoto & T. Tokuyama (Eds.), *28th International Symposium on Algorithms and Computation, ISAAC 2017, December 9-12, 2017, Phuket, Thailand* (Vol. 92, pp. 52:1–52:13). Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Komm, D. (2016). *An Introduction to Online Computation - Determinism, Randomization, Advice*. Springer.

Li, S., & Pomerance, C. (2002). Primitive roots: a survey. In *Number theoretic methods* (pp. 219–231). Springer.

Lund, C., & Yannakakis, M. (1994). On the hardness of approximating minimization problems. *Journal of the ACM (JACM)*, *41*(5), 960–981.

McRae, A. A. (1996). Generalizing NP-completeness proofs for bipartite graphs and chordal graphs.

Mojdeh, D. A., Musawi, S., & Nazari, E. (2019). Domination critical Knödel graphs. *Iranian Journal of Science and Technology, Transactions A: Science*, *43*(5), 2423–2428.

Mojdeh, D. A., Musawi, S. R., & Nazari, E. (2018). Domination in 4-regular Knödel graphs. *Open*

Mathematics, *16*(1), 816–825.

Ore, O. (1962). *Theory of graphs, AMS colloq* (Vol. 1962).

Racicot, J., & Rosso, G. (2021). *Domination in Knödel graphs.* arXiv preprint arXiv: 2102.00505.

Shun, J., Roosta-Khorasani, F., Fountoulakis, K., & Mahoney, M. W. (2016). Parallel local graph clustering. *arXiv preprint arXiv: arXiv: 1604.07515*.

Sleator, D. D., & Tarjan, R. E. (1985). Amortized efficiency of list update and paging rules. *Communications of the ACM*, *28*(2), 202–208.

Van Rooij, J. M., & Bodlaender, H. L. (2011). Exact algorithms for dominating set. *Discrete Applied Mathematics*, *159*(17), 2147–2164.

Varghese, S., Vijayakumar, A., & Hinz, A. M. (2018). Power domination in Knödel graphs and Hanoi graphs. *Discussiones Mathematicae Graph Theory*(1), 63–74.

Xueliang, F., Xu, X., Yuansheng, Y., & Feng, X. (2009). On the domination number of Knödel graph W (3, n). *IJPAM*, *50*(4), 553–558.