Eternal Domination Problems

by

John Ethan Thomas Williams
BSc, University of Victoria

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

MASTER OF SCIENCE

in the Department of Mathematics and Statistics

We acknowledge with respect the Lekwungen peoples on whose traditional territory the university stands, and the Songhees, Esquimalt, and W̱SÁNEĆ peoples whose historical relationships with the land continue to this day.

Eternal Domination Problems

by

John Ethan Thomas Williams
BSc, University of Victoria

Supervisory Committee

Dr. Gary MacGillivray, Co-Supervisor
(Department of Mathematics and Statistics)

Dr. Richard Brewster, Co-Supervisor
(Department of Mathematics and Statistics, Thompson Rivers University)

ABSTRACT

Consider placing mobile guards on the vertices of a graph. The vertices are then attacked by an assailant, requiring you to move guards to the attacked vertices. What is the minimum number of guards you need in order to be able to defend against any sequence of attacks? This question is the basis for the *eternal domination problem*. In this thesis we investigate this problem and introduce new parameters related to it.

These new parameters arise from changing three of the assumptions made when defining the game. Specifically we assume that any number of guards can move when defending against an attack; only one attack needs to be defended against at a time; and that any number of guards can occupy a vertex. Changing these assumptions gives rise to the *maneuver*, *invasion*, and *stacking* numbers respectively. We investigate these parameters throughout this thesis, especially as they relate to trees.

Additionally, we tackle the related problem of *eternal Roman domination*, which is based on the topic which originally gave rise to the eternal domination problem. We establish a best possible upper bound for this parameter over all graphs. Finally, we present exponential time algorithms for solving all of these problems, as well as a host of other related problems.

Table of Contents

List of Figures

List of Algorithms

ACKNOWLEDGEMENTS

Firstly, I would like to thank Dr. Gary MacGillivray and Dr. Rick Brewster for the unwavering support they gave me during my studies here. Your advice and guidance continues to be invaluable, and gives me confidence to pursue further studies elsewhere knowing I can turn to you.

I would also like to thank my mom, dad, and sister for always supporting me and telling me to believe in myself. Knowing that I always have you guys to come back home to means a lot. Additionally, thank you Bennie and Zeni for often being the easiest people to talk to, and always being the best pets someone could ask for.

I also want to thank all of my friends, both here at university and back home. Accomplishing what I have would not have been possible without you. We have had some fun adventures, and I hope we have many more. In particular, I would like to thank Tyler for putting up with me and my ideas for so many years, and Dasha for remaining my friend and helping me open up to everyone else. Anika, you took the opportunity to become my friend when I needed one, and for that I have to thank you so much.

To everyone I met in grad school: thank you. Ashna, Natalie, and Akina, thank you for always being there when I needed to talk to someone, or a kick in the pants. Elena, thank you for being my best friend, and for always wanting to chat with me about anything and everything. I hope to see you again some time soon e-dawg. Nubia, you were the first person I spoke to when I started my masters here, and I could not have picked a better person. My ideas maybe didn't always pan out, but you would always listen to them. Thank you Georgia for listening to some of those very same ideas and helping me see them through to fruition. Chapter 4 would not exist without you. Finally, I would also like to thank Dr. Natasha Morrison for her support and belief in me. I would not have pursued these studies without your encouragement.

Chapter 1
The Eternal Domination Problem

1.1 Introduction

Consider a forest in the summer. Lightning strikes and unintelligent campers will occasionally start fires, which may grow out of control and threaten cities. Clearly one wishes to extinguish these fires before any cities are damaged, but if you have a limited number of firefighters, where are the best locations to station them?

This topic is one example of an application of the *eternal domination game*, which began as a model of military defense strategies. To play the game, one makes a graph (with vertices representing cities and edges connections between those cities), and then places "guards" (representing firefighting crews) on the vertices. An adversary then selects an infinite sequence of cities to attack (in this case, representing encroaching fires), requiring that you move a guard from an adjacent vertex to defend that location and possibly relocate other guards. Given this game, *eternal domination number* is the answer to the question of how many guards you need to defend against any sequence of attacks.

This problem has been well studied over the past twenty years, but in this thesis we present a variety of new problems and graph parameters related to it. These new problems all make the game harder for the defender; our focus lies on finding those cases for which the additional difficulty does not require additional guards to be dealt with.

1.2 Parameters

We quantify this tolerance for additional difficulty with three new graph parameters related to eternal domination. Each follows a general pattern: subject to the constraint that we are only given a particular number of guards, how far can we push some other related parameter? We will rigorously define each parameter below in its own chapter, but for now we provide a high-level overview:

- First, consider the scenario where there are limits on how many guards are allowed to move when defending against the latest attack. If many guards are allowed to move, then in principle one needs fewer guards, with a minimum achieved when all guards are allowed to move. The *maneuver number* is then the minimum number of guards which must be allowed to move in order to successfully defend against all attacks using the minimum number of guards overall.

- Next, consider the scenario where multiple vertices are attacked simultaneously. More guards are likely needed to defend against sequences of multiple attacks, but are there graphs for which we can make do with the same number of guards needed to defend against a single attack? The answer to this question is yes, and the *invasion number* quantifies this as the maximum number of simultaneous attacks per turn which can be defended against without needing additional guards.

- Finally, consider the scenario where multiple guards are allowed to occupy a single vertex. It may seem that this provides no advantage, but in fact there are graphs for which this reduces the total number of guards needed. The *stacking number* answers the question of how many guards must be allowed to occupy a vertex if one wants to have the minimum possible number of guards.

In the rest of this chapter we formally introduce the *eternal domination game* and the corresponding optimization problem. We also present a few ways of analyzing and constructing the strategies used when playing the game as well as some simple but highly useful results regarding the game.

1.3 Definitions

Throughout this thesis we assume that the graphs we are working with are finite, simple, and undirected. We begin by providing some standard graph theory definitions we will use, and refer the reader to [7] for any terms not defined in this thesis. Let $G = (V, E)$ be a graph and let $v \in V$. The **open neighbourhood** of v is the set $N(v) = \{y : vy \in E(G)\}$ and the **closed neighbourhood** of v is $N[v] = N(v) \cup \{v\}$. We can also define the open and closed neighbourhoods in relation to sets of vertices: if $S \subseteq V$ is a set of vertices, then the open and closed neighbourhoods of S are

$$N(S) = \bigcup_{s \in S} N(s) \setminus S,$$
$$N[S] = N(S) \cup S.$$

In other words, the open neighbourhood of a vertex v is the set of all vertices adjacent to v, and the closed neighbourhood is that same set along with v itself. The open neighbourhood of a set S is the set of all vertices adjacent to a vertex in S but

which are not in S themselves, and the closed neighbourhood of S is all vertices which are either in S or adjacent to a vertex in S.

Now let $D \subseteq V$ be some subset of the vertices. We say that D is a **dominating set** if for every vertex $v \in V$ either v is in D or v is adjacent to some vertex D. Equivalently a set D is a dominating set if $N[D] = V$. The size of a minimum dominating set is called **domination number** of a graph, and is denoted $\gamma(G)$ (or just γ when the graph being considered is clear).

We will often work with trees in this thesis, and usually want to restrict ourselves to some notion of the "bottom" of a tree. Let T be a tree and $x \neq y$ be adjacent vertices. Then the **subtree rooted at x with respect to y** is obtained by rooting T at y and considering the subtree rooted at x. This is equivalent to the subtree induced by the set of all vertices v for which x lies on the unique path from v to y. We denote this tree with T_x^y. Note that y is not in T_x^y.

As mentioned in the introduction, the eternal domination game is played with mobile guards placed on the vertices of a graph. If there are k guards on the graph, then we typically consider the guards to be labelled with $1, \ldots, k$. We call a placement of guards a **k-configuration**, which may formally be considered a function $c : V(G) \to \mathcal{P}([k])$ such that the set $\{c(v) : v \in V(G)\}$ forms a partition of $[k]$. Intuitively, this definition assigns some (possibly empty) subset of the guards to each vertex of the graph, and the partition condition necessitates that each guard is assigned to exactly one vertex.

We consider the guards in a k-configuration to be mobile, and say that two k-configurations c_1 and c_2 can **reconfigure** to one another if it is possible to move the guards in c_1 to adjacent vertices in order to form c_2. Formally, two configurations c_1 and c_2 can reconfigure to each other if for each guard $i \in [k]$ we have that if $i \in c_1(v)$ and $i \in c_2(u)$ then $u \in N[v]$. There are other equivalent ways to define reconfiguration which we explore later in this chapter.

We can now define the **eternal domination game** with k guards. This is a two-player game played with perfect information on a graph. One player, Alice, begins by placing k guards in G to form a k-configuration. The other player, Bob, then picks a vertex t to attack. If Alice can reconfigure her guards from c to some other k-configuration c' where the attacked vertex t has a guard on it, then she has **defended** against the attack on t.

If Alice can defend against any infinite sequence of attacks, then she wins. Otherwise, Bob wins. The **eternal domination number** is the least k such that Alice has a winning strategy with k guards, denoted $D(G)$. This definition of the eternal domination game explicitly makes the following assumptions:

- Alice may move any number of guards when defending against an attack, but each guard may only move once per attack.

- Only a single vertex is attacked at a time.
- Multiple guards are allowed to occupy the same vertex, and are said to be *stacking* when this happens.

In later chapters we investigate variations of the game where each of these assumptions are changed and define corresponding new parameters. Accordingly, we introduce notation which generalizes many of these notions before we explore each in depth.

We use the notation $\mathcal{P} = (G, I_\gamma, a, m, s)$ to denote an arbitrary **instance** of the eternal domination problem, where each parameter is defined as follows:

- G is the underlying graph the game is played on.
- I_γ is some property which all configurations of guards must satisfy.
- a is the number of simultaneous attacks one must defend against.
- m is the maximum number of guards allowed to move when reconfiguring.
- s is the maximum number of guards allowed to occupy a vertex.

We have additional commentary regarding this definition:

- One can think of I_γ as either an algorithm which determines whether or not the given configuration is valid, or as a set of all valid configurations.
- In the case where $a > 1$, the rules of the game change to Bob selecting a set of a distinct vertices to attack, and Alice must move her guards so that there is at least one guard on each attacked vertex in order to defend against the attack.
- We call a k-configuration **valid** with respect to an instance $\mathcal{P}$ if it places no more than s guards on any vertex and satisfies I_γ.

Finally regarding notation, we use $D(\mathcal{P})$ to denote the minimum number of guards needed for an eternal dominating strategy given the constraints imposed by $\mathcal{P}$; we sometimes use $D(G; \mathcal{P})$ when we wish to impose a set of constraints but leave G as a free parameter. Later we will abuse this notation and use, e.g. $D_A(G; a)$ to denote the number of guards needed for an eternal dominating strategy where there are a simultaneous attacks but every other parameter is as defined above.

Finally, we give a rigorous way of describing a winning strategy for Alice. Let an instance $\mathcal{P} = (G, I_\gamma, a, m, s)$ and integer k be given, and let $V_\mathcal{P}$ be the set of all valid configurations. We can construct a graph $\mathcal{R}_\mathcal{P}$ with $V_\mathcal{P}$ as its vertex set, and edges between any two configurations which can reconfigure to one another. We call this graph the $(k, \mathcal{P})$ **reconfiguration graph**. A winning strategy is then a subgraph Λ of $\mathcal{R}_\mathcal{P}$ such that for each configuration $d \in V(\Lambda)$ and $t \subseteq V(G)$ with $|t| = a$, one of the two following conditions is true:

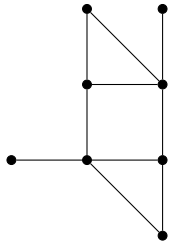


Figure 1.1: The graph on which Example 1.1 is played.

- Either $|d(v)| \geq 1$ for all $v \in t$; or
- d is adjacent to some configuration for which the first condition holds.

We call subgraphs where these conditions hold $(k, \mathcal{P})$ **strategies** for $\mathcal{P}$. We show in Lemma 1.2 that the existence of these strategies directly corresponds to a winning strategy for Alice. When k and $\mathcal{P}$ are clear from context (or not relevant), we refer to a winning strategy simply as an **eternal dominating strategy**.

1.3.1 A Note on Properties

The focus of much research in this area has been on ascribing different choices for I_γ, the property which each configuration of guards achieved must satisfy. The focus of this thesis is not on such problems though Chapter 5 does consider such a problem. Thus, throughout this thesis we will work with only a limited number of choices for I_γ. We list these below.

- We use U to denote the **universal property**, where any configuration is considered valid.
- We use γ to denote the **dominating property**, where each configuration c must have that the set $\{v : |c(v)| \geq 1\}$ is a dominating set of G.
- Finally, in Chapter 5 we use γ_R to denote the **Roman dominating property** where each configuration is a Roman dominating set. We define this more rigorously in that chapter.

Later in this chapter (Lemma 1.3) we will show that the universal property and the dominating property are in fact equivalent in some sense.

Example 1.1. We provide an example of the eternal domination game, and many of the above definitions. Let G be the graph shown in Figure 1.1. We will play in the instance $\mathcal{P} = (G, U, 1, \infty, \infty)$ where U denotes the universal condition (i.e. any set is allowed). This is the default assumption throughout this thesis. We begin by simulating some play of the eternal domination game with 4 guards. Alice first places

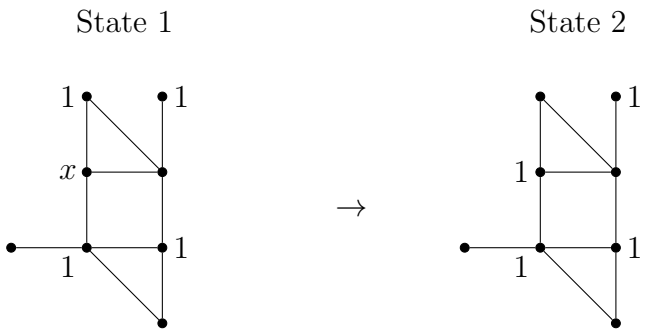


Figure 1.2: The first two steps of the eternal domination game played between Alice and Bob in Example 1.1.

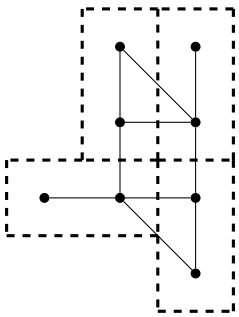


Figure 1.3: The partition of G used to describe the eternal dominating strategy in Example 1.1.

her guards on the 4-configuration shown in state 1 of Figure 1.2, where the numbers next to vertices represent the number of guards assigned to that vertex. Bob then attacks vertex x, and Alice responds by reconfiguring her guards to state 2 of Figure 1.2. Bob would then choose another vertex to attack, and Alice would again reconfigure her guards to defend against the attack.

Repeating this process to infinity is clearly an exhausting process, so we hope to find a more concise way to describe the strategy used by Alice. We can do so by first considering the partitioning of the graph given in Figure 1.3. Our idea for a strategy then is to place a single guard in each pair, and use that guard exclusively to defend against attacks. If there is an attack on an undefended vertex, then there is always some vertex paired with it which has a guard on it. If we move this guard to the attacked vertex to defend against the attack, then we can perhaps convince ourselves that this strategy is eternally dominating.

For the sake of rigour however, let us consider the $(4, \mathcal{P})$ reconfiguration graph $\mathcal{R}_{\mathcal{P}}$, and see if our described strategy is a $(4, \mathcal{P})$ strategy. If we label each 4-configuration of G with the multiset of vertices with guards on them, then we can see that $\mathcal{R}_{\mathcal{P}}$ has at least 8^4 states. We mention this to demonstrate that in practice $\mathcal{R}_{\mathcal{P}}$ is unwieldy, as

many of these configurations are clearly terrible choices for use in an eternal dominating strategy. But how about the subgraph induced by the states we used in the strategy described above?

For that strategy, let V_D be the set of possible configurations we use in our strategy. Notice that each pair has 2 possible positions for its guard, and so we have that $|V_D| = 16$. Looking at this set in terms of the reconfiguration graph, note that every state in V_D can transition to every other state in V_D. This leads us to believe that $\mathcal{R}_{\mathcal{P}}[V_D]$ is isomorphic to a complete graph, which is correct. It is not hard to verify that this subgraph satisfies the conditions to be a $(4, \mathcal{P})$ strategy. In Lemma 1.2 we show that in fact eternal dominating strategies and $(k, \mathcal{P})$ strategies are equivalent, and so we can conclude that we did in fact describe an eternal dominating strategy.

Finally, note that since we have provided an eternal dominating strategy which uses 4 guards, we have that $D(\mathcal{P}) = D(G; \mathcal{P}) \leq 4$. We leave it to the reader to convince themselves that $D(\mathcal{P}) > 3$, and so conclude that $D(\mathcal{P}) = 4$.

1.4 Previous Work

Eternal protection problems trace their origins to the military strategy of Constantine the Great of the Roman Empire. Faced with the problem of protecting the empire at its largest extent but also lacking the resources of former emperors, he decided to be more strategic about where he stationed his troops in the field. The military was divided into two groups: *limitanei* and *legions*. The limitanei were recruited locally and were paid less wages, whereas the legions were professional soldiers hailing from around the empire. The limitanei could only reasonably be expected to defend the regions in which they lived, but the legions could be used to respond to distant attacks. By carefully stationing a mix of legions and limitanei around the empire, Constantine could in theory defend his empire more efficiently than preceding emperors[1].

A mathematical formulation of this strategy is the subject of *Roman domination*, a variant of the classical *dominating set* problem, which was first introduced by Stewart in [23] based on a paper by ReVelle and Rosing in [22]. A given area one desires to defend is represented as a graph, and the placement of groups of soldiers is represented by assigning guards of particular strengths to the vertices of the graph. A Roman dominating set allows placing guards of strength 1 (representing limitanei) or 2 (representing legions). Any vertex left unguarded must be adjacent to a vertex containing a legion, as were an enemy were to attack that province a legion must be nearby in order to provide defense. However, Roman dominating sets do not account for the fact that the act of moving a legion to a new region may leave another undefended against

[1]The preceding paragraph is likely not a historically accurate representation of Roman military strategy. It is however sufficient for motivating the mathematical problem.

a subsequent attack. To remedy this, we turn to a dynamic model of domination.

Unlike in Roman dominating sets, eternal dominating sets no longer distinguish between having just one or more than one guard on a vertex. However, the guards are allowed to (and in fact, must) move when a vertex is attacked. First introduced by Burger et al. in [5], the eternal domination problem was defined as above, but with the additional restriction that only a single guard move in response to each attack. They showed that the number of guards needed is bounded above by the clique covering number and below by the independence number of the graph. Following this, Goddard et al. in [11] considered the variant where all guards are allowed to move. They showed that in this case the number of guards needed is bounded above (rather than below) by the independence number.

It was conjectured by Goddard et al. that there would be no advantage to allowing multiple guards to occupy a vertex during a single round of the eternal domination game when all guards are allowed to move [11]. This was shown to be false by Finbow et al. when they provided a construction of a graph which can be defended with fewer guards when 2 or more guards are allowed to occupy a single vertex [9].

Different domination requirements were first investigated in 2009 by Klostermeyer and Mynhardt where they required that every set be a vertex cover [17] (called edge protection), followed by a paper where they looked at variants where a connected or total dominating set must be maintained [18]. Other variants investigated include independent dominating sets and paired dominating sets[12, 21]. Finally, Hoepner has investigated the variant where a Roman dominating set must be maintained, bringing the field back to where it began [13].

In addition to the variants presented thus far, some also change what it means for a vertex to be attacked. Only two models of attacking have been investigated thus far: the classical attacks model (previously discussed) and eviction. In the eviction model, vertices containing guards are targeted, and Alice must move the guards to form a dominating set which does not contain the targeted vertex if possible. This problem was first investigated by Klostermeyer et al. [14]. They investigated the one guard moves and all guards move variants, and proved that the number of guards needed in the one guard moves model is at most the clique covering number (formally defined below), amongst other results.

Algorithmically, the question of determining if a graph has an eternal dominating set of size at most k is a difficult one. For both our default model and the one guards model, it is not even known whether there exists an algorithm that solves the problem in polynomial space [19]. However, some strategies exist for solving the problem in exponential time. When all guards are allowed to move during reconfiguration, there is a linear time algorithm for determining the eternal domination number when the underlying graph is a tree [15]; similar results have been found for other variants of the problem [21].

The research presented so far has focused on determining the minimum number of guards needed so that Alice has a winning strategy, but one can also look at the sequences of attacks which allow Bob to win when there are not enough guards. This was first investigated by Blazej et al. in [2], where they showed that for trees on n vertices, such sequences have length at most n. The length of the game can also be determined using the method developed by Bonato and MacGillivray in [4], where they show that for k guards, winning sequence for Bob have length at most n^{k+1}.

1.4.1 A Note on Existing Notation

Existing literature on the eternal domination problem uses a variety of notation which is largely different than the notation we use throughout this thesis. The reason for our choice of notation is simply that we introduce more parameters than can reasonably fit in subscripts and superscripts[2]. In this section we provide some examples of how our notation captures existing notation in the field.

Our default assumption throughout this thesis is that Alice is allowed to move all guards when responding to an attack. The first paper in this area allowed only a single guard to move in response to an attack, and also allowed only a single guard to occupy a vertex. This version, introduced by Burger et al. in [5] used the notation $\gamma_\infty(G)$ to denote the minimum number of guards needed for Alice to have a winning strategy. This corresponds to the problem instance $\mathcal{P} = (G, \gamma, 1, 1, 1)$, and thus in our notation, either $D(\mathcal{P})$ or $D(G; \mathcal{P})$.

A later model allowed all guards to move in response to an attack. In the literature either the notation $\gamma_m^\infty(G)$ or $\gamma_{all}^\infty(G)$ is used for the minimum number of guards needed. Either notation corresponds to the problem instance $\mathcal{P} = (G, \gamma, 1, \infty, 1)$ or $\mathcal{P} = (G, \gamma, 1, \infty, \infty)$ where the former indicates that at most one guard is allowed on a vertex and the latter allows an arbitrary number of guards to occupy a vertex.

The standard model throughout this thesis is corresponds to the instance $\mathcal{P} = (G, U, 1, \infty, \infty)$, and thus $D(G) = D(G; \mathcal{P})$. Explicitly, we allow all guards to move, any number of guards to occupy a vertex, and the guards to form any configuration they desire. Despite seeming less restrictive, we show later in this chapter that this model is equivalent to the the existing all-guards move model.

1.5 First Results

We begin by presenting a few simple results related to the eternal domination number. We begin with a few general results which apply to any instance of the problem, and then focus specifically on our default model, $\mathcal{P} = (G, U, 1, \infty, \infty)$.

[2]My first idea for solving this problem involved adopting notation from chemistry, i.e. ${}_b^a\gamma_d^c$. This was quickly rejected, both by me, and all sane people I spoke to (which may not include myself).

We begin by showing that $(k, \mathcal{P})$ strategies correspond directly to winning strategies for Alice, and then show that any state in such a strategy must place the guards so that they form a dominating set. This gives us a lower bound on the eternal domination number. We then make the simple observation that one can restrict their view to connected graphs when computing the eternal domination number.

Lemma 1.2. *For a given instance $\mathcal{P}$ and integer k, Alice has a winning strategy with k guards in the variant of the problem defined by $\mathcal{P}$ if and only if a $(k, \mathcal{P})$ strategy exists.*

Proof. We first show that the existence of a $(k, \mathcal{P})$ strategy corresponds to the existence of a winning strategy for Alice. To see this, let Λ be a $(k, \mathcal{P})$ strategy; we demonstrate a winning strategy for Alice. At the start of the game, Alice picks a state from $V(\Lambda)$. We maintain the invariant that the k-configuration of guards is always a vertex in Λ. Now suppose that the guards currently form a k-configuration, c. We now show that for any set t of a attacked vertices, Alice can always defend against the attacks. She can do this in the following ways (choosing arbitrarily if both are valid choices):

- If every vertex in t already has a guard on it, Alice does nothing.
- Otherwise, by definition of a $(k, \mathcal{P})$ strategy, there is some state neighbouring d in Λ for which every vertex in t has a guard stationed on it. Alice then reconfigures to one such state.

Such a reconfiguration is possible by the definition of edges in Λ. This maintains the invariant that we always use a configuration which is in Λ, and so Alice has an eternal dominating strategy.

We now show that the existence of an eternal dominating strategy corresponds to the existence of a $(k, \mathcal{P})$ strategy. Suppose we have played some instance of the eternal domination game, and determined that Alice can win with k guards. Let k_0 be the initial state Alice chooses. We call a configuration k_i achievable if either $i = 0$, or there is some configuration k_{i-1} which is achievable and there is some attack t such that Alice's response moves the guards to k_i.

Let V_D be the set of achievable configurations. Since there are at most $|V(G)|^k$ possible k-configurations, we know that V_D is finite. Let $\mathcal{R}_\mathcal{P}$ be $(k, \mathcal{P})$ reconfiguration graph, and consider the subgraph $\Lambda = \mathcal{R}_\mathcal{P}[V_D]$. We show that Λ satisfies the conditions necessary to be a $(k, \mathcal{P})$ strategy.

To see this, first notice that by definition of V_D and $\mathcal{R}_\mathcal{P}$, the configurations achieved by Alice are all present in V_D, and any reconfiguration she did is present as an edge. So suppose for contradiction that there is some state $c \in V_D$ which fails the conditions of a $(k, \mathcal{P})$ strategy. Then we must have that there is some t and some vertex in $v \in t$ for which $|c(v)| = 0$, and for each adjacent state $c' \in \Lambda$ we also have $|c'(v)| = 0$. As c is achievable, there must be some sequence of attacks Bob can make so that Alice places

the guards in the configuration c. If he then attacks t, this would imply that Alice loses the game, which contradicts that she wins. Thus there must be some adjacent state c' which places a guard on each vertex in t and is present in V_D. But this clearly contradicts our assumption. The result follows. □

Lemma 1.3. *Let Λ be a $(k, \mathcal{P})$ strategy for some k and $\mathcal{P}$. Then for every configuration $d \in V(\Lambda)$ the set $\{v : |d(v)| \geq 1\}$ is a dominating set.*

Proof. Suppose for contradiction that this is not true. Then there exists some state $d \in V(\Lambda)$ such that $D = \{v : |d(v)| \geq 1\}$ is not a dominating set. Let $u \in V(G)$ be a vertex which is not dominated by D. If Bob attacks u, then Alice cannot possibly move a guard to u, contradicting that Λ is a $(k, \mathcal{P})$ strategy. □

Corollary 1.4. *[11] For any graph G, we have that $\gamma(G) \leq D(G)$.*

Proof. This follows directly from Lemma 1.3. □

Lemma 1.5. *Let G be a graph with connected components $G_1, \ldots, G_k$. Then $D(G) = \sum D(G_i)$.*

Proof. Clearly we have that $D(G) \leq \sum D(G_i)$. Now if $D(G) < \sum D(G_i)$ there must be some component G_j which has strictly fewer than $D(G_j)$ guards. Since guards cannot move between components, this cannot be an eternal dominating configuration. □

We now present eternal domination strategies for particular classes of graphs, and then extend these to results about all graphs. Our first such strategy concerns cliques. Clearly one can dominate all the vertices in a complete graph with a single guard, and without much effort, this can be extended to an eternal domination as well. Extending this, recall that a **clique covering** of a graph is a partition of the vertices such that each set induces a clique. The minimum size of a clique covering is the **clique covering number**, denoted $\theta(G)$. We can then form an eternal dominating strategy for the entire graph by placing a single guard in each of these cliques.

Lemma 1.6. *Let G be a complete graph. Then,*

$$D(G) = D(G, \gamma, 1, \infty, \infty) = D(G, \gamma, 1, 1, \infty) = D(G, \gamma, 1, 1, 1) = 1$$

Proof. Place 1 guard on G. For each attack this guard can move to the attacked vertex, and clearly we always maintain a dominating set. □

Corollary 1.7. *[5] For any graph G we have $D(G) \leq \theta(G)$.*

Proof. Let $\mathcal{C}$ be a minimum clique of covering of G. We demonstrate a winning strategy for Alice: place a guard in each clique in $\mathcal{C}$. Whenever a vertex is attacked, we can restrict our view of the graph to the clique containing that vertex, and by Lemma 1.6 we can defend against the attack. This strategy uses $\theta(G)$ guards, and the result follows. □

Our next result gives us another upper bound, this time in terms of the domination number of G.

Lemma 1.8. *Let G be a graph and v a vertex. Then $N[v]$ can eternally dominated with 2 guards.*

Proof. Place 1 guard on v and 1 in its neighbourhood and maintain this condition as an invariant. If there is an attack in the closed neighbourhood of v, move the guard off of v to the attacked vertex and move the other guard to v, maintaining the invariant. Since we can do this for any sequence of attacks, the result follows. □

Corollary 1.9. *[19] For any graph G, we have $D(G) \leq 2\gamma(G)$.*

Proof. Let S be a minimum dominating set of G. We demonstrate an eternal dominating strategy G. For each vertex in $v \in S$ we begin by placing a guard on v and some neighbour of v. Suppose a vertex t has been attacked. We know that t lies in the closed neighbourhood of some $u \in S$, and by Lemma 1.8 we can reconfigure our guards to defend against this attack. The result follows. □

We now present a final pair of simple results, which intuitively allows us to move guards "long" distances under certain conditions. The guard seems to travel from one end of a path to another, but in reality we are shifting guards in a particular direction along a path in the graph. This lemma proves to be incredibly useful for many of our results, and can be used to find another simple upper bound on the eternal domination number. Recall that a **connected dominating set** is a dominating set whose vertices induce a connected graph. The minimum size of such a set is denoted as $\gamma_c(G)$ and called the **connected domination number**. Given a connected dominating set, we can add a single additional guard and obtain an eternal dominating set.

Lemma 1.10. *Let G be a graph, c be a k-configuration, and x and y vertices such that there is a path $P = (x, v_1, \ldots, v_t, y)$. Suppose there is a guard on x and on v_i for all i. Then c can reconfigure to the configuration c' which has a guard on each v_i and y.*

Proof. Let $P = (x, v_1, \ldots, v_t, y)$ be the path as described in the statement. We make the following moves:

- Move the guard on v_t to y.
- Move the guard on x to v_1.
- For each $1 \leq i < t$, move the guard on v_i to v_{i+1}.

This gives us the configuration c' as described in the statement, and the result follows. □

Corollary 1.11. *Let c_1 and c_2 be two configurations such that c_1 can reconfigure to c_2 by applying Lemma 1.10 to move guards along a path from x to y. Let $D_1 = \{v : |c_1(v)| \geq 1\}$ and $D_2 = \{v : |c_2(v)| \geq 1\}$. If $D_1 \setminus \{x\}$ is a dominating set, then D_2 is also a dominating set.*

Proof. Note that $D_2 = D_1 \setminus \{x\} \cup \{y\}$ as we have applied Lemma 1.10. Then clearly $D_1 \setminus \{x\} \subseteq D_2$, so D_2 is a dominating set. □

Corollary 1.12. *[11] Let G be a graph. Then $D(G) \leq \gamma_c(G) + 1$.*

Proof. Let S be a minimum connected dominating set of G, we demonstrate an eternal dominating strategy for G. Begin by placing guards on every vertex of S, and then place a guard on some vertex x of G not in S. We refer to this final guard as the "spare" guard. We maintain as an invariant that there is always a guard on each of the vertices in S. Now suppose the spare guard is located on the vertex v and some vertex u has been attacked. If $u \in S$ then we do nothing. If $u \notin S$, then let v_1 and v_t be vertices which dominate v and u respectively. As S induces a connected graph, there is a path from u to v which satisfies the conditions of Lemma 1.10. Therefore, we can reconfigure to the configuration which places a guard on u and still has a guard on every vertex of S. We now designate the guard on u as the spare guard, and since the invariant has been maintained, the result follows. □

1.5.1 Reconfiguration

In this final section, we explore two alternate definitions of reconfiguration between configurations c_1 and c_2. A **moves list** is a list M of ordered pairs of vertices such that for each $v \in V$, we have that v is the first coordinate in at most $|c_1(v)|$ pairs and the second coordinate in at most $|c_2(v)|$ pairs. A pair (x, y) in a moves list corresponds exactly to moving a guard from x to y. Additionally, note that the pairs in a moves list M can be taken to be the edge set of an oriented multigraph on the same vertex set, which we refer to as the **movement digraph** and denote as $MD(G, M)$. Recall that $deg_F^-(v)$ and $deg_F^+(v)$ denote the in and out degree of the vertex v in F respectively.

Lemma 1.13. *Let c_1 and c_2 be two k-configurations. Then c_1 can reconfigure to c_2 if and only if there is a moves list M with corresponding movement digraph $F = MD(G, M)$ such that for each vertex $v \in V(G)$ we have that*

$$deg_F^+(v) - deg_F^-(v) = |c_2(v)| - |c_1(v)|$$

Proof. First suppose that such an M exists. Beginning in c_1, if all the moves in M are made, then for each vertex we have that $deg_F^-(v)$ guards move on to v and $deg_F^+(v)$ guards move out of v. Thus, the number of guards on v changes by exactly

$deg_F^+(v) - deg_F^-(v) = |c_2(v)| - |c_1(v)|$. As we began with $|c_1(v)|$ guards, this means we now have $|c_1(v)| + |c_2(v)| - |c_1(v)| = |c_2(v)|$ guards on v. The result follows.

Now suppose that c_1 can reconfigure to c_2. Then it is possible to move the guards beginning from c_1 and obtain c_2. Let M be the corresponding moves list and $F = MD(G, M)$ the corresponding movement digraph. Suppose for contradiction that there is some vertex v such that $deg_F^+(v) - deg_F^-(v) \neq |c_2(v)| - |c_1(v)|$. However as noted above, the net change in the number of guards assigned to a vertex is exactly $deg_F^+(v) - deg_F^-(v)$. As we began with $|c_1(v)|$ guards, we now have $|c_1(v)| + deg_F^+(v) - deg_F^-(v) \neq |c_2(v)|$ guards assigned to v. This however contradicts that c_1 has reconfigured to c_2. □

Another equivalent way of characterizing reconfigurations is to look for matchings in a particular bipartite graph. In particular, let c_1 and c_2 be two k-configurations where v_i and u_i denote the location of the guard i in c_1 and c_2 respectively. Define the **bipartite movement graph** as $B(c_1, c_2) = (C_1, C_2, E)$ where $C_1 = \{(i, v_i) : i \in [k]\}$ and $C_2 = \{(i, u_i) : i \in [k]\}$ and there is an edge between (i, v_i) and (j, u_j) if $v_i u_j \in E(G)$ or if $v_i = u_j$. Note that this graph has $|C_1| = |C_2| = k$. We show that reconfiguration between c_1 and c_2 is equivalent to finding a perfect matching in $B(c_1, c_2)$.

Lemma 1.14. *Let c_1 and c_2 be two configurations. Then c_1 can reconfigure to c_2 if and only if $B(c_1, c_2) = (C_1, C_2, E)$ contains a perfect matching.*

Proof. First suppose that a perfect matching K exists. Beginning in c_1, for each edge $(i, v_i)(i, u_i) \in K$, move the guard i to the vertex u_i. Since u appears in exactly $|c_2(u)|$ vertices of C_2 and we have a perfect matching we have that exactly $|c_2(u)|$ guards move to u in this process. This applies for all vertices, and so the guards have moved to form c_2.

Now suppose that c_1 and c_2 can reconfigure to each other, i.e. we can move the guards from c_1 and obtain c_2. We will construct a perfect matching K. Consider the moves taken to go from c_1 to c_2 sequentially, and whenever the guard i moves from the vertex x to y, add the edge $(i, x)(j, y)$ to K where j is chosen arbitrarily so that $(j, y) \in C_2$ and (j, y) appears in exactly one edge (we will prove this is possible). Finally, for each vertex $(i, x) \in C_1$ which is currently unmatched add the edge $(i, x)(j, x)$ where j is chosen similarly.

We first prove that it is always possible to choose the j needed as above. Note that for any u there are exactly $|c_2(u)|$ vertices in C_2 with u in them, and as such we have a problem if we try to add more than $|c_2(u)|$ edges which go to (j, u) for some j. If this happens though, it implies that we now have more than $|c_2(u)|$ guards on u, which contradicts that we have reconfigured to c_2. Similarly, if there is some (j, u) which is not matched by K, then this implies that we now have fewer than $|c_2(u)|$ guards on u, which is again a contradiction.

Thus we can always construct K. We now show that it is a perfect matching. The above result shows that each vertex in C_2 is matched to exactly one vertex in C_1. Since $|C_2| = |C_1|$, this implies that K is a perfect matching. $\square$

We make one final note that in Lemma 1.14 the labels of the guards matter, whereas in reality we typically do not care. To account for this, note that permuting the labels of the guards does not affect the validity of any of configurations we are concerned with in this thesis. We implicitly use this fact in many proofs where we relabel guards arbitrarily.

Chapter 2
Maneuver Numbers

2.1 Introduction

In this chapter we investigate the first of our new eternal domination parameters, the *maneuver number* of a graph. The two standard forms of eternal domination lie at opposite extremes in terms of how many guards are allowed to move: either just a single guard, or all guards. If we let $D_M(G;k)$ denote the number of guards needed when at most k guards are allowed to move, then $D_M(G;1)$ and $D_M(G;\infty)$ denote the minimum number of guards needed in the on guard moves and all guards move models.

However, there is nothing preventing one from investigating, say, the three guards move model $D_M(G;3)$. Observe that any such intermediate model will need at most $D_M(G;1)$ guards, as one can always copy a previously used strategy. This leads to the natural definition of the *maneuver number*: what is the minimum k satisfying $D_M(G;k) = D_M(G;\infty)$? Or put more explicitly, given $D_M(G;\infty)$ guards, what is the maximum number of guards that must move in a single round, minimized over all strategies?

Our focus in this chapter is showing that the maneuver number of a tree can be determined in linear time.

2.2 Definitions

Let G be a graph and Λ an eternal dominating strategy for G. The **maneuver load** of Λ is the maximum number of guards which move when transitioning between adjacent states in Λ. We denote this as $m(\Lambda)$. The **maneuver number** of G is the minimum maneuver load among all eternal dominating strategies which use $D_M(G;\infty)$ guards, denoted as $M(G)$. An equivalent definition of the maneuver number is the minimum k such that $D_M(G;k) = D_M(G;\infty)$. We say an eternal dominating strategy Λ is **maneuver optimal** if we have that $m(\Lambda) = M(G)$.

2.3 Neocolonizations

Many of our results in this chapter rely on the notion of a *neocolonization*. These were first introduced by Goddard et al. when studying the eternal domination problem on trees [11].

Let G be a graph. Then a **neocolonization** $N = \{B_1, \ldots, B_t\}$ is a partition of $V(G)$ such that $G[B_i]$ is a connected graph for each i. We refer to the sets B_i as the **blocks** of N, and assign them a **weight** as follows:

$$w(B_i) = \begin{cases} 1 & \text{if } G[B_i] \text{ is a clique} \\ \gamma_c(G[B_i]) + 1 & \text{otherwise} \end{cases}$$

The **weight** of a neocolonization is then the sum of the weights of each of its blocks, denoted as $w(N)$. The minimum weight of a neocolonization of G is denoted as $\theta_c(G)$. Observe that the bounds on the eternal domination number presented in Lemma 1.7 and 1.12 can both be considered to be based on neocolonizations, just ones in which either all blocks are cliques, or there is only a single block.

2.3.1 Derived Strategies

The primary use for neocolonizations is in providing eternal dominating strategies for graphs (particularly trees). Let $N = \{B_1, \ldots, B_t\}$ be a neocolonization of a graph G. Then the **strategy derived** from N, Λ_N, is defined as follows: for each B_i such that $G[B_i]$ is not a clique, let C_i be the vertices in a minimum connected dominating set of $G[B_i]$ and $L_i = B_i \setminus C_i$. For each $1 \leq i \leq t$ let S_i be B_i if $G[B_i]$ is a clique and L_i otherwise. Define D as:

$$D = S_1 \times S_2 \times \ldots \times S_t$$

For each vector in $v \in D$, we add a configuration to Λ_N which places a guard on each vertex in v, as well as on each vertex in C_i for all C_i. We now show that this is an eternal dominating strategy on T which does not move too many guards.

Lemma 2.1. *Let G be a graph, N a neocolonization of G, and Λ_N the strategy derived from N. Then Λ_N is an eternal dominating strategy with*

$$m(\Lambda_N) = \max_{B_i \in N} diam(T[B_i])$$

Proof. Let D be as above and let $m = \max_{B_i \in N} diam(T[B_i])$. For any vector $v \in D$ let d_v be the configuration obtained from this vector. We first demonstrate some of the reconfigurations possible in Λ_N. Let u and v be any two vectors in D with corresponding

configurations d_u and d_v. We claim d_u can reconfigure to d_v while moving at most m guards if u and v differ on exactly one coordinate.

Let i be the coordinate on which u and v differ and let x_u and x_v be the values of each vector at that coordinate respectively. If $G[B_i]$ is a clique, then it is apparent that d_u can reconfigure to d_v while moving only a single guard. If $G[B_i]$ is not a clique, then x_u and x_v are in L_i by definition. We also know that each configuration in Λ_N places a guard on the vertices of a minimum connected dominating set of $G[B_i]$, and so by Lemma 1.10 d_u and d_v can reconfigure to each other while moving at most $diam(G[B_i])$ guards.

We now show that Λ_N is an eternal dominating strategy. Suppose we are currently in some configuration d and some vertex p has been attacked. Let B_i be the block of N containing p. If $G[B_i]$ is a clique we can reconfigure to the state which places a guard on p by the above. If $G[B_i]$ is not a clique, then we may assume that $p \in L_i$ as there is always a guard on each vertex of C_i. By construction there is some vertex $l \in L_i$ which has a guard on it, and by the above we can reconfigure to the state which places a guard on p. Thus Λ_N is an eternal dominating strategy. □

Theorem 2.2. *[11] For any graph G we have $D(G) \leq \theta_c(G)$.*

Proof. This follows directly from Lemma 2.1. □

The upper bound in Theorem 2.2 is especially an improvement when applied to trees, for which Klostermeyer and MacGillivray showed that this is best possible [15]:

Theorem 2.3. *[15] Let T be a tree. Then $D(T) = \theta_c(T)$.*

This result was shown using two reductions which can be applied to a tree. Let T be a tree and $x \neq y$ adjacent vertices in T. Recall that the **subtree rooted at x with respect to y** is obtained by rooting T at y and considering the subtree rooted at x, and is denoted T_x^y.

The reductions below generalize the ones used by Klostermeyer and MacGillivray. The results stated below are in relation to the original definitions, but the proofs are similar for these new definitions.

Let x and y be adjacent vertices and T_x^y the subtree rooted at x with respect to y. Then the reduction **R1** removes all leaves adjacent to x and can be applied if T_x^y is isomorphic to a star with at least 2 leaves. The reduction **R2** can be applied if $T_x^y \cong K_2$, and deletes both x and the leaf adjacent to it. See Figure 2.1 for an example illustrating each reduction. The eternal domination number can be computed recursively using these reductions:

Theorem 2.4. *[15] Let T be a tree.*

- *If we apply R1 to T and obtain H, then $D(T) = D(H) + 1$.*
- *If we apply R2 to T and obtain H, then $D(T) = D(H) + 1$.*

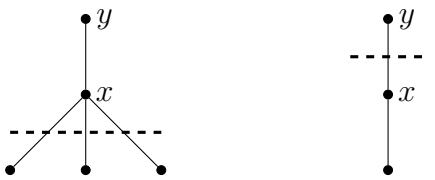


Figure 2.1: An example of R1 and R2 being applied to a tree.

2.3.2 Neocolonization Refinements

We require a handful of further definitions for our work with neocolonizations. Let $N = \{B_1, \ldots, B_t\}$ be a neocolonization of some graph G. We say that N can be **refined** if there exists another neocolonization $N' = \{B'_1, \ldots, B'_k\}$ such that we have:

- $w(N') \leq w(N)$;
- and for each $B'_i \in N'$ there exists $B_j \in N$ with $B'_i \subseteq B_j$.

In other words, if N is a neocolonization then a refinement of N is a neocolonization which is a refinement of the underlying partition such that the weight does not increase. If there does not exist a neocolonization N' which is a refinement of N and has $|N'| > |N|$, then we say that N is **locally finest**. Locally finest neocolonizations lie at the heart of many of our results.

We now define an auxiliary graph which we use to study neocolonizations. Let $N = \{B_1, \ldots, B_t\}$ be a neocolonization of a graph G. We say that two blocks B_i and B_j are **adjacent** if there exist vertices $v_i \in B_i$ and $v_j \in B_j$ such that $v_i v_j \in E(G)$. The **block adjacency graph** of G with respect to N is then the graph where each vertex is a block in N with edges between adjacent blocks. We denote this graph as $BA(G, N)$. Note that a graph does not necessarily have a unique minimum neocolonization, and thus the block adjacency graph is not necessarily unique.

We will exclusively work with neocolonizations of trees. In such cases it is helpful to note neocolonizations of trees produce block adjacency graphs which are also trees.

Lemma 2.5. *Let T be a tree and N a neocolonization of T. Then $BA(T, N)$ is also a tree.*

Proof. The graph $BA(T, N)$ is obtained from T be contracting each edge uv where u and v are in the same block of N. Contracting an edge cannot introduce a cycle nor disconnect a graph. □

Example 2.6. We end with a brief example of the above definitions. Figure 2.2 shows a tree T as well as two optimal neocolonizations of T. Both neocolonizations have

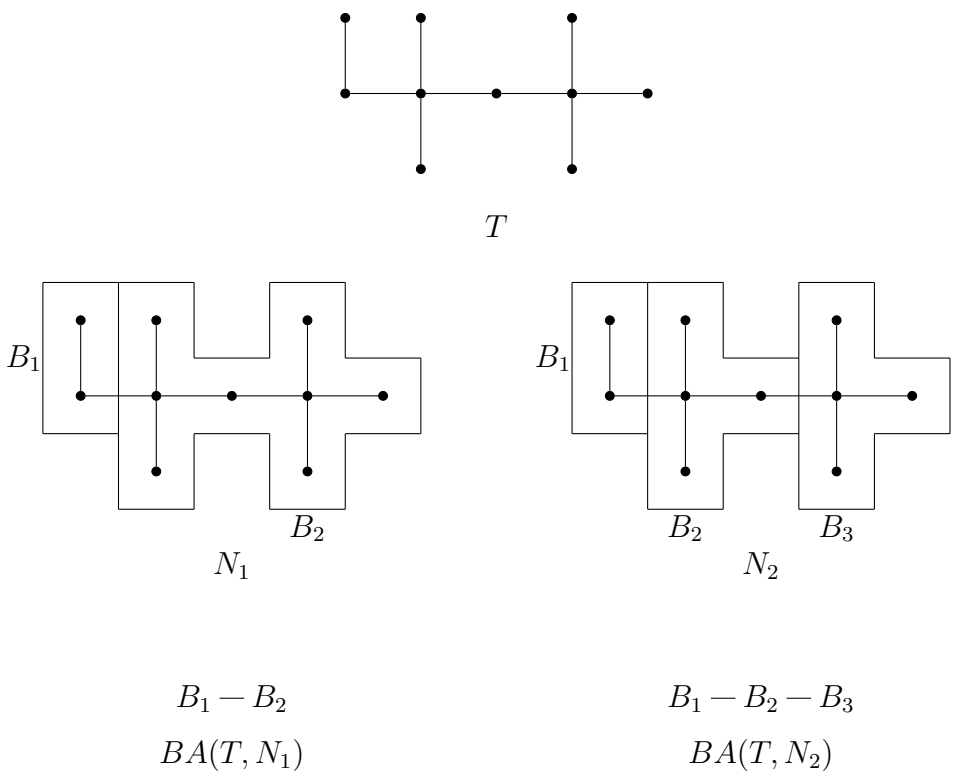


Figure 2.2: A tree T, an optimal and two locally finest neocolonizations N_1 and N_2 of T, and the block adjacency graph of T with respect to N_1 and N_2.

a weight of 5, which is optimal for T. However note that N_2 is a refinement of N_1 and thus N_1 is not locally finest. One can convince themselves that N_2 has no further refinements (aside from the trivial one), and thus is locally finest.

We now construct the block adjacency graphs for T with respect to N_1 and N_2. The blocks $B_1 \in N_1$ and $B_2 \in N_1$ are adjacent to each other, and are also the only blocks in N_1, so $BA(T, N_1) \cong K_2$. Similarly, N_2 has three blocks, with $B_2 \in N_2$ adjacent to both of the others. These are the only edges between blocks in N_2 and thus $BA(T, N_2) \cong P_3$.

2.4 Results

We begin with some simple observations regarding the maneuver number of a graph:

Theorem 2.7. *Let k be some integer. Then for any graph G we have,*

$$D_M(G; \infty) \leq D_M(G; k) \leq D_M(G; 1)$$

We can apply existing results for $D_M(G; 1)$ and $D_M(G; \infty)$ to quickly characterize those graphs which have $M(G) = 1$. Let $\alpha(G)$ denote the independence number of the

graph G. It was shown by Burger et al. that $\alpha(G) \leq D_M(G;1)$ [5]. Further, it has also been shown that $D_M(G;\infty) \leq \alpha(G)$ [11].

Lemma 2.8. *Let G be a graph. Then $M(G) = 1$ if and only if*

$$D_M(G;1) = D_M(G;\infty) = \alpha(G)$$

Proof. If $D_M(G;\infty) = D_M(G;1)$ then $M(G) = 1$ by definition. If $M(G) = 1$, then we know that $D_M(G;\infty) = D_M(G;1)$ by definition. By the inequalities presented above this is only possible if both equal $\alpha(G)$. □

We spend the remainder of this chapter characterizing the maneuver numbers of trees. We first characterize how these numbers change with regard to the reductions R1 and R2, and then provide a simple linear time algorithm which computes the maneuver number of a tree.

2.4.1 Reductions & Maneuver Numbers

Let T be a tree. Our goal in this section is to bound $M(T)$ in terms of $M(H)$ when H is obtained by applying either R1 or R2. We prove upper and lower bounds for each in terms of $M(H)$, and then characterize when certain bounds are achieved.

Lemma 2.9. *Let T be a tree. If we apply R1 and obtain H, then $M(T) \leq M(H) + 1$.*

Proof. Suppose we can apply R1 to some vertex x to remove leaves $L = \{l_1, \ldots, l_t\}$ and obtain H. Let Λ_H be a maneuver optimal strategy for H. We construct a new strategy Λ_T for T based on Λ_H and show that this strategy moves at most $M(H) + 1$ guards.

Let $k = D_M(H;\infty)$ and note that since we have applied R1 we know that T requires an additional guard by Theorem 2.4. We will label this guard $k + 1$. For each $c \in \Lambda_H$ we add the following states to Λ_T:

- If c does not place a guard on x (i.e. $|c(x)| = 0$), then let c_x be the state which places all guards in $[k]$ according to c, and places $k + 1$ on x. Add c_x to Λ_T.
- If c places a guard on x (i.e. $|c(x)| \geq 1$), then for each $1 \leq j \leq t$ let c_j be the state which places all guards in $[k]$ according to c, and places $k + 1$ on l_j. Add each c_j to Λ_T.

Adding the guards in this way preserves many of the reconfigurations originally possible in Λ_H. Specifically, if $uv \in E(\Lambda_H)$, and $u(x) = v(x) = \emptyset$, then $u_x v_x \in \Lambda_T$ and if $|u(x)| \geq 1$ and $|v(x)| \geq 1$, then $u_j v_j \in \Lambda_T$ for all $1 \leq j \leq t$. Note that the reconfiguration uv moves at most $M(H)$ guards, and thus both of these reconfigurations also move at most $M(H)$ guards. Additionally, note that for any $c \in \Lambda_H$ with $|c(x)| \geq 1$

and $i \neq j$, c_i can reconfigure to c_j by moving the guard from x to l_j, and then guard on l_i back to x. This reconfiguration moves exactly two guards, and as $M(H) \geq 1$, this is at most $M(H) + 1$ guards.

We now consider possible attacks on T. Let $c_T \in \Lambda_T$ be the current configuration of guards, obtained by adding a guard to $c_H \in \Lambda_H$ and suppose some vertex p has been attacked. By construction, there is always a guard on x in each state of Λ_T, so we may assume that $p \neq x$. If $p \in V(H)$, then let $u_H \in \Lambda_H$ be the state which c_H reconfigures to to defend against an attack on p in Λ_H. We now determine the state $c_T \in \Lambda_T$ obtained from u_H which defends against this attack:

If $|c_H(x)| = |u_H(x)| = 0$, then u_T is obtained by adding a guard to x, and we observe that $c_T u_T \in E(\Lambda_T)$. As noted above, this reconfiguration moves at most $M(H)$ guards. Similarly, if $|c_H(x)| = |u_H(x)| = 1$, then c_T has a guard on some leaf $l_j \in L$. Let the configuration u_j be obtained by adding a guard to u_H on l_j. Then $c_T u_j \in E(\Lambda_T)$, and this reconfiguration moves at most $M(H)$ guards.

If $|c_H(x)| \geq 1$ and $|u_H(x)| = 0$, then a guard is moved off of x to defend against the attack. Thus there is some leaf $l_j \in L$ for which $|c_T(l_j)| \geq 1$. The configuration u_T obtained from u_H adds a guard to x, and so we can reconfigure from c_T to u_T by moving the guard from l_j to x and then otherwise copying the moves in $c_H u_H$. This moves at most $M(H) + 1$ guards as required.

Finally, if $|c_H(x)| = 0$ and $|u_H(x)| = 1$, then a guard is moved on to x in order to defend against the attack. Let $l_j \in L$ be some leaf, and u_j the corresponding state which places a guard on l_j. Then c_T can reconfigure to u_j by moving the guard on x to l_j and copying all other moves in $c_H u_H$. This moves at most $M(H) + 1$ guards as required.

Now suppose that $p = l_j \in L$ for some $1 \leq j \leq t$. If the guard $k + 1$ is already on a leaf, then as noted earlier we can reconfigure to the state c_j while moving at most 2 guards. Finally, if $k + 1$ is on x, then let u_H be the state which c_H reconfigures to if there is an attack on x. By definition this state has a guard on x, and so there exists a state $u_T \in \Lambda_T$ obtained from u_H by adding a guard to l_j. We can then reconfigure to u_T by copying the moves of $c_H u_H$ and then moving the guard $k + 1$ to l_i. This moves at most $M(H) + 1$ guards. This defends against all possible attacks, and so completes the proof. □

We can now proceed with the proof of the lower bound that $M(H) \leq M(T)$ when R1 is applied. We view this as an upper bound on the maneuver number of H and prove that we can restrict a strategy for T to one for H such that at most $M(T)$ guards are moved.

Lemma 2.10. *Let T be a tree. If we apply R1 and obtain H, then $M(H) \leq M(T)$.*

Proof. Suppose we apply R1 to x to remove the leaves $L = \{l_1, \ldots, l_t\}$ and obtain H. Let Λ_T be some maneuver optimal strategy for T. We construct a strategy Λ_H for H.

Note that by Theorem 2.4 we have that $D_M(H;\infty) = D_M(T;\infty) - 1$, and so each state in Λ_H must have one fewer guard than its corresponding state in Λ_T. For each state $c \in \Lambda_T$ we add the following states to Λ_H:

- If there is at least one guard placed on L, we add the corresponding configuration c' where we have removed one of the guards on L and moved all others to x (if any remain).
- If there are no guards on L, add the corresponding configuration c' where a guard on x has been removed.

Note that we can always apply one of the above modifications, as if there are no guards on L, then there must be at least one guard on x in order to form a dominating set. Additionally, note that for each edge $uv \in E(\Lambda_T)$ we have that $u'v' \in E(\Lambda_H)$. Clearly these reconfigurations move at most $M(T)$ guards. We now claim that Λ_H is an eternal dominating strategy for H.

If not, then there is some configuration $c' \in \Lambda_H$ and vertex $p \in V(H)$ such that if p is attacked when the guards are in c', there is no configuration to which c' can reconfigure. First suppose that $p \in V(H) \setminus \{x\}$. Let $c \in \Lambda_T$ be the configuration from which c' was obtained. Since Λ_T is an eternal dominating strategy for T, there is some state $d \in \Lambda_T$ to which c can reconfigure to defend against an attack on p. Let d' be the corresponding configuration in Λ_H. As noted, we must have that $c'd' \in E(\Lambda_H)$, and since $p \neq x$, there must still be a guard on p. This is a contradiction.

Now suppose that $p = x$. We may assume that c' does not place a guard on x, and thus the corresponding configuration $c \in \Lambda_T$ does not place any guards on L. Let d be the configuration to which c reconfigures some vertex $l_i \in L$ is attacked. We claim that the corresponding configuration $d'\Lambda_H$ places a guard on x. If not, then there must only be a single guard on L, and none on x. But then d could not have had the dominating property, which is a contradiction. Thus there is a configuration to which c' can reconfigure in order to defend against this attack. The result follows. □

We now prove similar results for R2, again in two parts:

Lemma 2.11. *Let T be a tree. If we apply R2 and obtain H, then $M(T) \leq M(H)$.*

Proof. Suppose we apply R2 to remove x and l to obtain H. Let Λ_H be a maneuver optimal strategy of H. We will construct a strategy Λ_T for T and show that this strategy moves at most $M(H)$ guards. Note that since we have applied R2 we have that $D_M(T;\infty) = D_M(H;\infty)+1$ by Theorem 2.4. Thus, our new strategy will use one extra guard, which we call $k+1$. For each configuration $c \in \Lambda_H$ we add the configurations c_x and c_l; these configurations place the guard $k+1$ on x or l respectively and otherwise copy the placement of guards in c.

Observe that for any $cd \in E(\Lambda_H)$ we have that $c_x d_x \in E(\Lambda_H)$ and $c_l d_l \in E(\Lambda_H)$ trivially, and each such reconfiguration moves at most $M(H)$ guards. Additionally, we know that for any $c \in \Lambda_H$, $c_x c_l \in E(\Lambda_T)$; these reconfigurations move 1 guard.

We now show that Λ_T is an eternal dominating strategy. Suppose we are in the configuration $c_x \in \Lambda_T$; i.e. the guard $k+1$ is on x. If any vertex in $V(H)$ is attacked, then there is some $d \in \Lambda_H$ to which c reconfigures to defend against this attack. In Λ_T we can similarly reconfigure to f_x. If l is attacked, we can reconfigure to d_l. There are similarly valid reconfigurations for when the guard $k+1$ is placed on l. Thus we can defend against all attacks, and so Λ_T is an eternal dominating strategy which has $m(\Lambda_T) = M(H)$. □

Lemma 2.12. *Let T be a tree. If we apply R2 and obtain H, then $M(H) \leq M(T)$.*

Proof. First, note that since we are applying R2 that the two vertices being removed are a leaf and its degree 2 support vertex. Assume without loss of generality that x is the support vertex and l is the leaf, and let y be the other neighbour of x. Now let Λ_T be a maneuver optimal strategy for T. We will construct an eternal dominating strategy Λ_H for H by modifying the states of Λ_T. Note that by Theorem 2.4 we have that $D_M(H;\infty) = D_M(T;\infty) - 1$. For each $c \in \Lambda_T$, let c' be the configuration obtained by placing all but one of the guards which were on x or l on w. Add c' to Λ_H.

Note that we must always have at least one guard on x or l in order to maintain a dominating set, and so for any $c \in \Lambda_T$, c' is well-defined. Also note that for any $cd \in E(\Lambda_T)$ we have that $c'd' \in E(\Lambda_H)$. Observe that the $c'd'$ reconfiguration moves no more guards than the cd reconfiguration, and so $m(\Lambda_H) \leq M(T)$. We now claim that Λ_H is an eternal dominating strategy for H.

Suppose not. Then there is some configuration c' and vertex p such that if p is attacked while we are in configuration c', we are unable to defend. However note that for each configuration in Λ_H there are no vertices in H which receive fewer guards than their counterparts in Λ_T. Thus if such a pair c' and p exist, this would contradict Λ_T being an eternal dominating strategy. Thus Λ_H is an eternal dominating strategy for H which has $m(\Lambda_H) \leq M(T)$. □

We can combine the above four results into one:

Corollary 2.13. *Let T be a tree, then both of the following hold:*

- *If we apply R1 and obtain H, then $M(H) \leq M(T) \leq M(H) + 1$.*
- *If we apply R2 and obtain H, then $M(T) = M(H)$.*

Proof. This follows directly from Lemmas 2.9, 2.10, 2.11, 2.12. □

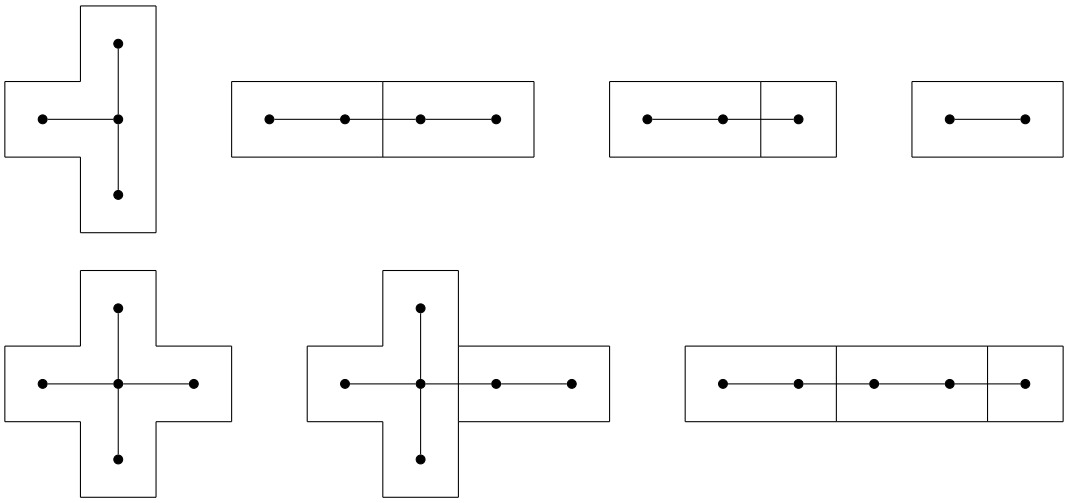


Figure 2.3: The neocolonizations which result from the maneuver numbers algorithm to all trees with at most 5 vertices. As can bee seen, all are locally finest.

2.4.2 The Maneuver Number Algorithm

We now provide a polynomial time algorithm for computing $M(T)$ when T is a tree. This algorithm is very similar to the one used to prove Theorem 2.3. That algorithm repeatedly applies either R1 or R2 until the tree has been reduced to a base case. With the simple modification that we always apply R2 when possible, and only apply R1 when necessary, we can obtain neocolonizations which are locally finest. In this section we will show that the strategies derived from such neocolonizations are maneuver optimal, which implies that we can compute $M(T)$ in polynomial time.

Theorem 2.14. *Algorithm 1 finds a locally finest neocolonization of minimum weight in linear time.*

Proof. First note that Algorithm 1 terminates, as either R1 or R2 can always be applied to a tree with diameter of at least 3. In addition, since at least 2 vertices are removed in each step, at most $\frac{n}{2}$ steps are taken. Initially determining where R1 and R2 can be applied can be done in linear time. After each reduction is applied, we determine whether this has created a new location to which we can apply a reduction in constant time. Thus, the algorithm as a whole runs in linear time.

We will prove this by showing that at each step of the algorithm, the neocolonization produced is locally finest and has minimum weight. The result is true for all trees on 5 or fewer vertices, see Figure 2.3; it is also trivially true for stars. We proceed by induction on the number of vertices in our tree T. Suppose the result is true for all trees with at most k vertices, and let T be some tree on $k+1$ vertices. If we can apply R2 to T to obtain T' then by induction there exists a locally finest neocolonization

Algorithm 1 An algorithm for finding locally finest neocolonizations of T when T is a tree.

procedure MANEUVER-NUMBER(T)
  if $|V(T)| \leq 2$ or T is isomorphic to a star **then**
    Let N be a neocolonization whose only block is $V(T)$.
    return N
  end if
  if R2 can be applied to T **then**
    Let x and l be such that R2 can be applied at x to remove x and l.
    Let $H = T - \{x, l\}$ be the graph after applying R2 to x.
    Let N_H be the neocolonization resulting from calling `Maneuver-Number(H)`.
    Let $N_T = N_H \cup \{\{x, l\}\}$.
    return N_T.
  else if R1 can be applied to T **then**
    Let x and $L\{l_1, \ldots, l_t\}$ be such that R1 can be applied at x to remove L.
    Let $H = T - L$ be the graph after applying R1 to x.
    Let N_H be the neocolonization resulting from calling `Maneuver-Number(H)`.
    Let B_x be the block of N_H containing x.
    Let $N_T = N_H \setminus B_x \cup (B_x \cup L)$.
    return N_T.
  end if
end procedure

N' of T' which has minimum weight. Let N be the neocolonization of T obtained by adding the two vertices deleted by R2 as a block. Then $w(N) = w(N') + 1$, which is best possible by Theorem 2.4. Additionally this new block cannot be refined, and thus if N was not locally finest we could refine N'. However this contradicts that N' was locally finest.

So suppose that we can only apply R1 to T to delete the leaves $L = \{l_1, \ldots, l_t\}$ adjacent to some vertex x to obtain T'. Then again by induction there is some locally finest neocolonization N' of T' of minimum weight. Let V_j be the block containing x in N' and form a neocolonization N for T by adding each vertex in L to V_j. Note that $w(N) = w(N') + 1$ as x must have been a leaf in V_j and is now an internal vertex. This is best possible by Theorem 2.4. We now contend that N is locally finest. Suppose not, and let V_i be some block which can be refined. If $V_i \neq V_j$, then we have a contradiction as we could have refined N'. If $V_i = V_j$, then let $B_1, \ldots, B_k$ be the refinement of V_j, and let B_l be the block containing x. Note that B_l must contain each of the deleted leaves, as otherwise our refinement would not be an optimal neocolonization.

However notice then that if we replaced V_j with $\{B_1, \ldots B_{l-1}, B_l \setminus L, B_{l+1}, \ldots, B_k\}$ this is a refinement of N'. This contradicts N' being locally finest. □

We now focus on showing that the strategy derived from any locally finest neocol-

onization is maneuver optimal. We first prove two helpful lemmas regarding locally finest neocolonizations and the structure of their underlying trees. Let T be a tree with leaves L, and consider $T' = T - L$. Note that the leaves of T' are exactly the vertices at which we can apply R1 or R2. We call a vertex x of T **moderate**[1] if $diam(T' - x) = diam(T')$, or in other words, if it is not diameter-critical in T'.

Lemma 2.15. *Let T be a tree and $N = \{V_1, \ldots, V_k\}$ an optimal locally finest neocolonization of T. Then for each non-clique block V_i, we have that every internal vertex of $T[V_i]$ has degree at least 3.*

Proof. Suppose for contradiction we have some block U such that $H = T[U]$ is not a clique and has an internal vertex x with $deg(x) = 2$. Let w and y be the neighbours of x. Let B_w and B'_y be the vertices in the components of $H - x$ which contain w and y respectively. Now let $B_y = B'_y \cup \{x\}$.

Now note that $B_w \cup B_y = U$ and $B_w \cap B_y = \emptyset$, and so they are a refinement of U. Given a block F let $i(F)$ denote the number of internal vertices in $T[F]$. Note that we have that $i(B_w) + i(B_y) \leq i(U) - 1$ as each internal vertex of $B_w \cup B_y$ is an internal vertex of U, but x is now a leaf. Thus, the weight of this refinement is at most $i(B_w) + 1 + i(B_y) + 1 \leq i(U) + 2 - 1 = i(U) + 1$, which is exactly the weight of U. Thus this is a refinement, which contradicts that N was locally finest. □

Lemma 2.16. *Let T be a tree whose only locally finest neocolonization contains a single block, and suppose T has no moderate vertices. Let L be the set of leaves of T. Then $T' = T - L$ is a path.*

Proof. Note that if T is a clique, then $T \cong K_2$ and T' is the empty graph, which can be viewed as the trivial path on 0 vertices. So suppose that T is not a clique. Then by Lemma 2.15, each internal vertex of T has degree at least 3. Observe that we can apply R1 at any leaf of T'. Since T contains no moderate vertices, it must be that the diameter decreases no matter where we apply R1. Suppose for contradiction that T does not have any moderate vertices but that T' is not a path. Since T' is not a path, it must have at least 3 leaves. Now let P' be a diametric path in T' and let P be an extension of P' in T obtained by appending leaves adjacent to each endpoint. Observe that P is a diametric path in T.

Now let Q' be some path between two leaves in T' which is distinct from P'. Let q be an endpoint of Q' which is not in P'. Let H be the graph obtained by applying R1 to q. Since q did not lie on P', the leaves adjacent to it certainly do not lie on P. Therefore P still exists, and $diam(H) = diam(T)$ contradicting our assumption and proving the result □

[1]Because it is not extreme.

We are now ready to show that optimal locally finest neocolonizations give maneuver optimal strategies. We will proceed by induction on the number of blocks in a locally finest neocolonization N, but first show that the result is true when N contains only a single block.

Lemma 2.17. *Let T be a tree and N_T a locally finest neocolonization of T containing only a single block B. If Λ_T is the strategy derived from N_T, then $M(T) = m(\Lambda_T)$.*

Proof. First note that because N_T contains only a single block, B, we have $T[B] = T$. If T is a clique, the result is clearly true. Therefore we may assume that T is not a clique. We proceed by induction. Suppose the result is true for all trees T on at most $n-1$ vertices, and let T be some tree with n vertices. We have two further cases:

First, suppose that there exists a moderate vertex x. Let H be the graph obtained by applying R1 to x. Because N_T is locally finest, we know that any locally finest neocolonization N_H of H must contain only a single block. If Λ_H is the strategy derived from N_H then by induction

$$M(H) = m(\Lambda_H) = diam(H) = diam(T) = m(\Lambda_T)$$

Since we have applied R1 to obtain H, by Lemma 2.10 we have that

$$M(H) = diam(T) \leq M(T)$$

Note that since Λ_T is an eternal dominating strategy of T, by definition we have that $M(T) \leq m(\Lambda_T) = diam(T)$. This gives us $M(T) = diam(T) = m(\Lambda_T)$ as desired.

So now suppose that T does not contain any moderate vertices and let L be the set of leaves of T. By Lemma 2.16 we know $T' = T - L$ is a path. Suppose for contradiction that there is some strategy Λ which eternally dominates T but has $m(\Lambda) < diam(T) = m(\Lambda_T)$. Let l_x and l_y be the endpoints of a diametric path in T. We claim that there exists a sequence of attacks which results in a configuration c which does not place a guard on every internal vertex of T.

Let c_y be the configuration of guards achieved after defending against an attack on l_y. If c_y does not place a guard on each internal vertex of T, then we are done. If it does, then we must have a guard on l_y and each internal vertex of T. If we then attack l_x to obtain c_x, we must end up with a guard on l_x. If c_x still places a guard on each internal vertex of T, let μ be the moves list which achieves this reconfiguration. Then the movement digraph of T, $F = MD(T, \mu)$ defined by μ must have a directed path from l_y to l_x, which contradicts that $m(\Lambda) < d(l_x, l_y) = diam(T)$. Therefore, c_x does not place a guard on each internal vertex.

We will count the number of guards whose location we know and use this to show that Λ cannot exist. Define each of the following for c_x:

- Let U be the set of internal vertices of T which do not have a guard on them.

- Let D be the set of vertices with guards on them.
- Let I be the set of internal vertices of T.
- Let x and y be the vertices adjacent to l_x and l_y respectively.

By Lemma 2.15 each vertex in U is adjacent to at least one leaf, and by Lemma 1.3 we must have that there is a guard on each such leaf. Thus there are at least $|U|$ guards on the leaves of T. By assumption we also know that there are guards on every vertex of $I \setminus U$, accounting for another $|I| - |U|$ guards. Finally, let $X = \{x, y\} \cap U$. Since x and y are leaves in T', each is adjacent to at least 2 leaves. Thus if $|X| \geq 1$, then we have undercounted the number of guards on the leaves of T. Specifically, we know the location of at least $|I| + |X|$ guards:

$$(|I| - |U|) + (|U| - |X|) + 2|X| = |I| + |X|$$

Note that $D_M(T; \infty) = |I| + 1$, so $|X| \leq 1$. We now claim that there is some component C of $T[D]$ containing x or y which contains no leaves of T and has no guards occupying the same vertex. We condition on $|X|$:

- If $|X| = 1$ (suppose $x \in U$ without loss of generality) then we know the location of every guard. Thus the component of $T[D]$ containing y cannot contain a leaf of T nor any vertices with multiple guards.
- If $|X| = 0$, then we know the location of at least $|I|$ guards and there is a single guard unaccounted for, call it g. Note that the components C_x and C_y containing x and y respectively must be distinct in $T[D \setminus \{g\}]$ and neither can contain a leaf of T. We know g must be placed on a leaf, but no matter where it is placed one of C_x or C_y will neither contain a leaf nor have any guards occupying the same vertex.

In either case there is a component of $T[D]$ which contains no leaves of T and contains one of x or y. Suppose without loss of generality that $C \subseteq D$ is the set of vertices which induce this component and that $y \in C$. We prove by induction on $|C|$ that there is a sequence of attacks which causes Alice to lose.

If $|C| = 1$, then $C = \{y\}$. Let w be the internal vertex adjacent to y and note that $w \in U$. If a leaf adjacent to y is attacked, the guard on y must move there. Because there is no guard on w, it is impossible to move a guard to y and we do not have a dominating set.

So now suppose that the result is true for components containing k vertices and let C be a component on $k + 1$ vertices. Let $F = T[N[C]]$ be the subgraph induced by the closed neighbourhood of C and I_F the set of internal vertices of F. Note that we have exactly $|I_F|$ guards assigned to the vertices of F, and since C forms a connected

component these guards must be stationed exactly on I_F. Now let $z \in C$ be the internal vertex at maximum distance from y, and let l some leaf adjacent to z. Consider an attack on l and let c' be the configuration of guards achieved by Λ when defending against this attack.

The guard on z must have moved to l in order to defend against this attack, and no matter how the guards have moved we must have $|c'(y)| \geq 1$, as otherwise there would be a leaf adjacent to y which is not adjacent to a vertex with a guard. We now consider the locations of the $|I_F|$ guards which were originally stationed on C. Let w be the internal vertex adjacent to z which did not have a guard on it before l was attacked. It may be the case that guards have moved from outside F to w, but this does not matter for the remainder of our argument.

Let U_F be the set of internal vertices of F which do not have a guard on them. Because we began with exactly $|I_F|$ guards on C and have moved a guard from z to l, it must be the case that $|U_F| \geq 1$. By similar logic to before, for each $u \in U_F$ there must be at least one guard on a leaf adjacent to u. By definition there is a guard on each vertex in $I_F \setminus U_F$, and so we know the location of all $|I_F|$ guards:

$$|I_F \setminus U_F| + |U_F| = |I_F| - |U_F| + |U_F| = |I_F|$$

Note that if $|c'(v)| > 1$ for any $v \in C \setminus \{w\}$ this would contradict that we began with $|I_F|$ guards. Let D' be the set of vertices in F with guards on them. Observe that the component C' of $T[D']$ which contains y cannot contain any leaves or vertices which have multiple guards occupying them. Since $|C'| < |C|$ there is a sequence of attacks which causes Alice to lose.

This completes the argument that there cannot exist an eternal dominating strategy Λ with $m(\Lambda) < diam(T)$. Thus $diam(T) \leq M(T)$. We know that the strategy Λ_T derived from N_T has $M(T) \leq m(\Lambda_T) = diam(T)$, and so $M(T) = m(\Lambda_T)$ as desired. □

Theorem 2.18. *Let T be a tree and N a locally finest neocolonization of T. Let Λ_T be the strategy derived from N. Then $m(\Lambda_T) = M(T)$.*

Proof. We proceed by induction on the number of blocks in N. Note that the result is true when N contains only one block by Lemma 2.17. So suppose that N contains more than one block and consider $BA(T, N)$, noting that it is a tree by Lemma 2.5. Consider the set $L = \{b : b \text{ is a leaf of } BA(T, N)\}$ and let U be some block in L which achieves $\min_{b \in L} diam(G[b])$. Note that $H = T \setminus U$ is a tree with neocolonization $N' = N \setminus \{U\}$. We claim that N' is locally finest; if this were not true, then we would be able to refine N.

Let Λ_H be the strategy derived from N'. As N' has fewer blocks than N and is locally finest, we have that $m(\Lambda_H) = M(H)$. Now consider the strategy Λ_T derived from N. Since each block in N' is a block in N we have that $m(\Lambda_H) \leq m(\Lambda_T)$. We

now claim that $m(\Lambda_T) \leq m(\Lambda_H)$. Suppose not. Then we must have that $diam(U) > diam(F)$ for each $F \in N'$. This further implies that $diam(U) > diam(b)$ for each $b \in L$. Since $BA(T, N)$ is a tree, we know that L contains at least two blocks of N, and so this contradicts our choice U.

Thus, we have that $m(\Lambda_T) = m(\Lambda_H) = M(H)$. Note that since Λ_T is an eternal dominating strategy for T, we have that $M(T) \leq m(\Lambda_T) = M(H)$. Now note that $M(H) \leq M(T)$ since we could restrict any maneuver optimal strategy for T to a strategy for H. Thus we have that $m(\Lambda_T) = M(H) = M(T)$ as desired. □

We can finally combine the above results to obtain our main result for this chapter:

Theorem 2.19. *For any tree T, $M(T)$ can be computed in linear time.*

Proof. By Theorem 2.14, Algorithm 1 finds a locally finest neocolonization N in linear time. By Theorem 2.18 and Lemma 2.1, $M(T) = \max_{B_i \in N} diam(G[B_i])$. Computing the diameter of each block can be done via breadth-first search and requires linear time in total. □

Chapter 3
Invasion Numbers

3.1 Introduction

This chapter is dedicated to studying a variant of eternal domination where not only is there an infinite sequence of attacks to defend, groups of these attacks occur simultaneously and must all be defended in one round. Suppose there are most a simultaneous attacks. It is clear that one needs at least as many guards as is needed to defend against a single attack, and unsurprisingly many graphs need more guards. However there exist graphs for which you do not need more guards, which leads to the concept of the *invasion number* of a graph, the maximum number of simultaneous attacks which can be defended without additional guards.

In this chapter we construct some classes of graphs which have invasion number 2, and prove that certain structures cannot appear in graphs with high invasion number. We then determine an upper bound on the number of guards needed to eternally defend any graph against 2 attacks.

3.2 Definitions

Let $D_A(G; a)$ denote the eternal domination number for the graph G where sets of at most a vertices are attacked simultaneously. Note that we take each of these to only include distinct vertices, and each attack is defended against is still defended against by moving a guard to each attacked vertex. The **invasion number** of a graph is then the maximum value of a such that $D_A(G; a) = D_A(G; 1)$, denoted $I(G)$. In other words, the invasion number is as the maximum number of simultaneous attacks against which we can defend using the same number of guards as we need to defend against a single attack.

Example 3.1. Consider P_3; we will determine the 1 and 2 attack eternal domination number, as well as the invasion number. Note that although there is a dominating set on P_3 which uses a single guard, there is only one. As such, it is not possible

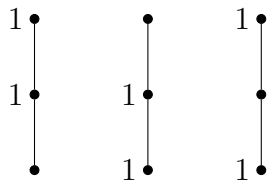


Figure 3.1: An eternal dominating strategy for P_3.

defend against attacks on the leaves when starting from such a configuration. Thus, $D_A(P_3; 1) \geq 2$, and Figure 3.1 shows a strategy using 2 guards. Thus $D_A(P_3; 1) = 2$.

Now consider a scenario where 2 vertices are attacked at a time. Note that the strategy in Figure 3.1 covers all possible pairs of attacks on P_3. Thus, $D_A(P_3; 2) \leq 2$, and as $2 = D_A(P_3; 1) \leq D_A(P_3; 2)$, we have that $D_A(P_3; 2) = 2$. This further implies that $I(P_3) \geq 2$. It is not hard to see that more guards are needed when 3 vertices are attacked, and as such $I(P_3) = 2$.

3.3 First Results

We begin by presenting a few simple observations regarding the multi-attack eternal domination number:

Proposition 3.2. *Let G be a graph with n vertices. Then for all $1 \leq k \leq n$:*

$$D_A(G; 1) \leq D_A(G; k) \leq D_A(G; n)$$

Proposition 3.3. *Let G be a graph. Then $D(G; a) \geq a$ for all a.*

Lemma 3.4. *Let G be a graph, and x a vertex adjacent to l leaves. Then in any a attack eternal dominating configuration for G there must always be at least $\min\{l, a\}$ guards on x or its leaves.*

Proof. Suppose for contradiction that there is some a attack eternal dominating configuration c of G which does not place at least $k = \min\{l, a\}$ guards on x or its leaves. Consider attacking k leaves adjacent to x. It is impossible to defend against this attack, giving us our contradiction and result. □

We finish with two simple upper bounds for particular restricted classes of graphs.

Lemma 3.5. *For any a and connected graph G on $n \leq 2a - 1$ vertices we have that*

$$D_A(G; a) \leq \left\lceil \frac{a}{a+1} n \right\rceil$$

Proof. Note that if $n < a+1$, then $\left\lceil \frac{a}{a+1} n \right\rceil = n$ and we are trivially done by placing a guard on each vertex. If $a+1 \leq n \leq 2a-1$, then $\left\lceil \frac{a}{a+1} n \right\rceil = n-1$. Consider placing $n-1$ guards arbitrarily on the vertices. Consider an attack on a set of a vertices; we may assume that the lone undefended vertex, x is attacked. There must be some vertex with a guard on it which was not attacked as $a \leq n-1$, and so there exists a path from x to this vertex. By Lemma 1.10 we can reconfigure to the configuration where x has a guard on it, as well as every other attacked vertex. The result follows. □

Lemma 3.6. *For any a and $t \geq 2a$ we have that $D_A(K_{1,t}; a) \leq 2a$.*

Proof. Let x be the universal vertex of $K_{1,t}$. Place a guards on x and a guards on arbitrary leaves. When defending against an attack, move guards from the universal vertex to the attacked vertices, and any guards on leaves which were not attacked back to x. This can be done for any attack, and the result follows. □

3.4 Classes with Invasion Number 2

In this section we show that a number of classes of graphs have invasion number 2. This is surely not a complete catalogue of such graphs, but these constructions show that these graphs exist. One construction in particular shows us that it is impossible to provide a forbidden subgraph characterization of these graphs.

Constructing these classes requires balancing two competing factors: if the graph is too spare, then it may be possible to force some section of the graph to have too few guard to defend against some pair of attacks, and if the graph is too dense then it may be possible to force a relatively small number of guards into an unfavourable configuration. These factors mean that our constructions tend towards one of two extremes:

- Either $D_A(G; 1)$ is linear in n (and is often close to $\left\lceil \frac{n}{2} \right\rceil$),
- Or $D_A(G; 1)$ is a small constant.

3.4.1 Modified Cliques

The first class of graphs we construct are those which begin as modifications of cliques. We construct two such families: those where the complement of the graph is a non-empty matching, and those where the vertices of a clique can be partitioned such that each set has a unique private neighbour. The first is derived from a more general statement.

Theorem 3.7. *Fix j and let G be a graph whose complement is the disjoint union of i copies of K_j and l copies K_1 where $i \geq 1$ and $i+l \geq 2$. Then $D_A(G; j-1) = D_A(G; j)$.*

Proof. Let G and j be as in the theorem statement. First, note that in G any set of j vertices forms a dominating set. If not there would be some set D of j vertices such that there is some vertex $v \notin D$ which is not adjacent to anything in D. This would then imply that v has degree j in $\overline{G}$, which is impossible.

We first show that $D_A(G; j-1) \geq j$. Suppose for contradiction that we can eternally dominate G with $k < j$ guards. Let J be some independent set of size j in G and attack k vertices from J arbitrarily. Since we only have k guards, we must end up with a guard on each attacked vertex. But then the remaining vertices in J are not adjacent to any guards, and so we do not have a dominating set.

We next show that $D_A(G; j) \leq j$ by showing an eternal dominating strategy Λ which uses j guards. Begin by placing j guards on the graph arbitrarily. Whenever a set of j vertices are attacked, move the guards to the attacked vertices. This is always a dominating set, so all we need to show is that each subset of j vertices can reconfigure to each other set of j vertices.

Let d_1 and d_2 be two configurations and let V_1 and V_2 be the sets of vertices which d_1 and d_2 place guards on respectively. Define $B = (V_1, V_2, E)$ as the bipartite graph with partite sets V_1 and V_2. For any $v_1 \in V_1$ and $v_2 \in V_2$ add the edge v_1v_2 if $v_1v_2 \in E(G)$ or $v_1 = v_2$. A perfect matching in this graph corresponds to a set of moves which allow d_1 to reconfigure to d_2.

Suppose for contradiction that a perfect matching does not exist. Let $S \subseteq V_1$ be the smallest set such that $|N(S)| < |S|$ guaranteed to exist by Hall's Condition. Note that if there exists $v \in S$ such that v is an isolated vertex in $\overline{G}$, then $|N(S)| = j \geq |S|$, which is a contradiction. Similarly, if there exist $u \in S$ and $v \in S$ such that u and v are in different copies of K_j in $\overline{G}$, then $|N(S)| = j$ as each vertex in $N(S)$ is adjacent to at least one of u or v.

Thus, there is some $C \cong K_j$ in $\overline{G}$ such that $S \subseteq V(C)$. If V_2 contains at least $|S|$ vertices from $V(G) \setminus C$ then again we have a contradiction. Therefore V_2 contains at most $|S| - 1$ vertices from $V(G) \setminus C$, and so contains at least $j - |S| + 1$ vertices from C. Thus, there is some vertex $v \in C$ such that $v \in V_2$. This vertex is only adjacent to itself and the vertices of V_2 in $V(G) \setminus C$, and so if we remove it we have a smaller counterexample, which is a contradiction. □

Corollary 3.8. *Let G be a graph whose complement is a non-empty matching. Then $I(G) = 2$.*

Proof. This follows directly from Theorem 3.7. □

An example of such a graph can be seen in Figure 3.2. Our next construction works by partitioning the vertices and adding a new vertex adjacent to only those in each set of the partition except one. This forces us to always have a guard stationed in each set, and allows one of these guards to move in order to defend pairs of attacks on other sets.

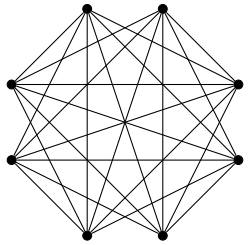


Figure 3.2: An example of a graph whose complement is a non-empty matching, and thus has invasion number 2.

Theorem 3.9. *Suppose $G \cong K_n$ for some n and let $A_0, \ldots, A_k$ be a partition of the vertices of G such that $|A_0| = 1$. Construct a new graph H by adding a vertex b_i adjacent to each vertex in A_i for each $1 \leq i \leq k$. Then $I(H) = 2$.*

Proof. Let H be constructed as above and suppose $A_0 = \{v\}$. We first show that $D_A(G; 1) \geq k+1$. Suppose not. Then there is some eternal dominating strategy which uses at most k guards. In order to form a dominating set there must be a guard in the neighbourhood of each b_i, and so their must be exactly one guard in each $A_i \cup \{b_i\}$ for $1 \leq i \leq k$. Consider an attack on v. In order to defend against this attack a guard must move to v, but no matter how the remaining guards move there must be some b_i which has no guard in its neighbourhood, which is not a dominating set.

Next we show that $D(G; 2) \leq k+1$ by demonstrating a 2 attack eternal dominating strategy which uses $k + 1$ guards. To begin, place one guard on each b_i and the final one on v. We refer to the guard beginning on v as the *spare guard*. We maintain as an invariant that the spare guard does not leave the neighbourhood of v. Now let t_1 and p_2 be some pair of vertices which are attacked:

- If $t_1 \in A_i$ and $p_2 \in A_j$ with $i \neq j$, then we can defend each with the guard in A_i and A_j.
- If $t_1 \in A_i$ and $t_2 = v$, then we can defend the attack on t_1 with the guard in A_i and move the spare guard back to v.
- Finally, if $t_1 \in A_i$ and $t_2 \in A_i$, then at least one of t_1 or t_2 must be in the neighbourhood of v (say t_2 without loss of generality). We move the guard in A_i to t_1 and move the spare guard to t_2.

This covers all possible cases, so $D(G; 2) \leq k + 1$. By the first claim and Theorem 3.2 we have that $D_A(G; 1) = D_A(G; 2) = k + 1$, and thus $I(G) = 2$. □

See Figure 3.3 for an example of this construction.

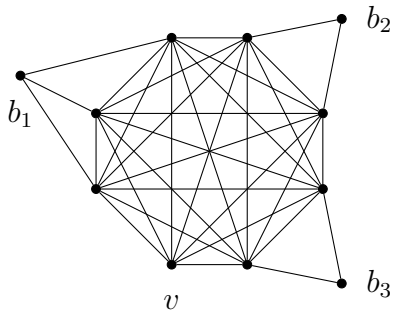


Figure 3.3: An example of the construction detailed in Theorem 3.9.

3.4.2 The Distinguished Corona

In this section we focus on one particular graph operation which we call the *distinguished corona* of a graph. When this operation is applied to a graph G it may produce a graph H which has invasion number 2. The goal of this section is to characterize when this occurs. Formally, let G be a graph with some vertex x. Then the **distinguished corona of G with respect to x** is obtained by adding a leaf adjacent to every vertex in G other than x. We denote this with $H = DC(G, x)$.

We begin by determining $D_A(H; 1)$ for any graph H obtained by applying the distinguished corona construction to some graph.

Lemma 3.10. *Let G be a graph with n vertices and let $H = DC(G, x)$ for some $x \in V(G)$. Then $D_A(H; 1) = n$.*

Proof. For each vertex of H in $V(G)$ other than x, we will refer to it as the parent and the leaf attached to it as its child. Since each leaf must either have a guard on it or adjacent to it, every parent-child pair must have a guard on it at all times. Observe that there are $n - 1$ of these pairs in H.

We first show that $D(G; 1) \geq n$. Suppose for contradiction that we have some eternal dominating set which uses at most $n - 1$ guards. Thus there must be exactly one guard in each such pair, and thus no guard on x. Suppose x is attacked. We must end up with a guard on x, and no matter how we reconfigure there are now at most $n - 2$ guards on parent-child pairs. Thus at least one pair has no guard on it, and so we do not have a dominating set.

Next we show that $D(G; 1) \leq n$ by providing a clique covering C. Add each parent-child pair as a clique in C, as well as the set $\{x\}$ (which is trivially a clique). C is clearly a clique covering and contains n cliques, so by Theorem 1.7, $D_A(G; 1) \leq n$. □

We now begin the proof of a partial dichotomy theorem for when the distinguished corona gives a graph with invasion number 2.

Lemma 3.11. *Suppose G is a 3-connected graph with n vertices and let $H = DC(G, x)$ where $x \in V(G)$. Then $I(H) = 2$.*

Proof. We know that $D_A(H; 1) = n$ by Lemma 3.10. Thus all we need to show is that $D_A(H; 2) \leq n$; we do this by constructing a 2-attack strategy for H with n guards.

Begin by placing a guard on each vertex in $V(G)$. We call the guard which begins on x the *spare guard*. For each configuration d in our strategy, we will maintain three invariants. Let $D = \{v : |d(v)| \geq 1\}$ be the set of vertices with a guard on them. Our invariants are the following:

- $H[V(G) \cap D]$ is connected.
- If the spare guard is on some vertex $v \neq x$ then there are at least 2 internally vertex disjoint paths from v to $N(x)$ in $H[V(G) \cap D]$.
- The spare guard is always located on some vertex of $V(G)$, and if it is located on $v \neq x$, then the corresponding child of v has a guard on it.

Let P be some path in H. We say that P has been **blocked** at the start of a round if there are not guards on each internal vertex of P. We call a path **disrupted** if some vertex has been attacked which is an external private neighbour of an internal vertex. In other words, P is blocked or disrupted if we cannot use it to move guards via Lemma 1.10. Now suppose some pair of vertices t_1 and t_2 have been attacked. There are a few cases to consider.

First suppose t_1 and t_2 are in different parent-child pairs, say C_1 and C_2 respectively. Begin by moving the guard assigned to each of those pairs to the attacked vertex. To maintain the invariants, we need to consider further cases: first, if the spare guard is on x, then we can maintain all invariants by moving the guard on each parent-child pair other than C_1 and C_2 back to the parent vertex in that pair. Thus, exactly two vertices in $V(G)$ will not have a guard on them. Since G is 3-connected we know that $H[V(G) \cap D]$ is still connected, and so our invariants are maintained.

Now suppose that the spare guard is on $v \neq x$ at the start of the round. Note that this implies that v is a parent vertex with some child vertex u which has a guard on it. Let P_1 and P_2 be the paths in $H[V(G) \cap D]$ guaranteed to exist by the second invariant, and let P_3 be some third path disjoint from P_1 and P_2 from v to x. Note that neither of P_1 or P_2 is blocked, but P_3 may be.

If at least one of the three paths is not blocked or disrupted by the attacks, we can move the guard from v to x by Lemma 1.10. We can then move all guards on children back to their parents (aside from those in C_1 or C_2). This leaves exactly two vertices in $V(G)$ without a guard, and so all invariants are maintained.

So suppose that all three of these paths are blocked or disrupted by attacks. Note P_1 and P_2 must be disrupted by the two attacks, and P_3 must be blocked. Move all

guards on children vertices back to their parents (except for those in C_1 or C_2). Note that this means that P_3 is no longer blocked. Observe that v is necessarily a parent vertex with some child u which has a guard on it. Without loss of generality, suppose t_1 disrupted P_1. We will be unable to move the spare guard back to x, and so to maintain our invariants we must ensure there are two paths from the location of the spare guard to x in $H[V(G) \cap D]$.

We begin by moving v along P_1 to the parent vertex p of C_1. This is the new location of the spare guard. The path from p to x along P_1 is in $H[V(G) \cap D]$, accounting for one of the paths we must guarantee. Now move the guard on u back to v. The path consisting of the segment of P_1 from p to v concatenated with P_3 is then another path from p to x, which is internally vertex disjoint from the first path constructed. Thus, our second invariant is maintained. To see that the first is as well, note that the only vertices of $V(G)$ which may lack a guard are x and the parent vertex in C_2 (if it was not attacked). Therefore since G is 3-connected, $H[V(G) \cap D]$ is still connected.

Next suppose that $t_1 = x$ and t_2 is in some parent-child pair. We can defend the attack on t_2 with the guard assigned to that pair. If the spare guard is currently on x then we are done, so we may assume that it is currently located on some $v \neq x$. Thus by our second invariant there were two paths P_1 and P_2 from v to x in $H[V(G) \cap D]$ at the start of the round. By definition neither of these were blocked, and at most one could have been disrupted by the attack on t_2. Thus we can move a guard from v to x by Lemma 1.10 and designate this guard as the spare guard. Finally, for each guard on a child vertex in some pair other than the one containing t_2, move it back to its parent. To see that the first invariant is maintained observe that at most one vertex in $V(G)$ does not have a guard on it, and so $H[V(G) \cap D]$ is still connected. Since the spare guard is on x, the other invariants are trivially maintained.

Finally suppose t_1 and t_2 are respectively the parent and child vertices of some pair. By our first invariant we know that $H[V(G) \cap D]$ is connected, and so there is a path from the location of the spare guard to t_1 which is not blocked or disrupted. We can move the guard in the attacked pair to t_2 and the spare guard to t_1 by Lemma 1.10. We can then move all guards on child vertices back to their respective parent vertices. Therefore the only vertex in $V(G)$ which does not have a guard is x, and so $H[V(G) \cap D]$ is connected, and each path from t_1 to x necessarily has guards along all internal vertices. Thus all invariants are maintained.

This completes the proof that $D_A(H; 2) \leq n$. Finally, to prove that $I(H) = 2$ we need to show that $D_A(H; 3) > n$. This is comparatively simple: suppose not and consider the triplet of attacks u, v, and x where u and v are parent and child respectively in some pair. In order to maintain a dominating set after this sequence of attacks, we must still have a guard on each parent-child pair. Thus we have at least $n - 2 + 2 + 1 = n + 1$ guards, which is a contradiction. $\square$

Lemma 3.12. *Let G be an n vertex graph with a cut-vertex r and consider $H = DC(G, x)$ for any x. Then $I(H) = 1$.*

Proof. By Lemma 3.10 we have $D_A(H; 1) = n$. We will show by contradiction that $D_A(H; 2) \geq n + 1$. Recall that in order to maintain a dominating set on H we must have at least one guard in each of the $n - 1$ parent-child pairs of H.

First, suppose $r = x$ and let G_1 and G_2 be different components of $G - r$ with vertices $u \in G_1$ and $v \in G_2$. If we attack v and its child, then no matter how the guards are moved we must end with one in each of the other $n - 2$ pairs and thus none on x. This implies that no matter how we reconfigure from this state we will not be able to move a guard in G_1 to G_2 in one step. Further, we know that there must be exactly as many guards in G_2 as there are parent-child pairs. Therefore if we attack u and its child, we must reconfigure to a state with some parent-child pair in G_2 which does not have a guard on it.

Now if $r \neq x$, let v be some vertex in $G - r$ which is not in the same component as x. Therefore any path from x to v must go through r. If we attack v and its child, we again must no longer have a guard on x. If we then attack x and the child of r, the guard on r must move to defend its child. However this means that no guards from the component containing v can move to the component containing x. Thus we will be unable to defend against the attack on x, which is a contradiction. Therefore $D_A(H; 2) \geq n + 1$ and so $I(H) = 1$. □

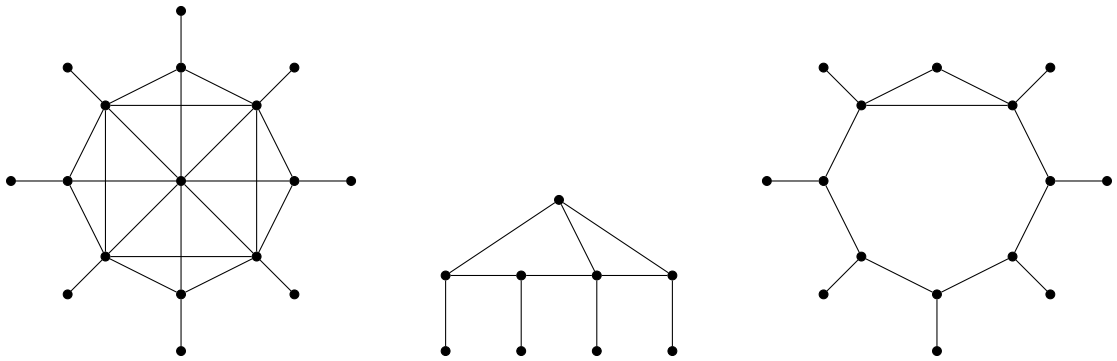
Lemmas 3.11 and 3.12 almost completely answer the question of when the distinguished corona gives a graph with invasion number 2, but the case when G is 2-connected remains open. We are able to prove a partial result in this case.

Lemma 3.13. *Let G be an n vertex 2-connected graph with a universal vertex x and consider $H = DC(G, x)$. Then $I(H) = 2$.*

Proof. By Lemma 3.10 we have $D(H; 1) = n$. We now show that $D(H; 2) \leq n$ by constructing a strategy. For our initial configuration, place a guard on each vertex of $V(G)$. Let d be some configuration in our strategy and let D be the set of vertices which have a guard placed on them. We will maintain as an invariant that if $x \notin D$, then every other vertex in $V(G)$ does have a guard on it. Note that this implies that if the spare guard is not on x, then it is on some vertex in $V(G)$. Now let t_1 and t_2 be a pair of attacked vertices.

If both t_1 and t_2 lie in different parent-child pairs, then we defend those attacks with the respective guard in those pairs, and move the spare guard back to x. Additionally, if $t_1 = x$ and t_2 is in some pair, then we move the spare guard to x and the guard in attacked pair to t_2.

Finally, if t_1 and t_2 are the parent and child of some pair respectively, then we first move the guard in the attacked pair to t_2. Let v be the current vertex with spare guard.



G_1 G_2 G_3

Figure 3.4: Three graphs obtained by taking the distinguished corona of some underlying graph, each of which was 2-connected. Additionally, $I(G_1) = 2$, $I(G_2) = 2$, and $I(G_3) = 1$.

If $v = x$, then we can trivially move the spare guard to t_1. Otherwise by our second invariant each vertex in $V(G)$ other than x must have a guard on it, and so there is a path from v to t_1 in $H[V(G) \cap D]$. This implies that we can use Lemma 1.10 to move a guard to from v to t_1, which we then designate as the spare guard. Finally, we move the guard on the leaf adjacent to v back to v in order to maintain our invariants.

This completes the proof that $D(H; 2) \leq n$. To see that $D(H; 3) > n$ (and thus that $I(H) = 2$) suppose for contradiction that there is some 3 attack eternal dominating strategy for G which uses n guards. Let u and v be a parent-child pair and consider an attack on x, u, and v. After responding to this attack, we must have at least one guard on each of the $n - 2$ parent-child pairs which were not attacked, and exactly 2 on the one which was. Additionally, there must be a guard on x. This implies that there are at least $n + 1$ guards, which is a contradiction. □

The above proof forms part of our partial dichotomy theorem for when this construction works with graphs which are 2-connected. In Figure 3.4 we show three graphs, G_1, G_2, and G_3 as well as the graphs resulting from applying the construction to these graphs. Each G is 2-connected, and has the following additional properties:

- G_1 has a universal vertex, and thus has invasion number 2 by Theorem 3.13.
- G_2 does not have a universal vertex, but we still have $I(G_2) = 2$.
- G_3 does not have a universal vertex, and we have that $I(G_3) = 1$.

However, just the result pertaining to universal vertices is sufficient to show that there is no forbidden subgraph characterization of graphs with invasion number 2.

Corollary 3.14. *There is no forbidden subgraph characterization for graphs with invasion number* 2.

Proof. Let F be some connected graph and construct F' by adding a universal vertex u. Note that F' is 2-connected as we have added a second path between any pair of vertices going through u. Thus by Theorem 3.13 the graph $H = DC(F', u)$ has invasion number 2. However, by construction F is an induced subgraph of H. □

3.4.3 Clique Expansion

The final class of graphs with invasion number 2 which we construct is again based on a particular operation, defined as follows: Let G be a graph with some maximal clique C. Then the **clique expansion** of G with respect to C, $CE(G, C)$, is the graph H obtained by adding two vertices v and l. Add edges between v and each vertex in C as well as l.

Let C be a maximal clique in a graph G. We call C **full** if for each $v \in C$, v has a neighbour outside of C. We call C **good** if it is not full, and either $|C| \geq 3$ or $|C| = 2$ and C contains a leaf of G. We now construct a class of graphs which have invasion number 2 using clique expansion.

Let $\mathcal{H}_0 = \{DC(K_n, x) : n \geq 1\}$ and note that every graph in $\mathcal{H}_0$ has invasion number 2, either by Lemma 3.13 when $n \geq 2$ or the fact that $I(P_3) = 2$ when $n = 1$. Recursively define

$$\mathcal{H}_i = \{CE(G, C) : G \in \mathcal{H}_{i-1} \text{ and } C \text{ is a good clique in } G\}$$

i.e. $\mathcal{H}_i$ is the set of graphs which can be obtained by applying a clique expansion to a good clique in a graph in $\mathcal{H}_{i-1}$. Finally, let $\mathcal{H} = \bigcup_{i=0}^{\infty} \mathcal{H}_i$. For a graph $G \in \mathcal{H}$, define the **generation** of G as the least i such that $G \in \mathcal{H}_i$. We denote this as $gen(G)$. See Figure 3.5 for examples of graphs in $\mathcal{H}$.

Theorem 3.15. *If $G \in \mathcal{H}$, then $I(G) = 2$.*

Proof. We proceed by induction on the generation of G. As noted above, the result is true for any graphs with $gen(G) = 0$. Thus we may assume that $gen(G) \geq 1$.

Let G be obtained by applying a clique expansion to some graph H with good clique C, and let v and l be the vertices added as described above. We first show that $D_A(G; 1) \geq D_A(H; 1) + 1$. Suppose not, and let Λ be some eternal dominating strategy for G which uses $D_A(H; 1)$ guards. In order to maintain a dominating set, there must always be a guard on v or l. Since $N(v) \setminus l$ induces a clique, we may assume that Λ does not move any guards from C to v. Thus $G' = G - \{v, l\}$ can be eternally dominated with $D_A(H; 1) - 1$ guards. Observe however that $G' \cong H$, and so we have a contradiction.

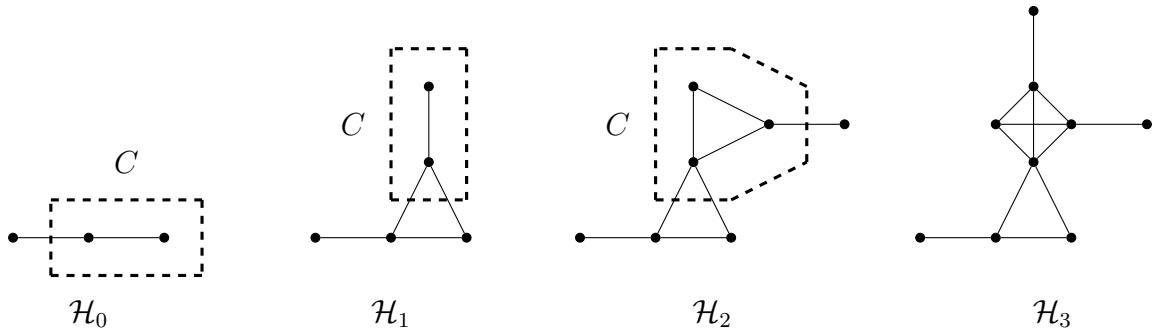


Figure 3.5: An example of graphs in $\mathcal{H}$, constructed by repeatedly applying a clique expansion to the previous graph. The clique which will be expanded is highlighted, and the generation of each graph is shown.

Let $D_A(H;1) = k$, we will show that there is a 2 attack eternal dominating strategy for G which uses $D_A(G;1) = k+1$ guards. Let $\mathcal{P}_G = (G, U, 2, \infty, \infty)$ be the instance of eternal domination we are considering, with $\mathcal{P}_H$ defined similarly. We will construct a 2 attack eternal domination strategy by considering two games of eternal domination, one played on $\mathcal{P}_G$ and the other on $\mathcal{P}_H$. We refer to these games as $\mathcal{G}_G$ and $\mathcal{G}_H$ respectively. By induction, there is some 2 attack eternal dominating strategy Λ_H for H which uses $D_A(H;1) = D_A(H;2)$ guards.

Let $v_1, \ldots, v_{k+1}$ and $u_1, \ldots, u_k$ denote the current location of the guards in $\mathcal{G}_G$ and $\mathcal{G}_H$ respectively. Additionally, let u be some arbitrary but fixed vertex in C with $N[u] = C$, guaranteed to exist as C is not full. We will maintain as invariants that if $v_i \notin \{v, l\}$ then $v_i = u_i$ and that if $v_i \neq u_i$, we have that $v_i = v$ and $u_i = u$. Our proof has two stages: first we show that for any configuration which satisfies this invariant we can defend any pair of attacks and reconfigure to another configuration which satisfies the invariant. We then show directly that one such configuration exists. This implies the existence of a 2 attacks strategy using $k+1$ guards.

Suppose we have played some amount of both games and ended up in the configurations c_G in $\mathcal{G}_G$ and c_H in $\mathcal{G}_H$. We assume that c_G satisfies the invariant. Now suppose the vertices p_1 and p_2 are attacked. We have a few cases.

If $p_1 \in V(H)$ and $p_2 \in V(H)$, then consider attacking both of these vertices in $\mathcal{G}_H$. As Λ_H is an eternal dominating strategy, there is a configuration c'_H to which we can reconfigure to defend against these attacks. Let M be the moves list which attains this configuration, and apply these exact moves to c_G. We can always do this by our second invariant. This clearly defends against the attacks and maintains the invariants.

If $p_1 \in V(H)$ and $p_2 \in \{v, l\}$, the consider attacking p_1 in $\mathcal{G}_H$ and let c'_H be the configuration to which Λ_H reconfigures. Let M be the moves list which attains c'_H and consider applying M to c_G. This clearly defends against the attack on p_1, and again is always possible by our second invariant. If we then move the guard on $\{v, l\}$ to p_2, we

will have defended against both attacks and maintained the invariant.

Finally, suppose that $p_1 = v$ and $p_2 = l$. Let x be some neighbour of v such that $N_H[x] = C$, guaranteed to exist as C was not full. Consider attacking x in $\mathcal{G}_H$. Then there is a moves list M which attains a configuration c'_H and defends against the attack on x. We have two cases, depending on whether or not a guard was moved to x to obtain c'_H. If no such guard was moved, then there must have already been one there, and in $\mathcal{G}_G$ we move this guard to v. If there was such a guard, let $m = (c, x)$ be the last move made which moves a guard to x. Replace m with $m' = (c, v)$ in M to get M' and apply M' to c_G. Finally, move the guard which began on $\{v, l\}$ to l in $\mathcal{G}_G$.

By our second invariant these reconfigurations are possible, and both clearly defend against the attacks. We now show that the invariants are maintained. By construction, the only guard which is not on the same vertex as it is in $\mathcal{G}_H$ is the guard on v, which is on u in $\mathcal{G}_H$ and so both invariants are maintained. Thus, we have found a winning strategy for $\mathcal{G}_G$[1] which uses $D_A(G; 1)$ guards, and so $I(G) \geq 2$. While it has yet to be proven, Theorem 3.16 shows the corresponding upper bound, and so $I(G) = 2$. □

3.5 Structural Characterizations

Despite showing in Corollary 3.14 that there is no forbidden subgraph characterization for graphs with invasion number 2, it is still possible characterize the structure of graphs with high invasion numbers. We prove three such results and then use them to bound the invasion number of trees.

Theorem 3.16. *Let G be a graph with adjacent vertices x and y such that $N(x) \subseteq N[y]$ and $G[N[y] \setminus \{x\}]$ is a clique. Then $I(G) \leq 2$.*

Proof. An diagram of this structure can be seen in Figure 3.6. Suppose for contradiction that there is G, x, and y as given but $a = I(G) \geq 3$. Consider any infinite sequence of a attacks on G which always attacks x and y. Then by assumption there is some eternal dominating strategy Λ_G which defends against this sequence of attacks using only $D_A(G; a) = D_A(G; 1)$ guards. Since x and y are always attacked there must always be guards on these vertices. Notice that since $N(x) \subseteq N[y]$ and $G[N[y]]$ is a clique, there is never any benefit to moving guards from x or y. This is because any guard moving from x to x' would need to be replaced. As $N(x)$ induces a clique, this second guard could have simply moved to x' originally. Thus, we may assume that no guard ever leaves x or y.

Let $H = G - \{x, y\}$. If we restrict Λ_G to H we obtain an eternal dominating strategy for H which defends against $a - 2 \geq 1$ attacks. As the guards on x and y never leave those vertices, this strategy uses exactly $D_A(G; a) - 2$ guards. Now consider $D_A(G; 1)$.

[1] Good game Bob.

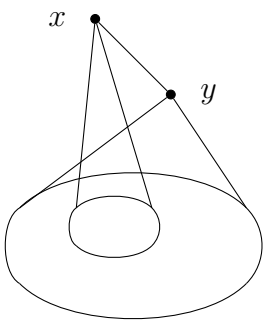


Figure 3.6: A diagram of the structure characterized in Theorem 3.16.

Since x and y are adjacent, we can defend them with a single guard. The rest of the graph is isomorphic to H, and so can be defended with $D_A(G;a) - 2$ guards. This gives us the following:

$$D_A(G;1) \leq D_A(K_2;1) + D_A(H;1) \leq 1 + D_A(G;a) - 2 = D_A(G;a) - 1$$

However this contradicts that $D_A(G;1) = D_A(G;a)$, and the result follows. □

Theorem 3.17. *Let G be a graph with some clique C. If there exist vertices x and y such that $N(x) \subseteq C$, $N(y) \subseteq C$ and $|N(x) \cap N(y)| \geq 1$, then $I(G) \leq 2$.*

Proof. A diagram of this structure can be seen in Figure 3.7. Suppose for contradiction that there exists G, C, x, y as described but with $a = I(G) \geq 3$. Then consider any infinite sequence of a attacks which always attacks x and y. Then there is some strategy Λ which defends G against this sequence of attacks. Note that since $N(x) \subseteq C$ and $N(y) \subseteq C$ and C is a clique, we can assume that this strategy never moves a guard off x or y.

Let $H = G - \{x, y\}$. If we restrict Λ to H, then we will have an eternal dominating strategy for H which uses $D_A(G;a) - 2 = D_A(G;1) - 2$ guards. Now let Λ_H be an eternal dominating strategy for H. We will construct an eternal dominating strategy Λ_G for G by modifying the states of Λ_H. For each state, we will add only a single new guard and thus show that $D_A(G;1) \leq D_A(H;1) + 1$. Let $v \in N(x) \cap N(y)$ be some arbitrary common neighbour of x and y. For each configuration $c \in \Lambda_H$ add the following states to Λ_G:

- If c does not place a guard on $N(x) \cap N(y)$, then let c_v be the configuration which places an additional guard on v. Add c_v to Λ_G.
- If c places a guard on $N(x) \cap N(y)$, then let c_x and c_y be the configurations where an additional guard has been placed on x and y respectively. Add both c_x and c_y to Λ_G.

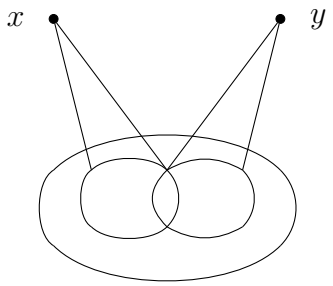


Figure 3.7: A diagram of the structure characterized in Theorem 3.17.

We now establish the reconfigurations possible in Λ_G.

- For any $cd \in E(\Lambda_H)$ each of $c_v d_v$, $c_x d_x$, and $c_y d_y$ are possible reconfigurations in Λ_G. This follows from the fact we do not need to move the newly added guard.
- For any $cd \in E(\Lambda_H)$ both of $c_v d_x$ and $c_v d_y$ are possible reconfigurations. To see this, note that by construction d has a guard on v and c does not. Thus the reconfiguration of c to d must move a guard to v. Since c_v already has a guard on v, we can move this guard to x or y respectively and get the desired reconfiguration.
- For any $c \in \Lambda_H$ which places a guard on C we have that $c_x c_y \in \Lambda_G$. To see this, note that there is some vertex $r \in C$ which has a guard on it in both states by construction. If we move the guard from r to y and the guard from x to r, then we achieve c_y.

Assume the guards form some configuration c_G obtained by adding a guard to some configuration c_H and suppose some vertex p has been attacked. If $p \in V(H)$, then there is some state d_H and corresponding d_G to which c_G can reconfigure to defend against this attack. So assume that $p \in \{x, y\}$. If c_G already places a guard on x or y, then we know we can reconfigure to an appropriate state. Otherwise, imagine that v has been attacked in H. Then c_H must have a neighbouring configuration d_H which places a guard on v. Then we can reconfigure from c_G to the state d_G which places a guard on x or y as appropriate. This gives us the following chain of inequalities:

$$D_A(G;1) \leq D_A(H;1) + 1 \leq D_A(G;a) - 2 + 1 = D_A(G;1) - 1$$

Where the last equality follows from our assumption that $I(G) = a$. This is clearly a contradiction, and so the result follows. □

The following result is similar to the above both in setup and proof, but allows us to determine the invasion number exactly.

Theorem 3.18. *Let G be a graph with some clique C. If there exist non-adjacent vertices x, y, and z such that $N(x) = C$, $N(y) = C$, and $N(z) = C$, then $I(G) = 1$.*

Proof. A diagram of the structure described can be seen in Figure 3.8. Suppose for contradiction that there are G, C, x, y, and z as in the theorem statement, but with $a = I(G) \geq 2$. Consider any infinite sequence of a attacks where the vertex x is always attacked. There is some strategy Λ which defends against this sequence of attack using at most $D_A(G; a)$ guards. Since $N(x) \subseteq C$, $N(y) \subseteq C$, and C is a clique, we may assume that the guard on x never leaves, and that no guard ever enters x (outside of perhaps the very first round).

Consider $H = G - \{x\}$. If we restrict to Λ to H then we have an eternal dominating strategy for H which uses $D_A(G; a) - 1 = D_A(G; 1) - 1$ guards. Let Λ_H be some 1-attack eternal dominating strategy for H. We demonstrate a 1 attack strategy Λ_G for G by having Alice consider two games of eternal domination $\mathcal{G}_G$ and $\mathcal{G}_H$, played on G and H respectively.

Suppose $D_A(H; 1) = k$ and let $v_1, \ldots, v_k$ and $u_1, \ldots, u_k$ be the locations of the guards in $[k]$ in $\mathcal{G}_G$ and $\mathcal{G}_H$ respectively. Additionally let w be some arbitrary but fixed vertex in C. We will maintain each of the following as invariants:

- If $v_i \notin \{x, y, z, w\}$ then $v_i = u_i$.
- There is always a guard on C.

For our initial configuration in $\mathcal{G}_H$ we choose a configuration in Λ_H which places a guard on w, and copy this configuration for $\mathcal{G}_G$. Now suppose we are in some configuration c_G in $\mathcal{G}_G$ and c_H in $\mathcal{G}_H$ and that Bob has just attacked the vertex t. We have three cases to consider:

First suppose that $t \in V(H)$ and $|c_G(t)| = |c_H(t)| = 0$. Since Λ_H is an eternal dominating strategy, there exists some c'_H to which c_H can reconfigure to defend against this attack. Apply the corresponding moves to c_G to obtain c'_G. Additionally, if there is no guard on C in c'_H, then we must have that there are guards on y and z. In this case, we additionally move a guard from y to w in $\mathcal{G}_G$.

To see that the invariants are maintained, observe that if c'_H placed a guard on C, then $v_i = u_i$ for all i and thus both invariants are clearly maintained. If c'_H did not place a guard on C, then let g be the guard which we moved from y to w. For all $i \neq g$ we still have that $v_i = u_i$, and thus the invariants are maintained.

Now suppose that $t \in V(H)$ but $|c_G(t)| = 0$ and $|c_H(t)| \geq 1$. By our first invariant and the fact that $x \notin V(H)$, we may assume that $t \in \{w, y, z\}$. If $t = w$, then by our second invariant there is some other vertex in C which has a guard on it; move this guard to w in both $\mathcal{G}_G$ and $\mathcal{G}_H$. Suppose $t \in \{y, z\}$. If there is a vertex $r \in \{x, y, z\}$ with $|c_G(r)| \geq 1$, then by our second invariant there is some $p \in C$ which has a guard

on it; move a guard from r to p and then the guard from p to t in $\mathcal{G}_G$. Thus, we may assume that $|c_G(r)| = 0$ for all $r \in \{x, y, z\}$. We now have two further cases.

If there are no guards on C in c_H, then there must be guards on both y and z in c_H. These guards must exist in c_G, and as our first invariant guarantees that c_G and c_H agree for all vertices other than w, x, y, and z, both guards must be on w. In this case, move one of these guards to t to defend against the attack in $\mathcal{G}_G$, and in $\mathcal{G}_H$ move the guards from y and z to w.

So now suppose that there are guards on C in c_H. Our second invariant and the property just established force the location of these guards to agree. If there are at least two guards, we can move one to t in both $\mathcal{G}_G$ and $\mathcal{G}_H$ to defend against the attack. So now suppose there is exactly one guard on C. We can move this guard to t to defend against the attack. The guard on t in $\mathcal{G}_H$ must still exist in $\mathcal{G}_G$, and thus it must be in $\{w, x, y, z\} \setminus \{t\}$. We can move this guard to w to maintain our invariants.

Finally suppose that $t = x$, as this is the only vertex in G which is not in H. First consider the case where at least one of y or z has a guard on it in $\mathcal{G}_G$. Without loss of generality, suppose that y has a guard on it. By our second invariant there is a vertex $p \in C$ with a guard on it. We can move the guard from y to p, and the guard from p to x to defend against the attack. Both invariants are clearly maintained.

So now consider the case where there are no guards on y or z in $\mathcal{G}_G$. Consider y and z in $\mathcal{G}_H$. First suppose that at least one of y or z *does not* have a guard on it in $\mathcal{G}_H$, and take this vertex to be y without loss of generality. Consider an attack on y in $\mathcal{G}_H$. By Λ_H there is a configuration c'_H which c_H can reconfigure to which places a guard on y. By Lemma 1.13 there is a list of moves M which obtains c'_H when applied to c_H. Let $m = (p, y)$ be the last move in M which moves a guard to y, and let M' be the sequence of moves where we have replaced m with (p, x). Now consider applying M' to c_G to obtain c'_G in $\mathcal{G}_G$. This gives a configuration which places a guard on x, and agrees with c_H everywhere outside of $\{x, y, z\}$ (so the first invariant is maintained). If c'_G does not place a guard on C then c_H does not either, and so there must be a guard on z in c_H. If the guard on z moved there when M was applied, then we simply remove that move from M' before applying it. If it was there already, then we add the move (z, w) to M'. In either case, both invariants are maintained.

We now consider the final case of this proof, where $t = x$, there are no guards on y and z in c_G, and there are guards on both y and z in c_H. By a similar argument to the above case, this implies that there must be two guards on w in c_G. We can move one of these to x to defend the attack and maintain both invariants. This completes all cases, and so we have shown that $D_A(G; 1) \leq D_A(H; 1)$. However, from earlier we have that $D_A(H; 1) \leq D_A(G; 2) - 1$. This gives us:

$$D_A(G; 1) \leq D_A(H; 1) \leq D_A(G; 2) - 1 = D_A(G; 1) - 1$$

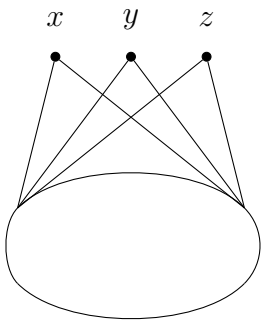


Figure 3.8: A diagram of the structure characterized in Theorem 3.18.

Where the last equality follows from our assumption that $I(G) \geq 2$. This is a contradiction, and so the result follows. □

We can use the above theorems to provide some specific structural characterizations of graphs with bounded invasion number.

Corollary 3.19. *Let G be a graph and v a vertex adjacent to exactly 2 leaves. Then $I(G) \leq 2$.*

Proof. Let l_1 and l_2 be two leaves adjacent to v. Observe that $N(l_1) \cap N(l_2) = \{v\}$, and that $G[\{v\}]$ is trivially a clique. The result follows from Theorem 3.17. □

Corollary 3.20. *Let G be a graph and v a vertex adjacent to at least 3 leaves. Then $I(G) = 1$.*

Proof. Let l_1, l_2, and l_3 be three leaves adjacent to v. Then $N(l_1) \cap N(l_2) \cap N(l_3) = \{v\}$ and $G[\{v\}]$ is trivially a clique. The result follows from Theorem 3.18. □

Corollary 3.21. *Let G be a graph with adjacent vertices x and y such that $deg(y) = 2$ and $deg(x) = 1$. Then $I(G) \leq 2$.*

Proof. Note that $N(x) = \{y\} \subseteq N[y]$ and $N[y] \setminus \{x\}$ induces a clique. The result follows from Theorem 3.16. □

The results above in particular allows us to bound the invasion number of trees.

Corollary 3.22. *Let T be a tree. Then $I(T) \leq 2$.*

Proof. This follows directly from Corollary 3.19, Corollary 3.21, and the fact that any tree must have a degree 2 vertex adjacent to a leaf or some vertex adjacent to 2 or more leaves. □

We conjecture that both of the above results could be strengthened to forbid invasion number 2, with the caveat that P_3 satisfies the conditions of both lemmas and has $I(P_3) = 2$.

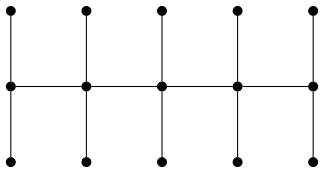


Figure 3.9: An example of the construction used in Theorem 3.23 to show that there are graphs with $D_A(G;a) \geq \frac{a}{a+1}|V(G)|$. This example is for the case $a = 2$ and $k = 5$.

3.6 The a-Attack Number of Trees

As our final result of this chapter, we set aside invasion numbers and instead consider the 2 attack eternal domination number of a graph. Our goal is to show that for any graph G, $D_A(G;2) \leq \lceil \frac{2}{3}n \rceil$. This result is obtained as a corollary of the identical result for trees, the proof of which heavily relies on a notion we refer to as *decoy constructions*.

Before we begin, we first show this upper bound is tight for all a.

Theorem 3.23. *For any a and n, there is a tree T with $|V(T)| \geq n$ such that*

$$D_A(T;a) \geq \left\lceil \frac{a}{a+1}|V(T)| \right\rceil$$

Proof. Let k be such that $(a+1)k \geq n$. We will construct a tree T with $(a+1)k$ vertices. Consider P_{a+1} with the vertices numbered sequentially from 1 to $a+1$. Construct T beginning with k copies of P_{a+1}, numbered $p_1, \ldots, p_k$. Then connect the vertex $\lfloor \frac{a+1}{2} \rfloor$ of p_i to the corresponding vertex of p_{i+1} for all $1 \leq i \leq k-1$. See Figure 3.9 for an example of this construction.

We claim that $D_A(T;a) = ak$, which is exactly $\lceil \frac{a}{a+1}(a+1)k \rceil$ as needed. To see that $D_A(T;a) \leq ak$, observe that we can put a guards on each p_i and eternally dominate each copy independently by Lemma 3.5.

Suppose for contradiction that $D_A(T;a) \leq ak - 1$. By the Pigeonhole Principle there is some p_i which contains at most $a-1$ guards. Consider attacking each vertex other than $\lfloor \frac{a+1}{2} \rfloor$ on p_i. Note that the only guards which can move to these vertices are the $a-1$ which were assigned to p_i, and so it is impossible to defend against these attacks. Thus, $D_A(T;a) \geq ak$ as needed. □

3.6.1 Decoy Constructions

Typically when working with a tree T one may wish to remove some vertices from T in order to obtain a smaller tree for which more results are known (e.g. by induction). Often only a single vertex needs to be removed in order to get these results, but many

reductions remove more than this. This leaves some room to *add* vertices to the tree when applying the reduction. We refer to these added vertices as the **decoy vertices**.

Given an eternal dominating strategy for this reduced graph, the decoy vertices allow us to make statements about the configuration of the guards in these strategies. If we then apply this strategy to the original graph, we can force guards to be in particular positions by identifying these decoy vertices with existing vertices in our original graph. This allows the construction of strategies which use fewer guards overall. We detail these constructions more in later sections, but begin with a quick lemma to simplify the algebra.

Lemma 3.24. *Let T be a tree with n_1 vertices and suppose we have applied some reduction to obtain a new tree T' such that $D_A(T';a) \leq \left\lceil \frac{a}{a+1}|V(T')| \right\rceil$ and $D_A(T;a) \leq D_A(T';a) + k$. Suppose T' is obtained by removing n_2 vertices and adding n_3 vertices, where $n_3 < n_2$. Then $D_A(T;a) \leq \left\lceil \frac{a}{a+1} \cdot n_1 \right\rceil$ if $\frac{k}{n_2 - n_3} \leq \frac{a}{a+1}$.*

Proof. We first show a bound on k in terms of n_2 and n_3:

$$\begin{aligned} \frac{k}{n_2 - n_3} &\leq \frac{a}{a+1} \\ k &\leq \frac{a}{a+1}(n_2 - n_3) \\ k &\leq \left\lfloor \frac{a}{a+1}(n_2 - n_3) \right\rfloor \end{aligned}$$

Where the last inequality follows from the fact that k must be an integer. Note that $|V(T')| = n_1 - n_2 + n_3$. Thus,

$$\begin{aligned} k + \left\lceil \frac{a}{a+1}(n_1 - n_2 + n_3) \right\rceil &\leq \left\lfloor \frac{a}{a+1}(n_2 - n_3) \right\rfloor + \left\lceil \frac{a}{a+1}(n_1 - n_2 + n_3) \right\rceil \\ &\leq \left\lceil \left\lfloor \frac{a}{a+1}(n_2 - n_3) \right\rfloor + \frac{a}{a+1}(n_1 - n_2 + n_3) \right\rceil \\ &\leq \left\lceil \frac{a}{a+1}(n_2 - n_3) + \frac{a}{a+1}(n_1 - n_2 + n_3) \right\rceil \\ &\leq \left\lceil \frac{a}{a+1} n_1 \right\rceil \end{aligned}$$

□

We will refer to reductions which satisfy $\frac{k}{n_2 - n_3} \leq \frac{a}{a+1}$ as **satisfactory**.

3.6.2 General Reductions

Before we considering the specific case of 2 attacks, we first show some general reductions which can be applied with any value of a. Let T be a tree and $x \neq y$ adjacent

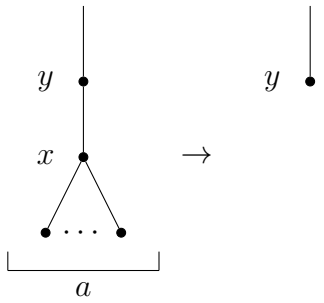


Figure 3.10: An example of the reduction described in Lemma 3.25.

vertices in T. Recall that the **subtree rooted at** x **with respect to** y is obtained by rooting T at y and considering the subtree rooted at x, and is denoted T_x^y.

Lemma 3.25. *Let T be a tree and suppose there are x and y such that $T_x^y \cong K_{1,a}$. Define T' by the reduction which deletes x and all leaves below it, and adds a guards to T. If $D_A(T';a) \leq \left\lceil \frac{a}{a+1}|V(T')| \right\rceil$, then this reduction is satisfactory.*

Proof. An example of this reduction can be seen in Figure 3.10. This reduction adds a guards and removes $a+1$ vertices (adding none), and so we need only show that there is an eternal dominating strategy for T which uses $a + \left\lceil \frac{a}{a+1}|V(T')| \right\rceil$ guards. Consider deleting the edge between x and y. This results in two trees, one isomorphic to $K_{1,a}$ by assumption. By Lemma 3.5 we can defend this with a guards as needed. □

Lemma 3.26. *Let T be a tree and suppose there are x and y such that $T_x^y \cong K_{1,t}$ with $t \geq 2a$. Define T' by the reduction which deletes x and all leaves beneath it, adds a leaves beneath y, and adds a guards to T. If $D_A(T';a) \leq \left\lceil \frac{a}{a+1}|V(T')| \right\rceil$ then this reduction is satisfactory.*

Proof. An example of this reduction can be seen in Figure 3.11. This reduction adds a guards, and removes a net $(t+1)-a \geq a+1$ vertices, so we need only show that there is an eternal dominating strategy for T which uses the correct number of guards. Let Λ be a strategy for T' which uses $\left\lceil \frac{a}{a+1}|V(T')| \right\rceil$ guards, and consider placing a guards on x. Suppose some set of vertices $A \subseteq V(T)$ has been attacked.

The attacks in $A \cap V(T')$ can be defended by following Λ. Note that any moves made by Λ will leave at least a guards in on y or the a decoy leaves by Lemma 3.4. Thus the attacks on $A \setminus V(T')$ can be defended with the guards on x. We then simulate an attack on $|A| - |A \cap V(T')|$ of the decoy leaves we added in our reduction. Copy the moves made to defend these attacks, but change them to instead move guards to x rather than the decoy leaves. This maintains that there are always a guards on x. □

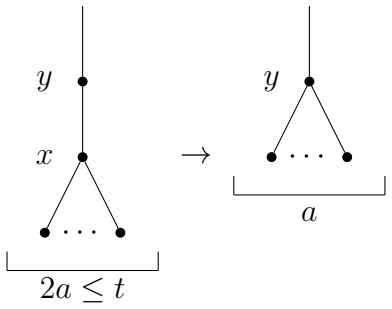


Figure 3.11: An example of the reduction described in Lemma 3.26.

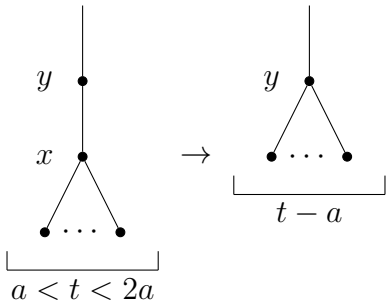


Figure 3.12: An example of the reduction described in Lemma 3.27.

Lemma 3.27. *Let T be a tree and suppose there are x and y such that $T_x^y \cong K_{1,t}$ with $a < t < 2a$. Define T' by the reduction which deletes x and all leaves beneath it, adds $t - a$ leaves beneath y, and adds a guards to T. If $D_A(T'; a) \leq \left\lceil \frac{a}{a+1}|V(T')| \right\rceil$ then this reduction is satisfactory.*

Proof. An example of this reduction can be seen in Figure 3.12. Note that this reduction adds a guards, and removes a net $(t+1) - (t-a) = a+1$ vertices, and so we need only show that there is an eternal dominating strategy for T which uses $a + \left\lceil \frac{a}{a+1}|V(T')| \right\rceil$ guards. Let L be the set of leaves adjacent to x in T and L' the set of leaves added in our reduction. Let Λ be an eternal dominating strategy for T' which uses $\left\lceil \frac{a}{a+1}|V(T')| \right\rceil$ guards.

Consider placing a guards on x in T and let $A \subseteq V(T)$ be some set of attacked vertices. We construct a second set of attacks $A' = (A \setminus L) \cup F'$ where $F' \subseteq L'$ with $|F'| = |L \cap A|$. Consider defending against A' in T' and copy the moves made to T. This will defend against all attacks of A on $V(T')$, and move $|F'|$ guards to x. We can then move the guards which were originally on x to the attacked leaves and defend against the attack, leaving a guards on x. □

The remaining cases are more difficult to deal with but can be dealt with in the case where $a = 2$.

3.6.3 The $a = 2$ Case

We now prove the main result of this section. We begin by showing a few specific reductions we need. Let z and y be such that T_y^z has height 2. Note that by our previous results we may assume that each vertex x below y has degree at most 2. Let x and l be vertices below y with $xl \in E(T)$ and $deg(x) = 2$ and $deg(l) = 1$. We refer to x and l as a twig. Let W_y be the set of twigs beneath y, and L_y the set of leaves adjacent to y.

Note that in each of the cases below we demonstrate the strategy with a figure. In these figures we depict the current state of both the game in the original tree and the reduced one. The vertices obtained by decoy constructions are marked with a star, and sample attacked vertices are marked with $\times$. The vertices removed in the reduction are those beneath the dashed line. Finally, not every possible state is shown; those missing are symmetric to one shown.

Lemma 3.28. *Let T and T_y^z be as given above with $|L_y| = 0$. Construct T' by applying the reduction which adds $|W_y|$ guards, deletes all vertices beneath y, and adds a vertex i adjacent to y. Then if $D_A(T'; 2) \leq \lceil \frac{2}{3}|V(T')| \rceil$ and $|W_y| \geq 2$, this reduction is satisfactory.*

Proof. Observe that we are adding $|W_y|$ guards and removing a net of $2|W_y|-1$ vertices, and thus the reduction satisfies the conditions to be satisfactory if $|W_y| \geq 2$. It remains to show that there is an eternal dominating strategy for T which uses $|W_y| + \lceil \frac{2}{3}|V(T')| \rceil$ guards. Figure 3.13 displays the strategy described below.

Let Λ' be an eternal dominating strategy for T' and note that if c' is a configuration in Λ', there must always be a guard on y or i. We will construct the configurations of our new strategy by playing 2 attack eternal domination on both T and T'; we refer to each game as $\mathcal{G}_T$ and $\mathcal{G}_{T'}$ respectively. Let $v_1, \ldots, v_k$ and $u_1, \ldots, u_k$ be the locations of the guards in $\mathcal{G}_T$ and $\mathcal{G}_{T'}$ respectively. We maintain as an invariants that if $u_j \neq i$, then $v_j = u_j$; if $u_j = i$ then $v_j \in N_T(y)$ and there is a guard on y in $\mathcal{G}_T$. Throughout out this proof, we will identify i with y in $\mathcal{G}_T$. We construct our initial configuration in $\mathcal{G}_T$ by copying some arbitrary configuration $c' \in \Lambda'$ and adding a guard on the leaf of each twig.

Let c be some configuration satisfying our invariants and consider a pair of attacks on the vertices p_1 and p_2. If $p_1 \in V(T')$ and $p_2 \in V(T')$, then we consider these attacks in $\mathcal{G}_{T'}$ and copy the corresponding moves in $\mathcal{G}_T$. To see why this is possible, for each guard not on i, their location must match in both games; and if any guard on i in $\mathcal{G}_{T'}$ moves, it must be to y, which is always possible by our invariants. If $p_1 \in V(T')$ and $p_2 \in (x_j l_j)$ for some twig $(x_j, l_j) \in W_y$, then we can consider the attack on p_1 in T' and again copy these response. We can then move the guard assigned to (x_j, l_j) to p_2 to defend that attack.

Therefore, assume that $p_1 = x_j$ and $p_2 = l_j$ for some twig $(x_j, l_j) \in W_y$. If there is not already a guard on i in $\mathcal{G}_{T'}$, then consider attacking y and i and copying the corresponding moves. As y and i have been identified with each other in $\mathcal{G}_T$, there must now be two guards on y, one of which has not moved. Move this guard to p_1 and the guard assigned to that twig to l_j. If there are guards on i in $\mathcal{G}_{T'}$, then by our invariants there must be a guard on y, and so we may move this guard to p_1 and the guard g to y. In $\mathcal{G}_{T'}$ we move g to y and the guard on y to i. This maintains our invariants in all cases, and the result follows. □

Lemma 3.29. *Let T and T_y^z be as described above with $|L_y| = 1$. Construct T' by applying the reduction which adds $|W_y|$ guards to T, deletes all vertices beneath y, and adds a vertex i adjacent to y. Then if $D_A(T'; 2) \leq \left\lceil \frac{2}{3}|V(T')| \right\rceil$ and $|W_y| \geq 1$, this reduction is satisfactory.*

Proof. Note that $\frac{|W_y|}{2|W_y|+1-1} \leq \frac{2}{3}$ for $|W_y| \geq 1$. By our decoy construction, we are guaranteed to always have a guard on y in T. Additionally, we can force two guards to be on y by attacking y and i in T', and can even let the guard on i move to vertices in the neighbourhood of y in T.

Figure 3.14 shows a strategy for defending T_y^z using the correct number of guards. All possible pairs of attacks within T_y^z are covered, as well as the possibility that vertices in T' are attacked. Additionally, each state can reconfigure to any other state (as well as itself if necessary). Thus, this is an eternal dominating strategy. □

Lemma 3.30. *Let T and T_y^z be as given above with $|L_y| = 2$. Construct T' by applying the reduction which adds $|W_y| + 1$ guards, deletes all vertices beneath y, and adds a vertex i adjacent to y. Then if $D_A(T'; 2) \leq \left\lceil \frac{2}{3}|V(T')| \right\rceil$ and $|W_y| \geq 1$, this reduction is satisfactory.*

Proof. Note that $\frac{|W_y|+1}{2|W_y|+1} \leq \frac{2}{3}$ if $|W_y| \geq 1$, as needed. By our decoy construction, we have that there is always a guard on y. If necessary, we can force 2 guard on y by attacking y and i in T', and additionally, the guard in T corresponding to the one on i can move to any vertex adjacent to y.

We now construct a 2 attack strategy for T using the proper number of guards. Figure 3.15 demonstrates this strategy. Note that all possible pairs of attacks within T_y^z are covered, as well as the possibility that vertices in T' are attacked. Thus, this is an eternal dominating strategy, and the result follows. □

Lemma 3.31. *Let T and T_y^z be as given above with $|L_y| = 3$. Construct T' by applying the reduction which adds $|W_y|$ guards, deletes all vertices beneath y and adds 3 decoy vertices i_1, i_2, and i_3. Then if $D_A(T'; 2) \leq \left\lceil \frac{2}{3}|V(T')| \right\rceil$ and $|W_y| \geq 1$, this reduction is satisfactory.*

Proof. Observe that $\frac{|W_y|}{2|W_y|+3-3} \leq \frac{2}{3}$ when $|W_y| \geq 1$, and so we need only demonstrate an eternal dominating strategy for T which uses the correct number of guards. By our decoy construction, we know that we always have at least 2 guards on $\{y, i_1, i_2, i_3\}$, and we can have up to at least 3 guards there by attacking some of $\{i_1, i_2, i_3\}$. Additionally, the guards on i_1, i_2, or i_3 can move to any vertices adjacent to y within T.

Figure 3.16 demonstrates a sufficient strategy. All possible pairs of attacks within T_y^z are covered, and all states shown can reconfigure to each other (and themselves when needed). Thus, this is an eternal dominating strategy. □

Lemma 3.32. *Let T and T_y^z be as given above with $|L_y| \geq 4$. Construct T' by applying the reduction which deletes all vertices below y, adds 4 decoy vertices $i_1, \ldots, i_4$ adjacent to y, and adds $|W_y|$ guards. Then if $D_A(T'; 2) \leq \lceil \frac{2}{3}|V(T')| \rceil$ and $|W_y| \geq 1$, this reduction is satisfactory.*

Proof. Note that $\frac{|W_y|}{2|W_y|+|L|-4} \leq \frac{2}{3}$ when $|L| \geq 4$ and $|W_y| \geq 1$. Thus all we need to do is demonstrate that there is a 2 attack eternal dominating strategy using the appropriate number of guards. Note that our decoy vertices allow us to guarantee that there are always at least 4 guards on y, and that the guards on $i_1, \ldots, i_4$ may in fact be on other vertices adjacent to y in T.

Figure 3.17 demonstrates this strategy. Note that all possible pairs of attacks within T_y^z are covered, and all states can reconfigure to each other (and themselves when necessary). Thus, this is an eternal dominating strategy. □

Lemma 3.33. *Let T and T_y^z be as given above. If $|W_y| = 1$ and $|L_y| = 0$, construct T' by applying the reduction which removes T_z^y from T and adds 2 guards. Then if $D_A(T'; 2) \leq \lceil \frac{2}{3}|V(T')| \rceil$ this reduction is satisfactory.*

Proof. This reduction adds 2 guards and removes 3 vertices, so we need only show an eternal dominating strategy for T which uses $2 + \lceil \frac{2}{3}|V(T')| \rceil$ guards. Observe that $T_y^z \cong P_3$ which has $D_A(P_3; 2) = 2$. Thus, we can remove the edge between y and z and defend this forest with $2 + \lceil \frac{2}{3}|V(T')| \rceil$ attacks, as desired. □

We are now equipped to prove the main result of this section, relying heavily on the above lemmas.

Theorem 3.34. *For any tree T on n vertices, $D_A(T; 2) \leq \lceil \frac{2}{3} \cdot n \rceil$.*

Proof. We proceed by induction on n. For our base cases, note that the result is true for all trees with at most 5 vertices by Lemma 3.5. Additionally, if T is a star with at least 6 vertices, then it can be eternally dominated with 4 guards, by Lemma 3.6. Thus the result is true for all trees with $diam(T) \leq 2$. Now suppose the result is true for all trees on at most $n-1$ vertices with diameter at least 3, and let T be a tree on n vertices and $diam(T) \geq 3$. By Lemma 3.24 the result is true if we can apply a

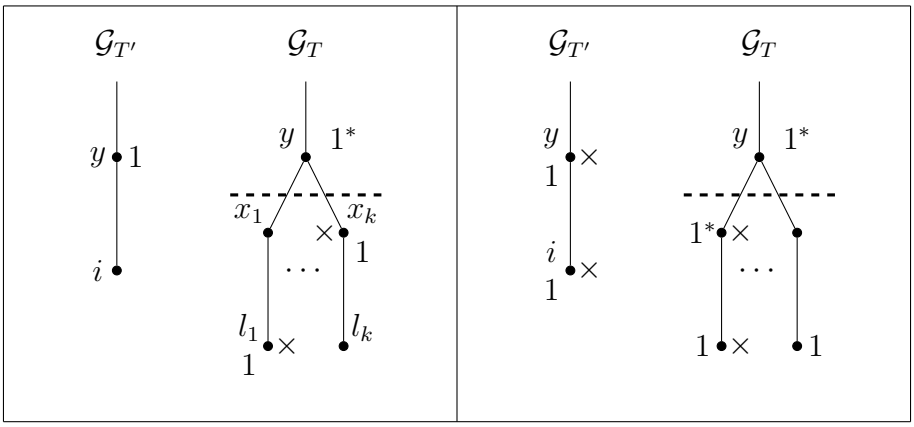


Figure 3.13: A general strategy for defending T_y^z when y is adjacent no leaves, and has at least 2 twigs. The states in each game are shown, and vertices below the dashed line in $\mathcal{G}_G$ are those removed by the reduction. Guards with a star are those provided by a decoy construction, and sample attacked vertices are marked with $\times$.

satisfactory reduction. If there exist x and y such that $T_x^y \cong K_{1,t}$ with $t \geq 2$, then we are done by induction and one of Lemma 3.25, Lemma 3.26, or Lemma 3.27 as appropriate. Thus we may assume that for any internal vertex x adjacent to exactly one internal vertex y, we have that x is adjacent to exactly one leaf.

Now suppose that there are vertices y and z such that T_y^z has height 2. Then in all cases, we are done by induction and Lemmas 3.28, 3.29, 3.30, 3.31, 3.32, or 3.33. □

Corollary 3.35. *For any graph G on n vertices, $D_A(G;2) \leq \lceil \frac{2}{3} \cdot n \rceil$.*

Proof. Let T be some spanning tree of G. By Theorem 3.34 there exists a 2 attack eternal dominating strategy for T, Λ_T, which uses $\lceil \frac{2}{3} \cdot n \rceil$ guards. We can apply this strategy to G. The result follows. □

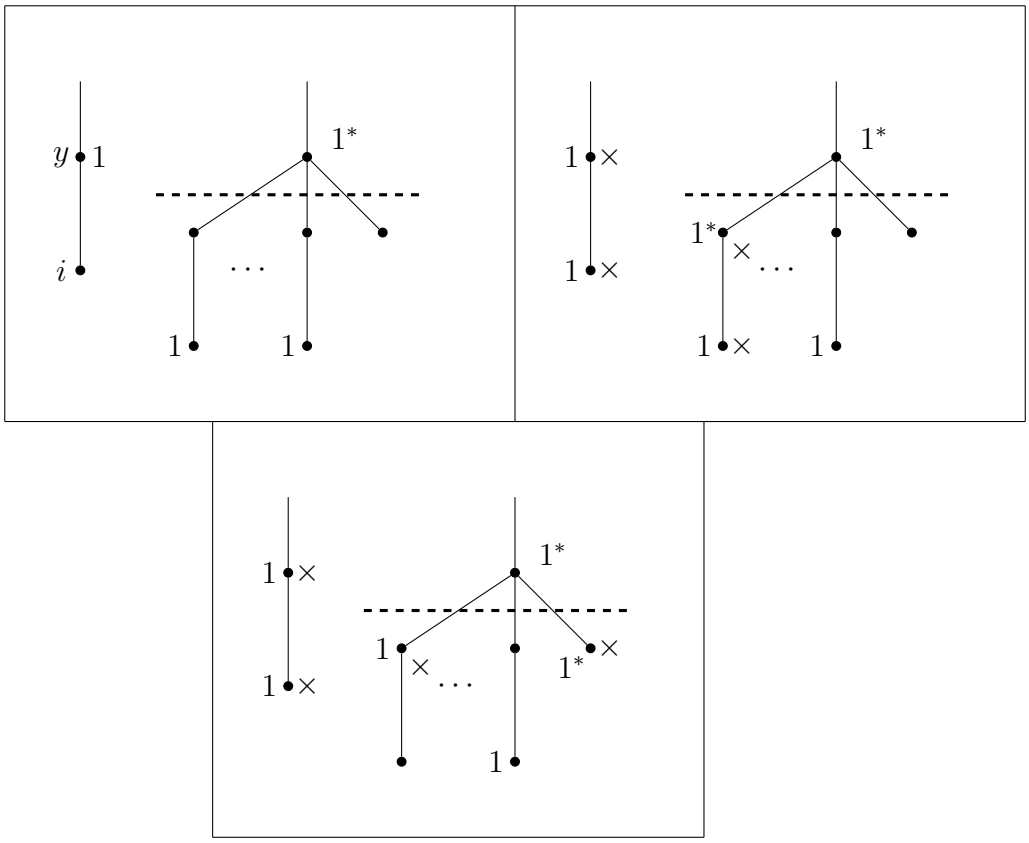


Figure 3.14: A general strategy for defending T_y^z when y is adjacent one leaf, and has any number of twigs. The states in each game are shown, and vertices below the dashed line in $\mathcal{G}_G$ are those removed by the reduction. Guards with a star are those provided by a decoy construction, and sample attacked vertices are marked with $\times$.

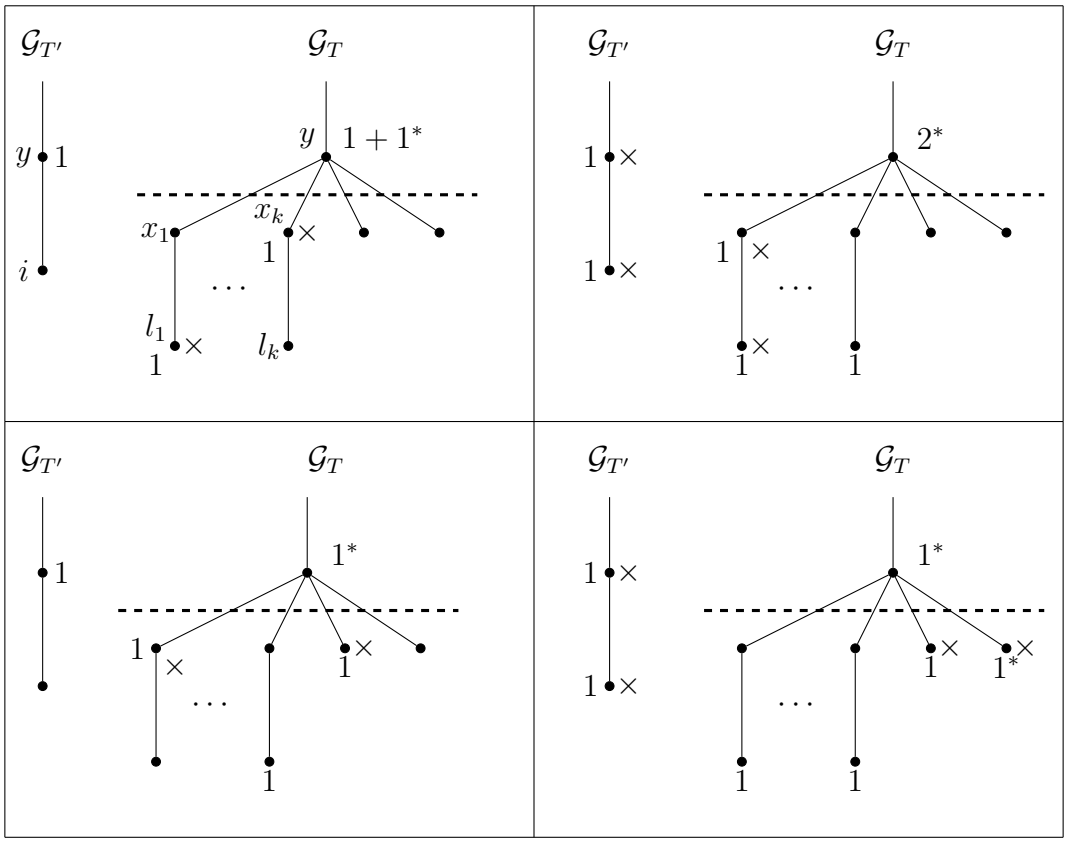


Figure 3.15: A general strategy for defending T_y^z when y is adjacent to 2 leaves, and has any number of twigs. The states in each game are shown, and vertices below the dashed line in $\mathcal{G}_G$ are those removed by the reduction. Guards with a star are those provided by a decoy construction, and sample attacked vertices are marked with $\times$.

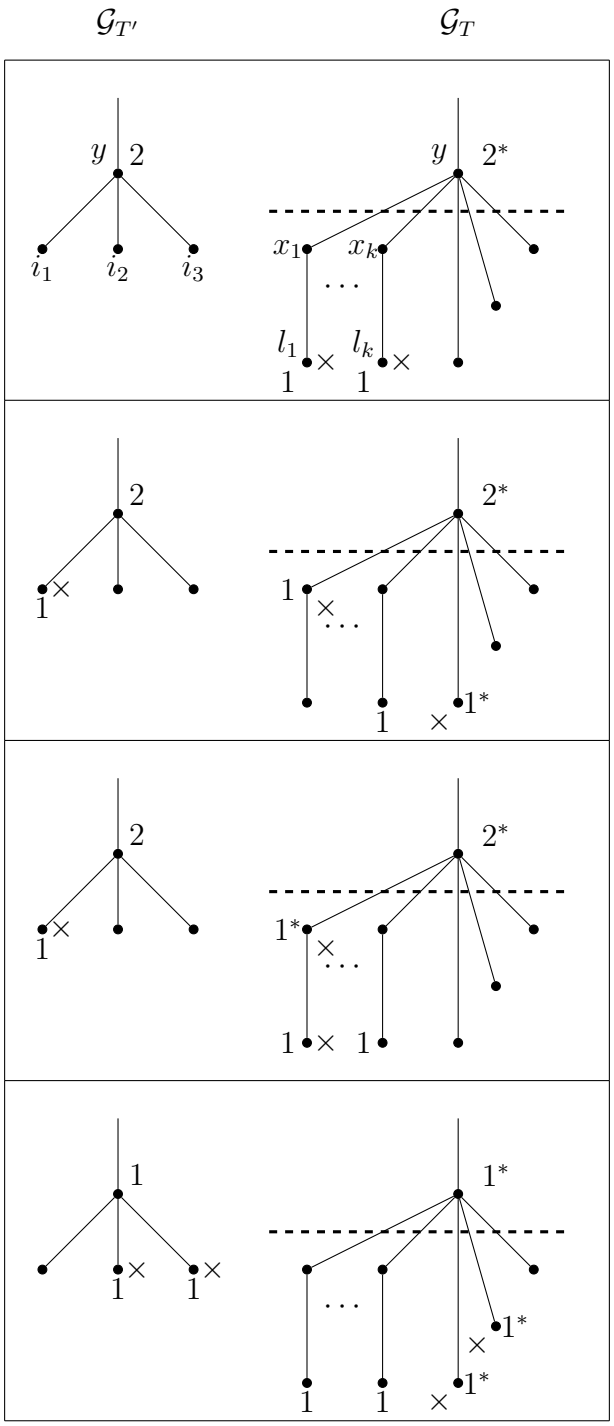


Figure 3.16: A general strategy for defending T_y^z when y is adjacent to 3 leaves, and has any number of twigs. The states in each game are shown, and vertices below the dashed line in $\mathcal{G}_G$ are those removed by the reduction. Guards with a star are those provided by a decoy construction, and sample attacked vertices are marked with $\times$.

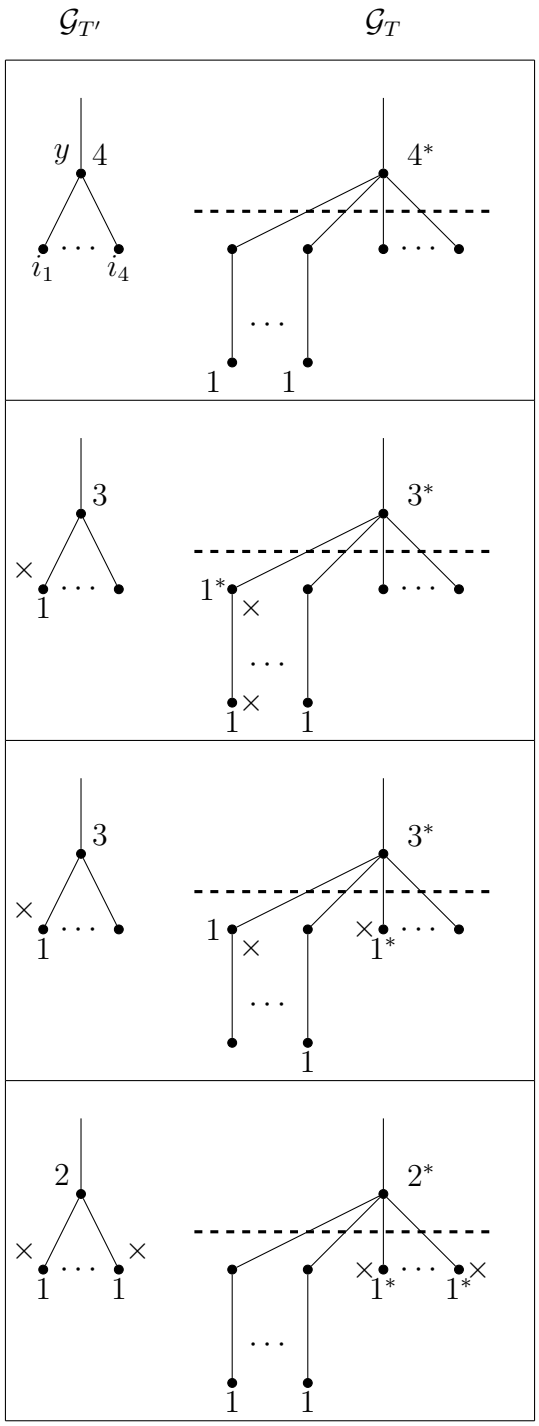


Figure 3.17: A general strategy for defending T_y^z when y is adjacent to at least 4 leaves, and has any number of twigs. The states in each game are shown, and vertices below the dashed line in $\mathcal{G}_G$ are those removed by the reduction. Guards with a star are those provided by a decoy construction, and sample attacked vertices are marked with $\times$.

Chapter 4
Stacking Numbers

4.1 Introduction

In this chapter we investigate the third of our three eternal domination parameters, the *stacking number* of a graph. Recall that our definition of the eternal domination number of a graph allows any number of guards to occupy a vertex. In most cases this is unnecessary and only a single guard is needed on any given vertex at any time.

Formally, Burger et al. [5] showed that allowing multiple guards to occupy a vertex does not save any guards in the 1 guard moves model. This led Goddard et al. [11] to conjecture that there were no graphs for which allowing multiple guards to occupy a vertex reduces the number of guards needed. However, this was proven false by Finbow et al [9], who showed that there are infinitely many graphs for which allowing 2 guards to occupy a vertex reduces the number of guards needed, and that the number of guards saved can be arbitrarily large.

In this chapter we use the notation $D_S(G;k)$ to denote the eternal domination number of G when at most k guards are allowed to occupy a vertex. We use $D_S(G;\infty)$ to denote the number of guards needed to eternally dominate G when any number of guards may occupy a vertex. Clearly as we increase k, the number of guards needed does not increase, and the minimum is achieved when we allow any number of guards to occupy a vertex. The least value of k for which this minimum is achieved is called *stacking number* of a graph and is denoted with $S(G)$.

Phrased in terms of this new parameter, the results of Finbow et al. showed that there are graphs which have stacking number at least 2. An example of such a graph is given in Figure 4.1. However the graphs they constructed all contain a cut-vertex, leading them to ask the following question "Does there exist a 2-connected graph for which the difference between $D_S(G;1)$ and $D_S(G;\infty)$ is arbitrarily large?" In this chapter we show that for any k and s there are k-connected graphs for which the gap between $D_S(G;1)$ and $D_S(G;s)$ is arbitrarily large. Our construction is a generalization of theirs. We also show that if G is a tree, then $S(G) = 1$.

The results in this chapter are joint work with Georgia Penner.

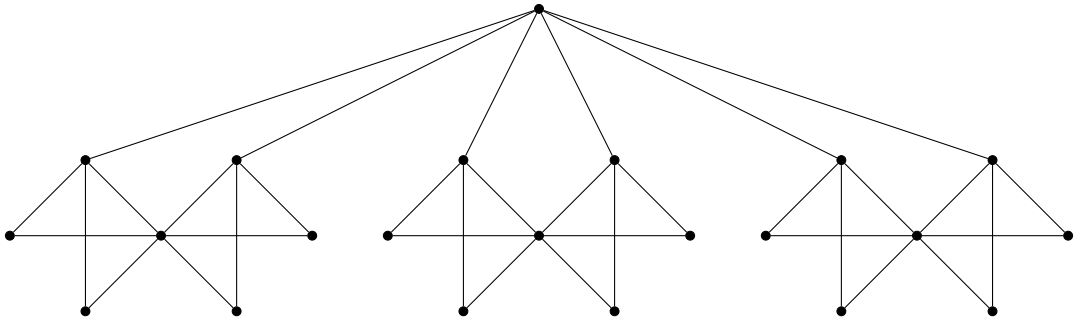


Figure 4.1: [9] An example of the graphs constructed by Finbow et al. to show that there are graphs with stacking number 2. Note that this graph is similar to the one in Example 4.1; these graphs formed the basis for our construction.

4.2 Definitions

Let G be a graph and Λ an eternal dominating strategy for G. The **stacking load** of Λ is the maximum number of guards which occupy a vertex in any state of Λ. Then, the **stacking number** of G is the minimum stacking load of any optimal strategy for G. We denote the stacking number with $S(G)$. An equivalent definition of the stacking number of G is the minimum value of k for which $D_S(G;k) = D_S(G;\infty)$.

The name stacking number is in reference to the notion of the guards being physical tokens occupying vertices, and needing to stack them on one another when multiple occupy a vertex. As mentioned in the introduction, the stacking number can be viewed as an example of how the number of guards needed may drop as the number of guards allowed to occupy a vertex increases. Accordingly, we define the **stacking sequence** of a graph G as $D_S(G;1), D_S(G;2), \ldots, D_S(G;\infty)$. The stacking number and the stacking sequence are clearly related, as the stacking number is simply the maximum index i such that $D_S(G;i-1) > D_S(G;i)$.

Our results in this chapter concern a particular kind of graph, which we refer to as an *airport*. This graph is constructed from several pieces called *wings*, *gates*, and *terminals*, which we define below:

Let $k \geq 1$ and $s \geq 2$ be integers. We refer to copies of $K_{k+2} - e$ as a k-**terminal** or simply a terminal if k is clear. If the missing edge is $e = (u, v)$ then we will call the vertices u and v the **matched pair** of the terminal, and the rest of the vertices the **gates**. Now let $H(k, s)$ be the graph defined as follows:

- Begin by taking s disjoint k-terminals.
- Add a universal vertex u.

We call such a graph a **wing** with **capacity** s. See Figure 4.2 for an example of these graphs within our larger construction.

As wings have a universal vertex, they have $D_S(G;1) = D_S(G;\infty) = 2$ by Lemma 1.8 and thus have stacking number 1. Additionally, they are not k-connected as their universal vertex is a cut-vertex. However, note that removing any other set of $k-1$ vertices is not a vertex cut. Thus, we can use several wings to construct a graph which has the properties we desire:

- Let $k \geq 1$ be a positive integer;
- $S = \{s_i\}$ be some strictly increasing finite sequence of integers bounded below by 2;
- $C = \{c_i\}$ be some finite sequence of integers such that for each i we have $2ks_i < c_i$.

Construct the graph $G(k, S, C)$ as follows:

- For each i, add c_i distinct copies of $H(k, k \cdot s_i)$, labelled $H_{i,1}, \ldots, H_{i,c_i}$.
- Add a copy of K_k disjoint from all wings and call it B[1].
- Join the vertices of B to the gates of each terminal by adding all possible edges between them.

We call $G(k, S, C)$ an **airport**.

Example 4.1. We now provide an example to illustrate each of the above definitions. Let G be the graph shown in Figure 4.2 where there are 7 copies each of the top and bottom modules. Note that the top modules are isomorphic to $H(1,3)$ while bottom modules are copies of $H(1,2)$. Then $G = G(1,(2,3),(7,7))$. We will determine the stacking sequence for this graph by considering $D_S(G;i)$ for each i.

When $i = 1$, one can verify that the best we can do is to put 1 guard on the central vertex, and then defend each module with two guards using the universal vertex present in each. Thus, $D_S(G;1) = 29$.

When $i = 2$, then we can save guards using the following strategy: place 1 guard on the universal vertex of each bottom module, and then 2 on the central vertex and 2 somewhere in its neighbourhood. We again defend the top modules with 2 guards each as above. To see that this strategy is an eternal dominating strategy, note that if there is an attack in a bottom module we can move the guard there to defend, and then move the 2 guards from the central vertex so that there is a guard on each terminal in that module. Finally, we move the 2 guards which began in the neighbourhood of our central vertex to the central vertex. This forms a dominating set, and the rest of the strategy can be generated similarly. This strategy uses 25 guards, and so $D_S(G;2) \leq 25$, and one can verify that in fact $D_S(G;2) = 25$.

[1]For baggage.

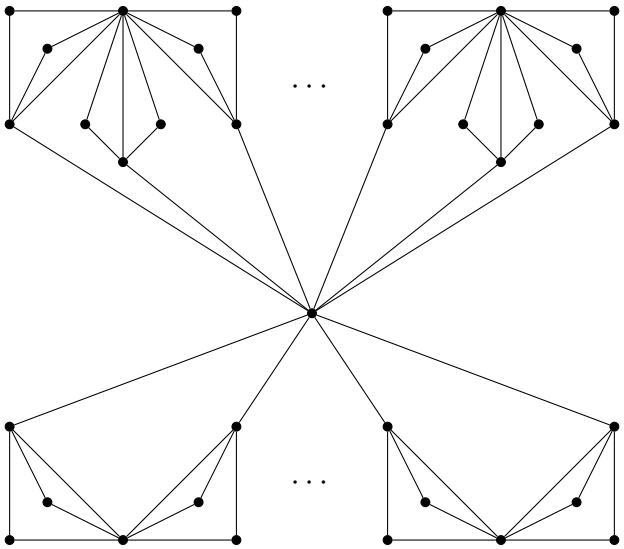


Figure 4.2: Let G be the graph above, where there are 7 each of the top and bottom modules. Then $S(G) = 3$ and the stacking sequence of G is $(29, 25, 20, 20, \ldots)$.

When $i = 3$, then we can again save guards in a similar fashion to the above. In this case we need to put 3 guards on the central vertex and 3 somewhere in its neighbourhood, and then we can remove 1 guard from each of the top modules and defend them similarly to the above strategy. Thus $D_S(G; 3) \leq 20$, but again it can be verified that $D_S(G; 3) = 20$. It turns out that allowing more guards to stack is not beneficial, and so G has stacking number 3 and stacking sequence $(29, 25, 20, 20, \ldots)$.

4.3 Results

As observed in the introduction, $D_S(G; i)$ is bound by a simple inequality chain:

Proposition 4.2. *For any graph G and integer k, we have*

$$\gamma(G) \leq D_S(G; \infty) \leq D_S(G; k) \leq D_S(G; 1)$$

Note that these inequalities show that any graph for which $\gamma(G) = D_S(G; 1)$ holds must have stacking number 1. We now show that an airport $G(k, S, C)$ is both k-connected and has $D_S(G; j-1) > D_S(G; j)$ exactly when $j = s_i$, with the size of the gap depending on c_i. We first show that $G(k, S, C)$ is k-connected:

Lemma 4.3. *The graph $G(k, S, C)$ is k-connected.*

Proof. Let $G = G(k, S, C)$, with B given as above. Let R be some set of $k-1$ vertices, and consider $G' = G \setminus R$. We will show that G' is connected, thus implying that G is k-connected.

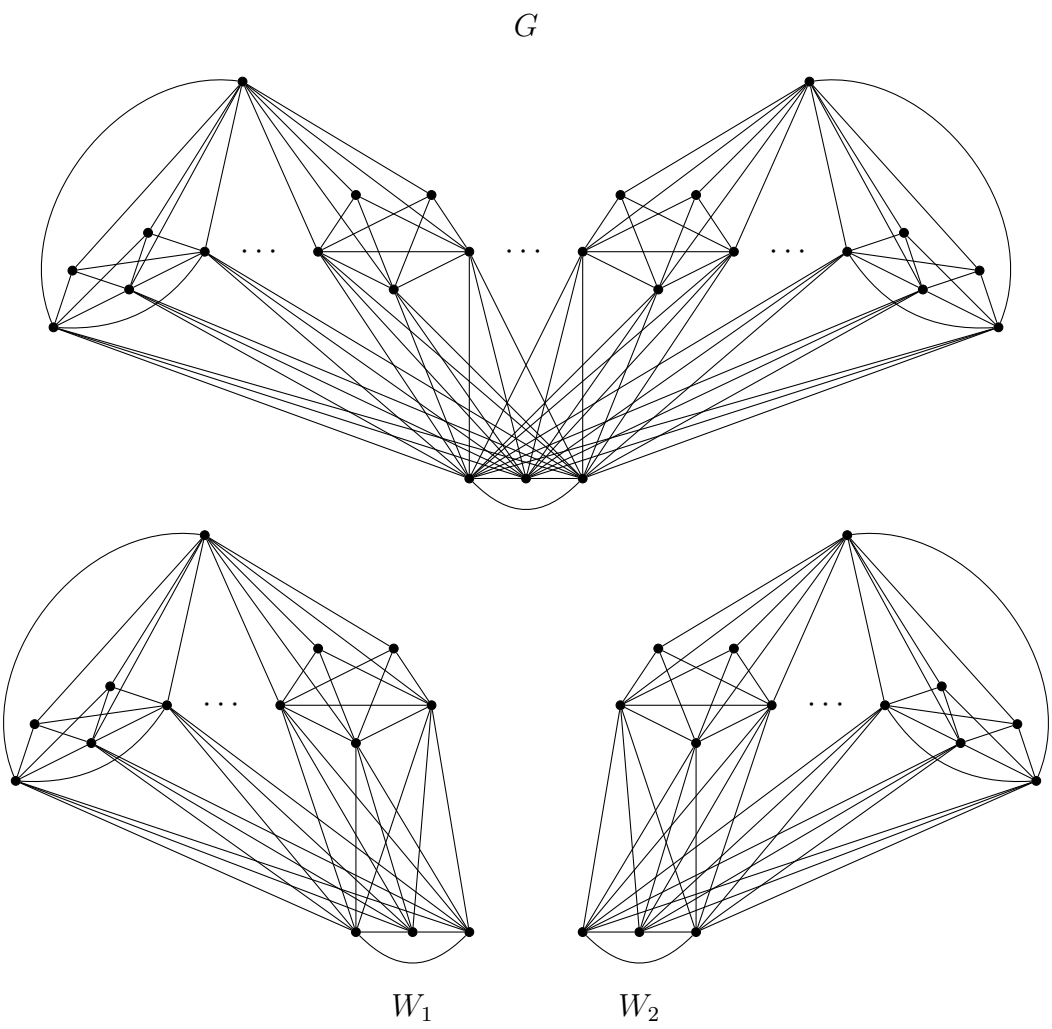


Figure 4.3: An example of the subgraphs of W_i, consisting of a single wing connected to the central clique of an airport. Note that many sections of the graph have been omitted in the interest of size.

As $|B| = k$, there must be some vertex $v \in B$ such that $v \notin R$. We will show that there is a walk from each other vertex to v. Let $u \neq v$ be some vertex in G'. If $u \in B$, or u is a gate in some terminal, then $uv \in E(G')$ and there is trivially a walk from u to v.

Let T be some terminal in wing $H_{i,j}$. By construction T has k gates and thus there must be at least one gate remaining. Call this vertex g. The matched pair in T are adjacent to g, as is the universal vertex of $H_{i,j}$. Thus there is a walk from each of these vertices to v. The result follows. □

We now prove a lemma regarding the nature of dominating sets of a wing in an airport:

Lemma 4.4. *Let $H = H(k, s)$ be a wing with universal vertex u. Let G' be the graph obtained by joining the gates of H to a clique (of any size) with all possible edges. If $D \subseteq V(G')$ is a dominating set not containing u, then D must contain at least one vertex from each terminal.*

Proof. Suppose for contradiction that D does not contain any vertices in some terminal. Let x and y be the matched pair in this terminal. These vertices are only adjacent to the gates of their terminal and u, and so D is not a dominating set. □

We now prove our main result in this chapter. The structure of this proof may seem unusual, but in fact leads directly to the characterization of the stacking sequence of an airport as shown in Corollaries 4.6 and 4.7. Parts of this proof are based on the proof of Theorem 2 in [9].

Theorem 4.5. *Let $G = G(k, S, C)$ be as above with $|S| = |C| = l$. Let (a_j) be the stacking sequence of G. Then (a_j) satisfies the following properties:*

1. *If $j < s_1$, then $a_j = 2\sum_{k=1}^{l} c_k + 1$.*
2. *For all i, if $j \geq s_i$, then $a_j \leq \sum_{k=1}^{i} c_k + 2\sum_{k=i+1}^{l} c_k + 2ks_i$*
3. *For all i, if $j < s_{i+1}$, then $a_j \geq \sum_{k=1}^{i} c_k + 2\sum_{k=i+1}^{l} c_k + 2ks_i$*
4. *If $j \geq s_l$, then $a_j \geq \sum_{k=1}^{l} c_k + 2ks_l$*

Proof. We will first prove that if $j < s_i$, then all wings with more than s_i gates must have at least two guards on them at the end of each round. Suppose for contradiction there is some wing which has fewer than 2 guards on it at the end of a round. If there are no guards, then we clearly cannot have a dominating set. Thus, there must be exactly 1 guard, which must be on that wings universal vertex u in order to have a dominating set.

If we then attack one of the matched vertices in the wing, The guard from u must necessarily move to defend against this attack. Any resulting configuration will not have a guard on u and thus by Lemma 4.4, we must have at least one guard on each terminal. The only guards that can move into the terminals must come from B, and by construction there can be at most kj of them. However since $j < s_i$ we have $kj < ks_i$ and so the resulting configuration cannot be a dominating set. The claim follows.

Proof of Statement 1 We need to show that when up to s_1 guards are allowed to stack, it is both necessary and sufficient to use $1 + 2\sum_{k=1}^{l} c_k$ guards to eternally dominate G.

To see that this number of guards is sufficient, consider the strategy which places a guard on the universal vertex of each wing, another guard somewhere else in each

wing, and then 1 in B. We can then partition the graph into B and a star for each wing, both of which can be defended by Lemmas 1.8 and 1.6.

We now show that $D_S(G;1) \geq 1 + 2\sum_{k=1}^{l} c_k$. Suppose we have only $2\sum_{k=1}^{l} c_k$ guards. By the above we must have at least 2 guards in each wing at the end of each round, and so there can be no guards in B at the end of any round. However if a vertex in B is attacked we must move a guard to B, a contradiction. Thus, we need at least $1 + 2\sum_{k=1}^{l} c_k$ guards, and the first statement follows.

Proof of Statement 2 We do this by showing that there is a strategy using

$$\sum_{k=1}^{i} c_k + 2\sum_{k=i+1}^{l} c_k + 2ks_i$$

guards. We will call those wings with more than s_i gates to be **high-capacity**; otherwise they will be called **low capacity**. Begin with the following initial configuration:

- Place 1 guard on the universal vertex in each wing.
- Place 1 additional guard on some other vertex in each high-capacity wing.
- Place s_i guards on each vertex in B.
- Arbitrarily place another ks_i guards on the gates of some wings.

This is clearly a dominating set as there is a guard on the universal vertex of each wing, and there are guards on the vertices of B.

We now show that this is sufficient. We will move the guards to maintain the following invariants:

- For each wing, there is a guard on u or there is some gate with two guards and one guard in every other gate.
- There are always s_i guards on each of the k vertices in B; additionally there are always ks_i guards in the neighbourhood of the clique.

Clearly these invariants hold for our initial configuration. Suppose there is an attack somewhere in G. If the attack is one of the high-capacity wings, it can be defended using the guards in that wing by Lemma 1.8. If the attack is in one of the low capacity wings, there are two cases depending on the current location of the guard in that wing:

- If the guard is on u, it can move to the attacked vertex. By Lemma 4.4 we must now have at least one guard in each terminal. There are at most ks_i terminals in the low capacity wings, so we can move the guards we need from B to a gate in each terminal.

- If there are no guards on u, then there is some terminal with two guards; let g be one of these guards. If the attack was on u, we can move g to u. Otherwise there is some path from g to the attacked vertex through B which has guards on every internal vertex so by Lemma 1.10 we can defend the attack.

In any of the above cases, we may need to make some of the following moves in order to maintain the invariants:

- If a low capacity wing was not attacked, we move its guard to its universal vertex.
- If there are less than ks_i guards on B, move guards from the wings to B so that there are ks_i guards on B.

To see that the invariants have both been maintained, note the following:

- Each non-attacked low-capacity wing now has a guard on its universal vertex.
- If a low capacity wing was attacked, then we have moved a guard to each gate from the clique, and have also moved the guard from u to one of the gates.
- Each high capacity wing always has 2 guards in it, one of which is always located on u.
- To begin, there were $2ks_i$ guards distributed between B and the terminals of G. Since we always end with ks_i guards on B, and these guards never leave $N[B]$, there must be ks_i guards somewhere in $N(B)$.

Thus our invariants are always maintained, and so we have an eternal dominating strategy. This completes the proof of statement 2.

Proof of Statement 3 Suppose for contradiction that we have an eternal dominating strategy which uses strictly fewer than $\sum_{k=1}^{i} c_k + 2\sum_{k=i+1}^{l} c_k + 2ks_i$ guards. By Lemma 4.4 we know there are at least $2\sum_{k=i+1}^{l} c_k$ guards in the high-capacity wings. In order to have a dominating set there must be at least one guard in each low-capacity wing, so these wings hold at least $\sum_{k=1}^{i} c_k$ guards. This leaves at most $2ks_i - 1$ guards whose locations we have not determined.

By assumption there are at least $2ks_i + 1$ wings with ks_i terminals. Thus there are at least two such wings with exactly one guard on them, call them w_1 and w_2. If we attack a matched vertex in w_1, then we can move the guard which was already in w_1 to the attacked vertex. By Lemma 4.4 we know that we need to move in at least ks_i guards to now form a dominating set. After this attack there are $ks_i - 1$ guards whose locations we do not know. We can now attack a matched vertex in w_2. Similarly we must move in ks_i guards to form a dominating set. However we only have $ks_i - 1$ guards available, so this is impossible, and the statement follows.

Proof of Statement 4 Similarly to the above, suppose we have strictly fewer than $\sum_{k=1}^{l} c_k + 2ks_l$ guards. In order to form a dominating set, we must place one guard on each wing, of which there are $\sum_{k=1}^{l} c_k$. This leaves at most $2ks_l - 1$ guards unaccounted for.

By assumption there are at least $2ks_l + 1$ wings with capacity s_l, and as such there are at least 2 such wings which have only 1 guard on them. Label them as w_1 and w_2. If we attack a matched vertex in w_1, we must move the guard on w_1 to the attacked vertex. By Lemma 4.4 we then need to move another ks_l guards in in order to form a dominating set. Thus we have at most $ks_l - 1$ guards whose locations are undetermined. Attack a matched vertex in w_2. Similarly, we must move ks_l guards in to the wing in order to dominate it. However, we do not have enough guards available, and so we did not have an eternal dominating set. □

Corollary 4.6. *Let $G = G(k, S, C)$ and consider $D_S(G; j)$. Then:*

- *If $j < s_1$, then $D_S(G; j) = 2\sum_{k=1}^{l} c_k + 1$.*
- *If $s_i \leq j < s_{i+1}$, then $D_S(G; j) = \sum_{k=1}^{i} c_k + 2\sum_{k=i+1}^{l} c_k + 2ks_i$.*
- *If $s_l \leq j$, then $D_S(G; j) = \sum_{k=1}^{l} c_k + 2ks_l$*

Proof. All of these claims follow directly from applying the results of Theorem 4.5. □

Corollary 4.7. *Let $G = G(k, S, C)$ as above. Then the stacking sequence for G decreases exactly at each $s \in S$.*

Proof. This follows directly from the characterization of the stacking sequence of G given in Theorem 4.5. □

We can now apply Theorem 4.5 to show that there are in fact graphs with arbitrary stacking number and connectivity.

Corollary 4.8. *For any k and s there is a k-connected graph G with $S(G) = s$.*

Proof. Let k and s be given and consider $G = G(k, (s), (2s + 1))$. Observe that the stacking number is equivalent to the last index at which the stacking sequence of G decreases. By Corollary 4.7 this occurs exactly at s for the graph G. Thus G has stacking number s, and is k-connected by Theorem 4.3. □

We can also use the characterization of the stacking sequence of an airport to show that one can completely specify the location and size of any decreases in a graph's stacking sequence.

Corollary 4.9. *Let k be some integer and A some positive non-increasing sequence of integers. Let $S'_i = (i : a_i - a_{i-1} < 0)$ and let $C'_i = (2(a_{i-1} - a_i) + 1 : i \in S_i)$. Then A is a stacking sequence for a k-connected graph when the following conditions hold:*

- $\max A = 2\sum c'_i + 1$
- $\min A = k \cdot \max s'_i + \sum c'_i$
- $2ks_i < c_i$ *holds for all* i.

Proof. Consider the graph $G(k, S', C')$. The result then follows from Theorem 4.6. □

4.4 Stacking Number of Trees

We conclude this chapter by proving that trees have stacking number 1. This is exactly the result one likely expects, and the proof uses very similar techniques to the others involving trees in this thesis. Note that we apply the reductions for generating neocolonizations shown in Chapter 2, but do not use neocolonizations themselves.

Theorem 4.10. *Let T be a tree. Then $S(T) = 1$.*

Proof. Let T be a tree. We will prove this result by induction on both the order and diameter of T. Suppose for induction that the result is true for all trees T' with $diam(T') < diam(T)$, and also for all trees T' where $diam(T') = diam(T)$ but $|V(T')| < |V(T)|$. The result is obviously true for trees with diameter at most 1. Any tree with diameter 2 is a star, and by Lemma 1.8 we know that we can defend T with 2 guards. Clearly there is no advantage to placing both of these guards on a single vertex. Now, by Theorem 4.2 we need only show that $D_S(T; \infty) \geq D_S(T; 1)$. We have two cases depending on which reductions we can apply to T.

First suppose that we can apply R2 to delete the vertex x and an adjacent leaf y from T to obtain T'. By induction we know that $D_S(T'; \infty) = D_S(T'; 1)$, and by Theorem 2.4 we know that $D_S(T; 1) = D_S(T'; 1) + 1$. Suppose for contradiction that $D_S(T; \infty) < D_S(T; 1)$; then there is some strategy Λ which has stacking load $k \geq 2$ and uses fewer than $D_S(T; 1)$ guards. Note that by definition of a dominating set, there must always be a guard on x or y in every state of Λ. Let Λ' be the strategy where we have removed any moves along the edge connecting x to the rest of T. Then Λ' must be an eternal dominating strategy when restricted to T', which uses at most $D_S(T; \infty) - 1$ guards. Thus

$$D_S(T'; \infty) \leq D_S(T; \infty) - 1 < D_S(T; 1) - 1 = D_S(T'; 1),$$

Which is a contradiction.

Next suppose that we obtain T' by applying R1 to delete the leaves $l_1, \ldots, l_t$, with $t \geq 2$. By induction we again know that $D_S(T'; \infty) = D_S(T'; 1)$, and again by Theorem 2.4 we know that $D_S(T; 1) = D_S(T'; 1)+1$. Suppose for contradiction that $D_S(T; \infty) < D_S(T; 1)$ and let Λ be a strategy which uses this number of guards. We construct an

eternal dominating strategy Λ' for T' as follows: copy Λ, and for each state let g be some guard which is either on x or some l_i (guaranteed to exist by definition of a dominating set). Delete g from that state, and then move any other guards on a leaf l_i to x. This clearly forms an eternal dominating strategy for T' which uses $D_S(T;\infty)-1$ guards, and we again get a chain of inequalities:

$$D_S(T';\infty) \leq D_S(T;\infty) - 1 < D_S(T;1) - 1 = D_S(T';1),$$

Which is a contradiction. Thus $S(T) = 1$ for any tree T. □

Chapter 5
Eternal Roman Domination

5.1 Introduction

This chapter is focused on another variant of eternal domination. Here, we instead require that the guards form a different sort of dominating set. The idea of a *Roman dominating set* arises from the military strategy of Constantine the Great, and was a predecessor to eternal domination.

We complete the circle in this chapter by investigating eternal Roman domination. We prove an upper bound on the eternal Roman domination number of a graph, which we also show to be best possible. It clearly suffices to prove the result for trees, as we can generalize any bound for trees to one for all graphs by taking a spanning tree.

5.2 Definitions

Let $G = (V, E)$ be a graph. A **Roman dominating function** of G is a function $r : V \to \{0, 1, 2\}$ such that every vertex x with $r(x) = 0$ is adjacent to a vertex y with $r(y) \geq 2$. The **weight** of a Roman dominating function r is $\sum_{v \in V} r(v)$, and the **Roman domination number** of a graph is the minimum weight of a Roman dominating function, denoted $\gamma_R(G)$ or γ_R when G is clear from context. Recall that a k-configuration is an assignment of k guards to the vertices of a graph. We say that a k configuration c has the **Roman dominating property** or is a **Roman dominating configuration** if the function $r' : V \to \mathbb{Z}$ defined by $r'(v) = |c(v)|$ is a Roman dominating function.

The **eternal Roman domination game** can be viewed as the specific instance of the eternal domination game $\mathcal{R} = (G, \gamma_R, 1, \infty, 2)$ where γ_R indicates that every configuration must be a Roman dominating configuration. We denote the **eternal Roman domination number**, $D(G; \mathcal{R})$, as $D_R(G)$.

Example 5.1. We provide an example of the eternal Roman domination game. Figure 5.1 shows two Roman dominating configurations of P_4. In either one, any attack on

Figure 5.1: Two eternal Roman dominating configurations on P_4.

a vertex without a guard can be defended by reconfiguring to the other one. This demonstrates that $D_R(P_4) \leq 3$; as it is not even possible to form a Roman dominating set with only 2 guards, we have that $D_R(P_4) = 3$. In general we have $D_R(P_n) = \left\lfloor \frac{3}{4}(n+1) \right\rfloor$ [13].

5.3 Decoy Constructions

Throughout our proof of the main result we will work exclusively with trees, and remove subtrees in order to apply an induction hypothesis. Similar to the proof of Theorem 3.34, we will sometimes add vertices and employ a decoy construction. Two of these constructions are driven by the following lemma:

Lemma 5.2. *Let x and y be adjacent vertices such that x has degree 1 and y degree 2. Then, for any eternal Roman dominating strategy:*

- *There is always at least 1 guard on x or y.*
- *If y is attacked, then there are at least 2 guards on the vertices in $\{x, y\}$.*

Proof. For the first claim note that if it were not true, then x has no guards on it and is not adjacent to any vertices with 2 guards, a contradiction.

For the second claim, suppose for contradiction that it is not true. Then there is exactly 1 guard on an end of the edge xy. If this guard is on x, then we have not defended against the attack on y. If this guard is on y, then we do not have a Roman dominating configuration. The result follows. □

The decoy constructions work on the principle that we can add some new vertices when applying a reduction (as long as some yet-to-be-detailed conditions are met), and force a structure like the one in Lemma 5.2 to appear in the reduced tree. These new vertices do not appear in our original tree, and we will see that we can use the guards which appear on them in the structure removed by the reduction, as long as we obey certain rules to not disrupt the strategy given for the reduced tree.

Suppose we have applied a reduction that removes the subtree rooted at x. The first way to apply the decoy construction is to add a new decoy vertex i below x's parent, and when considering a strategy on the whole tree, identify it with x. This allows us to have a guard appear on x by simulating an attack on it, with the restriction that we cannot move this guard from x unless it is replaced by another guard.

The second way to apply the decoy construction is to add two new decoy vertices, i_1 and i_2 with i_2 adjacent to i_1 and i_1 adjacent to x's parent. Again when considering a strategy on the whole tree we identify both i_1 and i_2 with x. By Lemma 5.2 we know that this forces a guard to always be on x, but also that we can have 2 guards on x by simulating an attack on i_1. As above, this is valid as long as these guards do not leave x unless replaced by another guard.

In both the above decoy constructions, it is necessary that we have a state which corresponds to the case where some vertex other than i_1 or i_2 has been attacked in T'. For the first case, this means one where there is no decoy guard on x, and in the second only a single decoy guard.

We apply the decoy constructions in both ways in this thesis, and will denote the guards provided by them with superscript $*$. We will differentiate between the applications by denoting those reductions with I for the first construction and II for the second.

5.4 Main Result

In this section we prove a tight upper bound on the eternal Roman domination number of trees. Because the eternal Roman domination number of a connected graph is at most the eternal Roman domination number of a spanning tree, an upper bound on the eternal Roman domination number of all graphs is obtained as a consequence.

Proposition 5.3. *If T is a tree then $D_R(T) \leq \lceil \frac{5n}{6} \rceil$.*

Proof. We prove this result by considering rooted trees and focusing on particular subtrees which appear. We show that these subtrees can individually be defended with at most $\lfloor \frac{5k}{6} \rfloor$ guards, where k is the number of vertices in the subtree. We can then combine this with an inductive strategy to get a strategy which uses at most $\lceil \frac{5n}{6} \rceil$ guards and defends the whole tree. Complicating this matter is that some of our reductions *add* vertices as well as remove them, though in general the total number of vertices decreases. This requires a touch more arithmetic. The following lemma captures the situation.

Lemma 5.4. *Let T be a tree with n_1 vertices and suppose we have applied a reduction to get a new tree T'. Suppose that this reduction adds k guards, removes n_2 vertices, and adds n_3 vertices to the resulting tree, where $n_3 < n_2$. If $\frac{k}{n_2 - n_3} \leq \frac{5}{6}$, then $D_R(T) \leq \lceil \frac{5n_1}{6} \rceil$.*

Proof. Note that by induction we can defend T' with at most $\lceil \frac{5}{6} \cdot (n_1 - n_2 + n_3) \rceil$ guards.

$$\begin{aligned} & \frac{k}{n_2 - n_3} \leq \frac{5}{6} \\ \iff \quad & k \leq \frac{5}{6}(n_2 - n_3) \\ \iff \quad & k \leq \left\lfloor \frac{5}{6}(n_2 - n_3) \right\rfloor \end{aligned}$$

Where the last inequality holds because k is an integer. Thus when defending the whole tree, we need at most $k + \lceil \frac{5}{6} \cdot (n_1 - n_2 + n_3) \rceil$ guards. The statement follows, as

$$\begin{aligned} k + \left\lceil \frac{5}{6} \cdot (n_1 - n_2 + n_3) \right\rceil &\leq \left\lfloor \frac{5}{6}(n_2 - n_3) \right\rfloor + \left\lceil \frac{5}{6} \cdot (n_1 - n_2 + n_3) \right\rceil \\ &\leq \left\lceil \left\lfloor \frac{5}{6}(n_2 - n_3) \right\rfloor + \frac{5}{6} \cdot (n_1 - n_2 + n_3) \right\rceil \\ &\leq \left\lceil \frac{5}{6}(n_2 - n_3) + \frac{5}{6} \cdot (n_1 - n_2 + n_3) \right\rceil \\ &\leq \left\lceil \frac{5n_1}{6} \right\rceil \end{aligned}$$

□

We now need two things to complete the proof: a set of reductions satisfying the above condition and the number of guards needed for those trees to which we cannot apply any of the reductions. We begin with the reductions.

We call a reduction **satisfactory** if it satisfies the conditions in Lemma 5.4. In the following we will consider rooting the tree at some vertex and considering one of the subtrees beneath the root. Let T be a tree and $x \neq y$ adjacent vertices in T. Recall that the **subtree rooted at x with respect to y** is obtained by rooting T at y and considering the subtree rooted at x, and is denoted T_x^y.

Lemma 5.5. *Let x and y be vertices such that T_x^y is a star with L leaves. Then if $L \geq 3$, the following is a satisfactory reduction:*

- *Add 2 guards.*
- *Construct T' by removing T_x^y and adding a new vertex i joined to y.*

Proof. We first show that we can extend a strategy Λ' for defending T' to one which defends T with only 2 more guards. We can augment the strategy for defending T' by adding the following states:

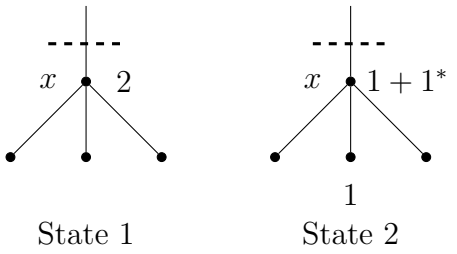


Figure 5.2: The two types of states added when applying the reduction given in Lemma 5.5

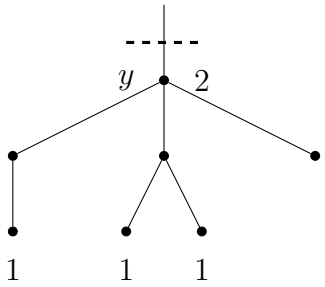


Figure 5.3: A general height 2 subtree.

- For each state with no guard on i, add a corresponding state where 2 guards are placed on x.
- For each state in Λ' with a guard on i, add a corresponding state where there are 2 guards on x and one on l for each $l \in L$. One of these guards comes from the decoy construction.

Examples of these states are given in Figure 5.2. We can reconfigure between each state added here by moving a guard off of x to the attacked leaf, and then simulating an attack on i in T'. Thus, this new strategy is eternal Roman dominating. This is a satisfactory reduction as $\frac{2}{(L+1)-1} < \frac{5}{6}$ when $L \geq 3$. □

By repeatedly applying the reduction in Lemma 5.5, we may assume that our tree has no rooted subtree isomorphic to a star with 3 or more leaves. Let y be some vertex with parent z and consider T_y^z. Assume it has height 2. Let L be the number of leaves adjacent to y in T_y^z. Let x be some arbitrary neighbour of y in T_y^z where $deg(x) \geq 2$. Since T_y^z has height 2, all neighbours of x other than y are leaves, and by assumption there are at most 2 such leaves. Let X_1 be the number of such neighbours adjacent to 1 leaf, and X_2 the number such neighbours which are adjacent to 2 leaves. A general depiction of such a tree is given in Figure 5.3.

Definition 5.6. Define the **general height 2 reduction** with respect to a rooted subtree T_z^y as follows:

- Add $2 + X_1 + 2X_2$ guards.
- Construct T' by deleting T_y^z and adding a new vertex i below z.

We now characterize when this reduction is satisfactory, and provide alternate reductions in almost all of the cases when it is not.

Lemma 5.7. *The general height* 2 *reduction is satisfactory when*

$$12 \leq 4X_1 + 3X_2 + 5L$$

Proof. We first show that there is an eternal Roman dominating strategy which uses $X_1 + 2X_2 + 2$ guards: place a guard on each leaf at distance 2 from y; never moving these guards, while defending the remainder of T_y^z as if it were a star. The condition quickly follows from the definition of satisfactory. □

As we are now considering height 2 subtrees, we may assume that $X_1 + X_2 \geq 1$; thus the only cases where the general reduction is not satisfactory are when (L, X_1, X_2) is any of the following triples:

$$\begin{array}{ccccc} (0,0,1) & (0,0,2) & (0,0,3) & (0,1,0) & (0,1,1) \\ (0,1,2) & (0,2,0) & (0,2,1) & (1,0,1) & (1,0,2) \\ & & (1,1,0) & & \end{array}$$

We show that there exist satisfactory reductions for all of these except $(0, 1, 0)$. In each case, we delete T_y^z and add some number of guards. Further, in two of the cases we use a decoy construction. The strategies for these cases are depicted in Figure 5.7 and Figure 5.8. Recall that we denote guards provided by the decoy construction with a superscript $*$.

When we cannot apply remove any height 2 rooted subtrees, we will need another general reduction. Consider vertices z and w such that the subtree rooted at z with respect to w has height 3. A general depiction of such a tree can be found in Figure 5.4. Let L, X_1, and X_2 be as for the general height 2 reduction, and additionally define Y_{010} be the number of vertices y in T_z^w such that T_y^z is isomorphic to the height 2 tree characterized by the triple $(0, 1, 0)$.

Definition 5.8. Define the **general height** 3 **reduction** with respect to a rooted subtree T_z^w of height 3 as follows:

- Add $2 + 2 \cdot Y_{010} + X_1 + 2 \cdot X_2$ guards.
- Construct T' by deleting T_z^w and adding an decoy vertex below w.

Lemma 5.9. *The general height* 3 *reduction is satisfactory when*

$$12 \leq 3Y_{010} + 4X_1 + 3X_2 + 5L$$

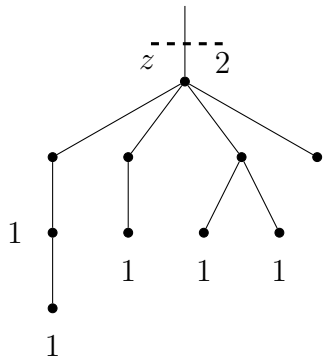


Figure 5.4: A depiction of a general height 3 subtree.

Proof. The proof is similar to the one given for the general height 2 reduction. □

As with the general height 2 reduction, the general height 3 reduction is satisfactory in all but a few cases. Namely, if (Y_{010}, X_1, X_2, L) is any of the following quadruples:

$$\begin{array}{ccccc} (1,0,0,0) & (1,0,0,1) & (1,1,0,0) & (1,0,1,0) & (2,0,0,0) \\ (1,0,1,1) & (1,2,0,0) & (2,0,0,1) & (1,1,1,0) & (2,1,0,0) \\ & (1,0,2,0) & (2,0,1,0) & (3,0,0,0) & \end{array}$$

There are satisfactory reductions in all of these cases, consisting of removing T_z^w and adding some number of guards, as well as in some cases applying one of the decoy constructions. The strategies for all but one reduction can be found in Figures 5.9, 5.10, 5.11, and 5.12.

We now handle the last remaining exception: $(2,1,0,0)$, depicted in Figure 5.5. We use a decoy construction unique to this subtree. We claim that the following reduction is satisfactory:

- Add 5 guards.
- Delete T_z^w and add 3 new vertices, i_1, i_2, and i_3 such that they form a path of length 3 beneath w.

This reduction satisfies the conditions to be satisfactory. We need to show that it gives an eternal Roman dominating strategy. By the induction hypothesis, since i_1, i_2, i_3 is a path of length 3 we know there are always at least 2 guards on i_1 and i_2, and that if i_3 is attacked we will have 3 guards, with one on i_3. When constructing a strategy for the whole tree we identify i_1 and i_2 with z, but leave i_3 alone. We place the 5 guards on the vertices of depth 2 and 3. This configuration is also depicted in Figure 5.5.

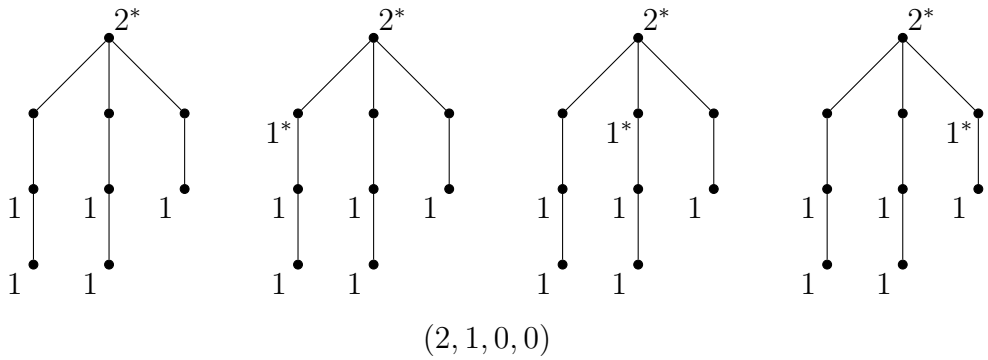


Figure 5.5: The exceptional case $(2, 1, 0, 0)$ which needs to be handled especially exceptionally.

If an unguarded vertex is attacked, we simulate an attack on i_3 in T' and copy the strategy except for the move that places a guard on i_3. We replace this move with one that moves a guard from z to the attacked vertex. If another unguarded vertex in this subtree is attacked, we again simulate an attack on i_3 in T'; note that since this has happened twice in succession we are in fact attacking an already guarded vertex, and can assume no guards move. In T however, we move a guard from z to the attacked vertex and move the guard on the previously attacked vertex back to z. This corresponds to swapping the guards on i_3 and i_2 in T', but this does not matter as guards are indistinguishable.

Note that since this covers all cases for height 3 subtrees, this gives us a complete set of reductions. We now only need to consider the base cases to which we cannot apply any of the above reductions. Note that any tree of diameter 4 or higher has a height 3 subtree, which we can remove by one of our reductions. So we need only consider those trees of diameter at most 3. Additionally, we also need only consider those which cannot have one of our reductions applied to them. Also note that because of Lemma 5.4 we only need to show that these graphs can be defended with $\lceil \frac{5n}{6} \rceil$ guards.

Note that the only tree of diameter 0 is K_1, which needs 1 guard; this meets the bound. There is only one tree of diameter 1, which is K_2. This needs $2 = \lceil \frac{5 \cdot 2}{6} \rceil$ guards. The trees of diameter 2 are stars, and if they have 4 or more leaves, we can apply the height 1 reduction. Otherwise, we have either $K_{1,2} \cong P_3$, which needs $3 = \lceil \frac{5 \cdot 3}{6} \rceil$ guards, or $K_{1,3}$ which can be defended with $3 < \lceil \frac{5 \cdot 4}{6} \rceil$ guards. Finally, the trees with radius 3 are double stars. If either centre is adjacent to more than 2 leaves we can apply either the height 1 reduction (if there are 3 or more leaves), or the $(0, 0, 1)$ height 2 reduction. Thus we may assume that the graph is isomorphic to P_4, which needs $3 < \lceil \frac{5n}{6} \rceil$ guards (as illustrated in Figure 5.1). This completes the proof of the main result. □

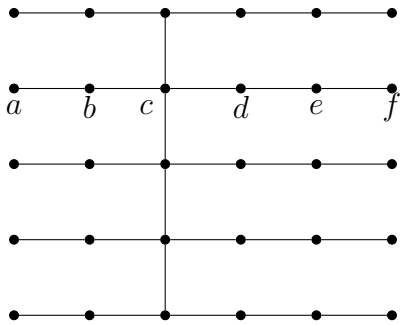


Figure 5.6: The graph G_5, with the labelling used in the proof of the tightness of the bound.

5.5 The Bound is Tight

In this section we provide an example of an infinite family of graphs which need exactly $\lceil \frac{5n}{6} \rceil$ guards to be eternally Roman dominated. Let G_k be the tree obtained by taking k copies of P_6 and joining the third vertex in each successive copies with one another. The tree G_5 is shown in Figure 5.6.

Theorem 5.10. *For any n there exists a graph with $|V(G)| > n$ and $D_R(G) = \frac{5}{6}|V(G)|$.*

Proof. Choose k such that $6k > n$ and let G_k be as given above. We can eternally Roman dominate G_k with $5k$ guards by defending each copy of P_6 independently with 5 guards. We now show that this is optimal. Suppose we try to eternally Roman dominate G_k with $5k-1$ guards. Then by the Pigeonhole Principle, for any state there is some copy of P_6 with at most 4 guards on it. Since each state is a Roman dominating configuration, we must have exactly 4 guards on this copy. We consider the possible ways these guards could be split between the sets $\{a, b\}$ and $\{d, e, f\}$ as labelled in Figure 5.6.

Clearly if there are 0 guards in $\{a, b\}$, then we do not have a Roman dominating configuration. Similarly if there are 3 or more, then there are not enough guards remaining to form a Roman dominating configuration on $\{d, e, f\}$. If there is only 1 guard, then it must be on a and there must be 2 guards on c. This then leaves only a single guard for $\{d, e, f\}$ with which it is impossible to form a Roman dominating configuration. Thus we can assume there are 2 guards on $\{a, b\}$, placed arbitrarily. This leaves only 2 guards for $\{d, e, f\}$, which must be placed on e to form a Roman dominating set. Note that this implies there are no guards on c. If we then attack d, the guard which defends it must come from e. No matter how this occurs, either f or e will have no guards stationed on it and neither will be adjacent to a vertex with 2 guards. Thus $5k - 1$ guards cannot be used to eternally Roman dominate G_k. □

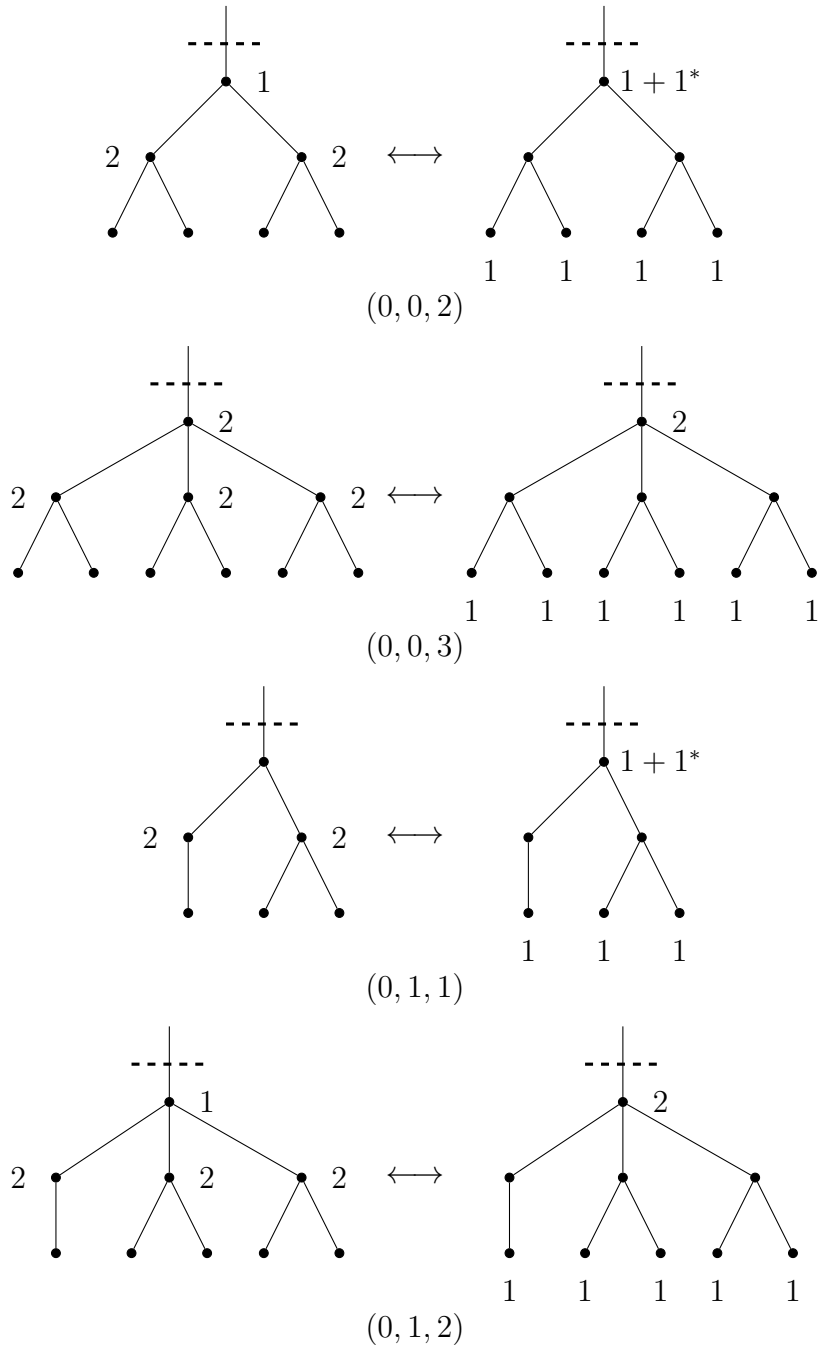


Figure 5.7: Satisfactory reductions for some of the exceptional height 2 subtrees.

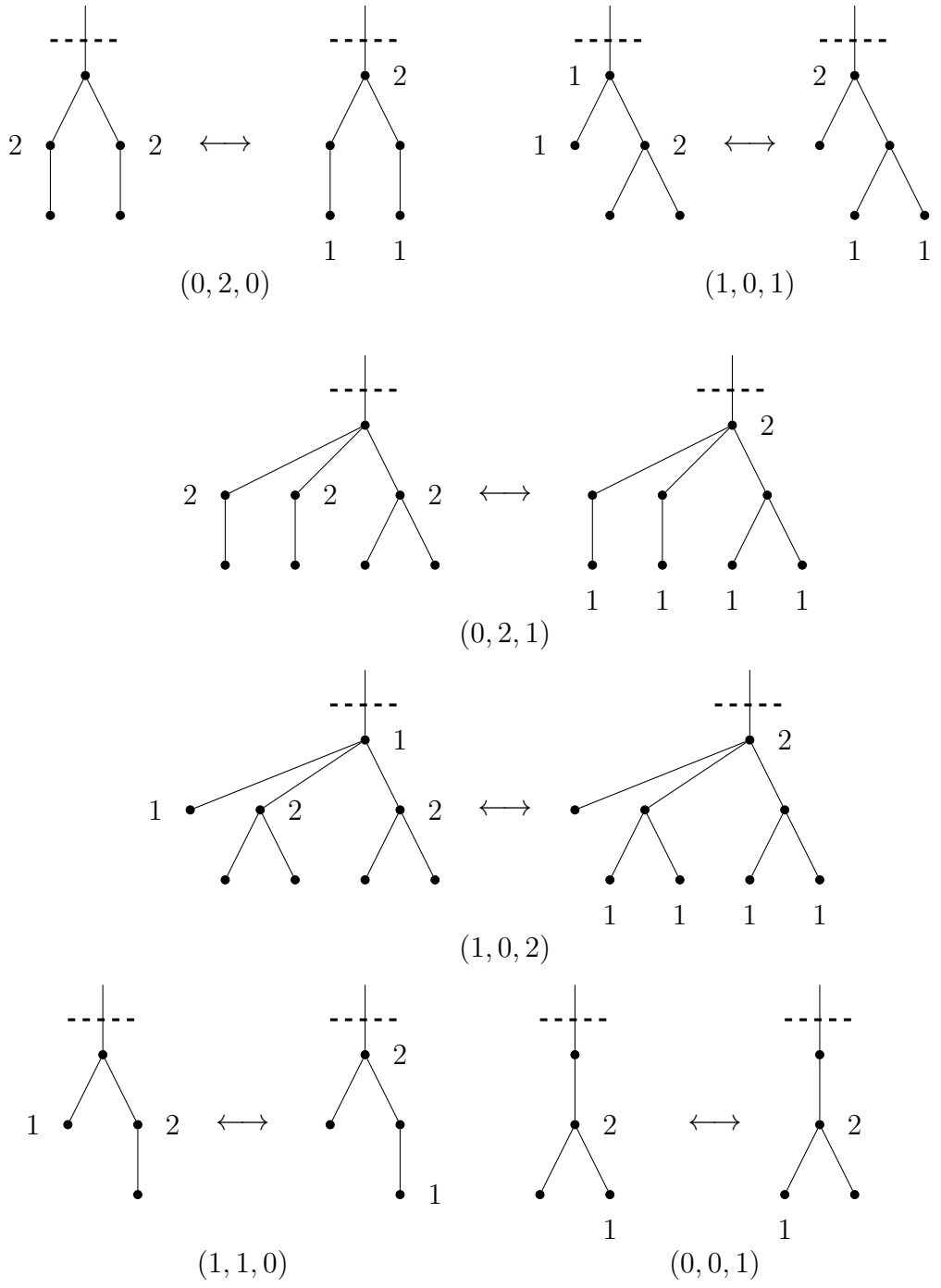


Figure 5.8: Satisfactory reductions for some of the exceptional height 2 subtrees.

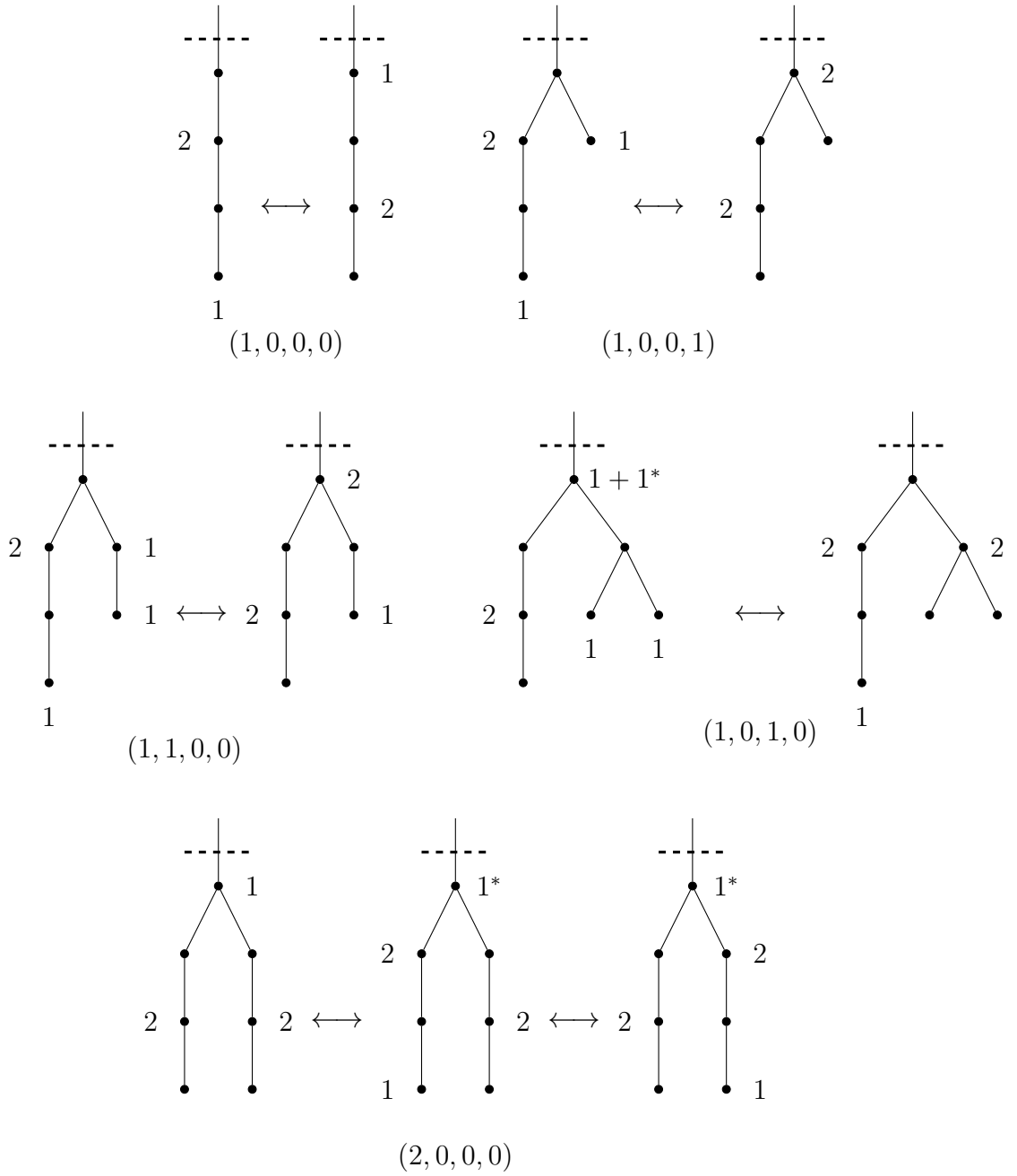


Figure 5.9: Satisfactory reductions for some of the exceptional height 3 subtrees.

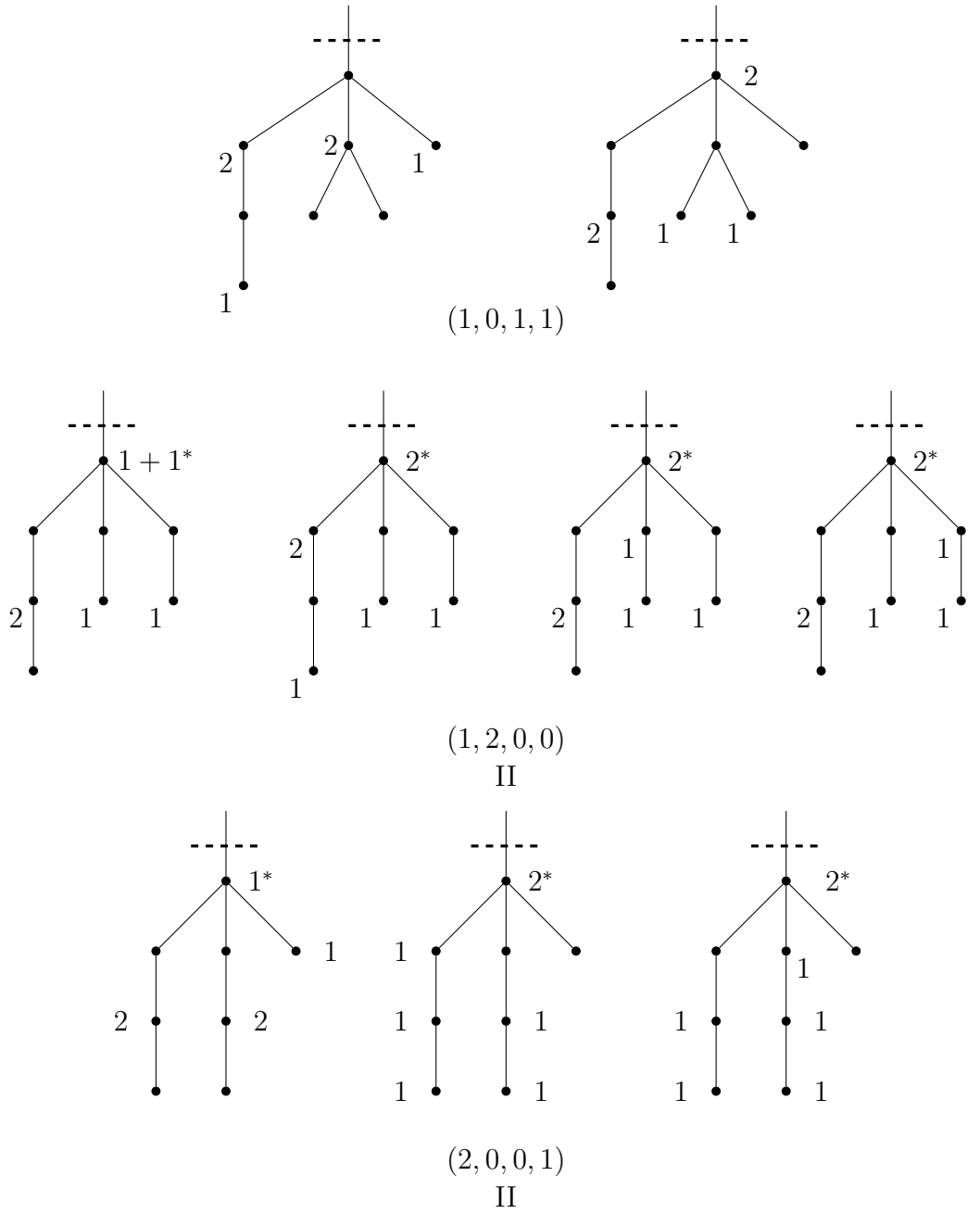
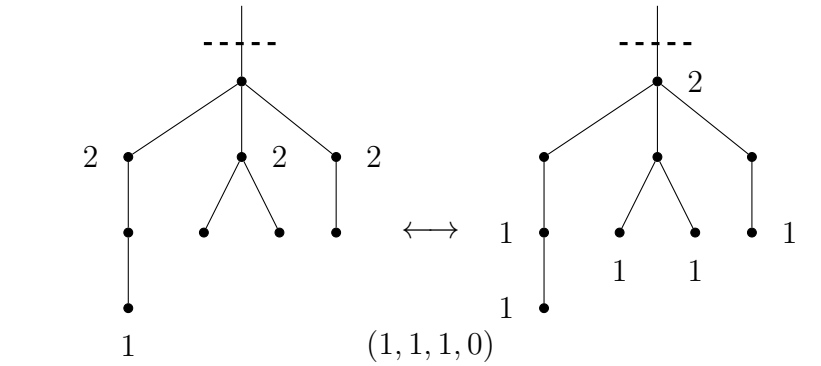
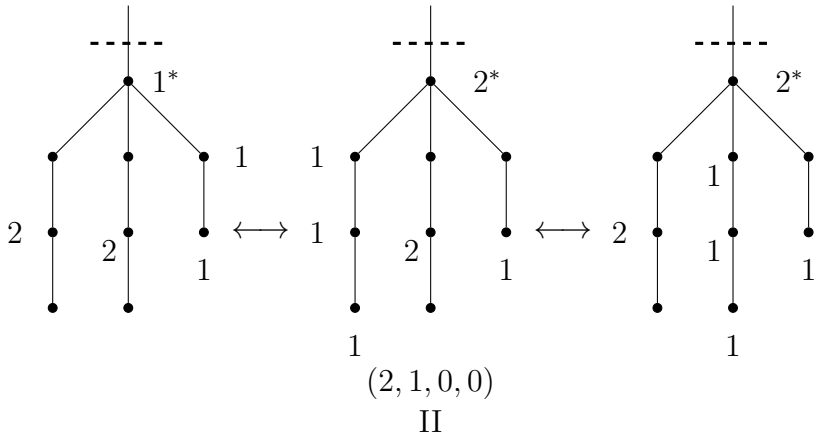


Figure 5.10: Satisfactory reductions for some of the exceptional height 3 subtrees.





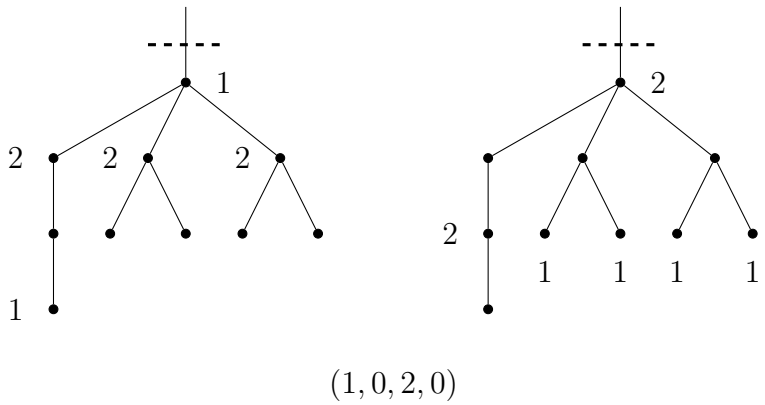
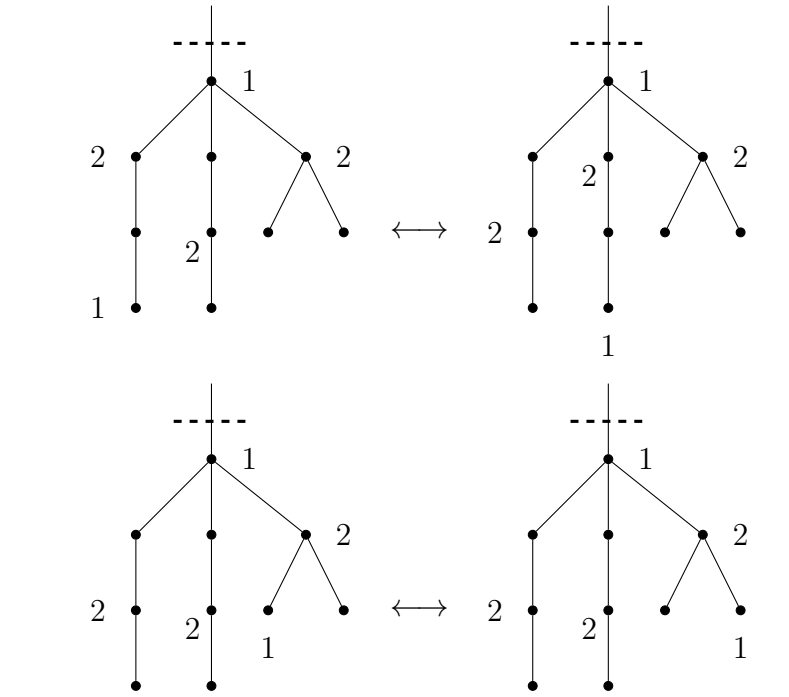
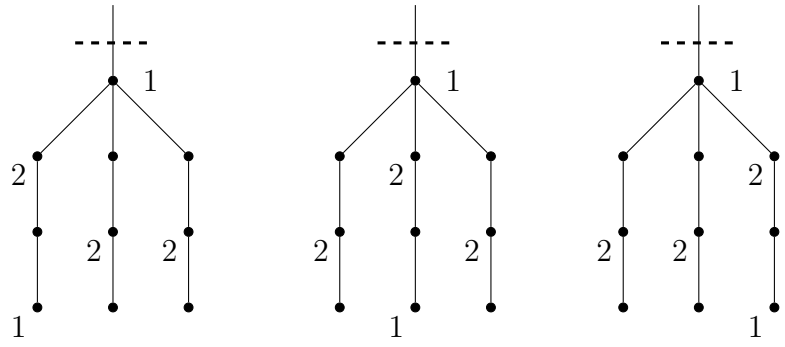


Figure 5.11: Satisfactory reductions for some of the exceptional height 3 subtrees.



$(2, 0, 1, 0)$



$(3, 0, 0, 0)$

Figure 5.12: Satisfactory reductions for some of the exceptional height 3 subtrees.

Chapter 6
Algorithms

6.1 Introduction & Definitions

In this chapter we present a general purpose algorithm which can be used to solve variations of the eternal domination problem. We then present a range of applications of this algorithm, in particular to the variations presented earlier in this thesis. The pseudocode for each algorithm can be found at the end of this chapter.

Recall that we use $\mathcal{P} = (G, I_\gamma, a, m, s)$ to denote *instances* of the eternal domination problem, where

- G is a graph.
- I_γ is a property each configuration of guards must satisfy.
- a is the number of simultaneous attacks being considered.
- m is the maximum number of guards which can move when reconfiguring.
- s is the maximum number of guards which can occupy a vertex.

Given an instance $\mathcal{P}$ we call a configuration **valid** if it places no more than s guards on any vertices and satisfies I_γ. For any valid configuration c we say that another configuration c' is a **subconfiguration** of c if for all $v \in V(G)$ we have $c'(v) \subseteq c(v)$ and c' is itself a valid configuration. Conversely, we call c' a **superconfiguration** of c if c is a subconfiguration of c'.

We call a set S of configurations an **upset** if for every $c \in S$ we have that all superconfigurations of c are also in S. This terminology is analogous to that of subsets, supersets, and upsets in set theory.

6.2 A General Purpose Algorithm

We begin by presenting a high-level description of our general algorithm. Computing the eternal domination boils down to successively answering the question of whether

or not there exists an $(h, \mathcal{P})$ strategy. Let l and u be two integers such that we know

$$l \leq D(\mathcal{P}) \leq u$$

We can determine the actual value for $D(\mathcal{P})$ in one of two ways:

- We can successively query whether or not there exists a $(h, \mathcal{P})$ strategy for each value of $h \in [l, u]$. This method works regardless of the instance $\mathcal{P}$.
- We can utilize a process similar to binary search where we query whether or not a $(h, \mathcal{P})$ strategy exists for $h = \frac{l+u}{2}$ and then update l or u accordingly.

The binary search method can only be applied if the existence of a $(h, \mathcal{P})$ strategies implies the existence of $(h', \mathcal{P})$ strategies for $h' > h$, and the non-existence of any $(h, \mathcal{P})$ strategies implies the non-existence of $(h', \mathcal{P})$ strategies for $h' < h$. These conditions are generally true for the problems we have investigated, and so we use this method by default. Note that if no bounds are known, $l = 1$ and $u = |V(G)|$ are sufficient in most cases.

This approach merely shifts the problem of determining $D(\mathcal{P})$ to the problem of determining if a $(h, \mathcal{P})$ strategy exists for any h. We cover this (and other subproblems) next. First, we briefly prove the correctness of this algorithm.

Theorem 6.1. *Algorithm 2 determines $D(\mathcal{P})$ for a given instance $\mathcal{P}$.*

Proof. By assumption we know $l \leq D(\mathcal{P}) \leq u$. We also assume that the existence of a strategy with h guards implies the existence of strategies with $h' > h$ guards, and the non-existence of a strategy with h guards implies the non-existence of a strategy with $h' < h$ guards.

Thus, we never remove $D(\mathcal{P})$ from the list of values we may check when conducting binary search, and as such will eventually check if it contains a $(h, \mathcal{P})$ strategy. By assumption we will find such a strategy successfully. Additionally, whenever we check if there is a $(h, \mathcal{P})$ strategy for $h < D(\mathcal{P})$, we will not find a strategy, and the result follows. □

6.2.1 Finding $(h, \mathcal{P})$ Strategies

Recall that an $(h, \mathcal{P})$ strategy is a subgraph Λ of the $(h, \mathcal{P})$ reconfiguration graph such that for every set of vertices $A \subseteq V(G)$ and h-configuration $c \in V(\Lambda)$, we have that $|c(v)| \geq 1$ for all $v \in A$ or c is adjacent to some other configuration c' such that $|c'(v)| \geq 1$ for all $v \in A$.

Fix h and $\mathcal{P}$ and let $\mathcal{R}$ be the $(h, \mathcal{P})$ reconfiguration graph. We could find such a subgraph by first constructing the entirety of $\mathcal{R}$ and then determining if a $(h, \mathcal{P})$ strategy exists. In practice however, $\mathcal{R}$ is very large and we often need only a small

part of it to find a strategy. Therefore, we adopt the practice of constructing subgraphs of $\mathcal{R}$ and periodically checking if a strategy exists, whilst also removing configurations we determine cannot be in any strategy. In particular, we need only consider each connected component of $\mathcal{R}$, as any $(h, \mathcal{P})$ strategy is a connected graph.

Our algorithm for finding $(h, \mathcal{P})$ strategies goes as follows: begin by iterating over each valid configuration c. We have two cases

- If we have not seen c yet, generate the connected component it is a part of.
- If we have seen c before, skip this configuration.

The reason for this distinction is that when generating the component containing a particular vertex we will visit many other vertices, and thus will have already generated the component containing them. When generating one of these components, we may find that it contains an $(h, \mathcal{P})$ strategy. If so, we return that strategy. Otherwise, return to checking all valid configurations. If we exhaust every possible valid configuration without finding an $(h, \mathcal{P})$ strategy, then no such strategy exists and we return as such. Pseudocode for this algorithm can be found in Algorithm 3. We now prove the correctness of this algorithm:

Theorem 6.2. *Algorithm 3 correctly determines if there is a $(h, \mathcal{P})$ strategy for fixed h and $\mathcal{P}$.*

Proof. First note that as there are at the very most $s^{|V(G)|}$ possible valid configurations, Algorithm 3 terminates. Note that any $(h, \mathcal{P})$ strategy will be a connected graph, and as such will be a subgraph of some component of $\mathcal{R}$.

First, assume that there is a $(h, \mathcal{P})$ strategy Λ in $\mathcal{R}$. Then there exists some component C such that Λ is a subgraph of C. Let c be the first configuration in c to be generated by `Enumerate-Valid-Configurations`$(h, \mathcal{P})$. Note that $c \notin S$ at this point, as if it were, this would imply that there was some configuration in $c' \in C$ which was generated earlier. Thus we run `Generate-Component`$(c, \mathcal{P})$ and will, by assumption, find a $(h, \mathcal{P})$ strategy.

If there is no $(h, \mathcal{P})$ strategy in $\mathcal{R}$, then the algorithm will clearly return this fact. □

6.2.2 Generating Components

This algorithm generates the connected component of $\mathcal{R}$ containing a particular configuration d while iteratively checking if it contains an $(h, \mathcal{P})$ strategy. Our algorithm uses the fact that certain *troublemaker* vertices can be determined to not belong to any $(h, \mathcal{P})$ strategy. This allows us to ignore them and progressively prune the graph we are considering. We begin by defining these rigorously:

Fix h and $\mathcal{P}$ and let $\mathcal{R}$ be the $(h, \mathcal{P})$ reconfiguration graph, with r a subgraph of $\mathcal{R}$. A configuration $c \in V(r)$ is called a **troublemaker** if there is some attack $A \subseteq V(G)$ with $|A| = a$ where for each $d \in N_r(c)$ there is some $t \in A$ such that $|c(t)| = 0$ and $|d(t)| = 0$. In other words, c is a trouble maker if it cannot be part of a $(h, \mathcal{P})$ strategy which is a subgraph of r.

Note that this does not imply that troublemakers cannot be a part of a strategy at all, it may merely be the case r does not contain enough configurations to which c can reconfigure. However, if $N_{\mathcal{R}}(c) = N_r(c)$ we call c **useless**, and as the name implies, these configurations cannot be part of any $(h, \mathcal{P})$ strategy:

Lemma 6.3. *Let $\mathcal{R}$ be the $(h, \mathcal{P})$ reconfiguration graph for some h and $\mathcal{P}$. If c is some useless configuration in $\mathcal{R}$, then c cannot be part of any $(h, \mathcal{P})$ strategy.*

Proof. For the purposes of finding a contradiction let Λ be a minimal $(h, \mathcal{P})$ strategy which contains c. As Λ is minimal, then no matter which configuration Alice begins in, there is some sequence of attacks which results in her reconfiguring to c. But then c is useless, and so there exists a vertex $v \in G$ such that $|c(v)| = 0$ and for each $d \in N_{\mathcal{R}}(c)$ we have that $|d(v)| = 0$. Thus if Bob attacks v, Alice cannot defend and will lose the game. □

Let c be some configuration for which we wish to generate the component containing it. Let r_0 be a graph, initially with just the vertex c and no edges. Additionally, let T_0 and U_0 be our set of troublemakers and useless vertices respectively, both initially empty. Repeat the following for each iteration i:

- For each configuration in T_i, generate its neighbourhood N and add any edges and vertices in $N \setminus U_i$ which were not already present to r_i to form r_{i+1}.
- Now, check if there is an $(h, \mathcal{P})$ strategy in r_{i+1}. If there is, we return this strategy, or else we are returned a set of troublemakers we label as T_{i+1}.
- Update U_{i+1} to include all prior useless vertices as well as those in $T_i \cap T_{i+1}$
- Delete all vertices in $T_i \cap T_{i+1}$.

If we ever end up with an empty graph, we return that there is no $(h, \mathcal{P})$ strategy in this component. Pseudocode for this algorithm can be found in Algorithm 4. Note that the initial vertex we start has no impact on the output of the algorithm, and thus during our proof of its correctness we simply ignore this.

Theorem 6.4. *Algorithm 4 correctly determines if a connected component C of $\mathcal{R}$ contains a $(h, \mathcal{P})$ strategy.*

Proof. First note that the algorithm will check each configuration in C at most once. There are finitely many such configurations, and as such Algorithm 4 terminates.

First suppose that there exists $(h, \mathcal{P})$ strategy Λ in C. By definition, we only return false when we end up with the empty graph. However, we only remove vertices if they are determined to be useless, and these vertices cannot belong to Λ. Thus, we never delete any vertices of Λ, and will return true.

Now suppose that there does not exist a $(h, \mathcal{P})$ strategy in C. Again, we only delete vertices which we determine to be useless. Let $C_0 = C$ and D_0 be the set of vertices in C_0 which are useless. Recursively define $C_i = C_{i-1} \setminus D_{i-1}$ and D_i to be the useless vertices in C_i. As there are no $(h, \mathcal{P})$ strategies in C, each vertex belongs to some D_i, and as such is deleted in some iteration of the algorithm. Thus, we eventually end up with the empty graph and correctly return false. □

6.2.3 Pruning Troublemakers

This algorithm successively removes troublemakers from a subgraph r of the reconfiguration graph in order to find an $(h, \mathcal{P})$ strategy. The end result of this algorithm is always a graph with no troublemakers, as such if this graph is non-empty it is an $(h, \mathcal{P})$ strategy, and otherwise this subgraph does not contain such a strategy. If there is no strategy, then the initial set of troublemakers removed is returned for use in determining which configurations are useless.

To begin, for all configurations c we define two vectors (both indexed by the vertices of G), b_c and m_c. These vectors store the number of guards on a vertex, and the number of configurations in $N[c]$ which can defend attacks on a vertex respectively. We initialize these vectors as follows $b_c(v) = \min\{|c(v)|, 1\}$ and $m_c = {b_c}$[1][2]. We find our initial troublemakers by iterating over each edge $c_1c_2 \in E(r)$ and update m_{c_1} and m_{c_2} by adding b_{c_2} and b_{c_1} to them respectively. Our troublemakers are then the configurations for which there is some $v \in V(G)$ such that $m_d(v) = 0$; these can be found either by again checking each m_d after it has been computed or by keeping tracking of which vectors have a 0-element as they are computed.

Let T_0 be our initial set of troublemakers. In each iteration i of the pruning process we do the following steps: for each $t \in T_i$ delete t from r and then for each configuration $c \in N_r(t)$ update m_c as $m_c(v) = \max\{m_c(v) - b_t(v), 0\}$. If this produces a zero entry in m_c then we add c to T_{i+1}. We repeat this process until there are no troublemakers remaining. At this point if the graph is non-empty, we return it as our $(h, \mathcal{P})$ strategy and if not, we return T_0 as our initial set of troublemakers. Pseudocode for this algorithm can be found in Algorithm 5. We now prove its correctness:

[1] In reality, as we only care about when these entries are zero, it suffices to set $b_c(v) = |c(v)|$.

[2] Finally, a useful footnote.

Theorem 6.5. *Algorithm 5 correctly determines if a subgraph r of $\mathcal{R}$ contains an $(h, \mathcal{P})$ strategy.*

Proof. First observe that as we process each configuration at most once, this algorithm terminates. Additionally, we only delete configurations which are troublemakers. Observe that by deleting configurations we may cause other configurations to be considered troublemakers. Observe that in this case, these configurations also cannot have been part of a $(h, \mathcal{P})$ strategy, as their only neighbours defending some vertex were themselves troublemakers (and thus could not be part of a strategy). The correctness of the algorithm follows from these facts. □

6.2.4 Generating all Valid Configurations

This algorithm generates all h-configurations satisfying I_γ while placing at most s guards on a single vertex. These sorts of enumeration problems have been studied extensively before, and the algorithm presented here is based on the `EDS` algorithm shown in [20]. We will assume that the set of valid configurations forms an upset, which may not be true in all cases. This assumption can be changed (and these changes accounted for), but there are too many possibilities to survey here.

We employ a reverse search technique to generate configurations: begin with the trivial configuration i which places s guards on each vertex. Because the set of valid configurations is an upset, we know that any valid configuration must be a subconfiguration of i. Thus, we can generate all valid configurations by finding all valid subconfigurations of i.

We do this recursively, generating all subconfigurations of our initial configuration where a single guard has been removed. If any of these are valid and have strictly more than h guards, then we call the algorithm again with this new configuration as our initial configuration. If any are valid and have exactly h guards, then we return them. The fact that the described algorithm only returns valid configurations with exactly h guards is trivial, and so we need only show that all such configurations are eventually reached. Algorithm 6 does this, and we now show its correctness.

Theorem 6.6. *Algorithm 6 generates all valid subconfigurations of some initial configuration i.*

Proof. Suppose for contradiction that after running Algorithm 6 there are some subconfigurations which were not generated. Let c be the largest valid subconfiguration of i which is not generated. As the set of valid configurations is an upset, there is a valid superconfiguration of c, say d, which places exactly one more guard than c. By our assumption that c was the largest configuration not reached we know that d is reached. Thus when we consider removing the additional guard, we obtain c. Now the only way c is not returned is if it is not valid, which is a contradiction. □

While this method works, recall that our goal is to generate all configurations with h guards specifically. In many cases, the reverse search tree generated will be gargantuan, with the vast majority of these configurations having more than h guards. Therefore in practice we adopt a more complex method. Note that for any h-configuration c, there must be some c' which is a minimal subconfiguration of c (note that c may be minimal in which case $c' = c$). Let C be the set of all minimal valid configurations which have at most h guards. As the set of valid configurations is an upset, there is some P such that all configurations in C are subconfigurations of P. Note that we do not need the smallest such P; in practice this means that trivial choices of P exist which have far fewer guards than the initial configuration used above.

Given P we can adapt Algorithm 6 to find all minimal subconfigurations of h-configurations. Given a minimal configuration with $j \leq h$ guards, we can generate h-configurations by adding guards $h - j$ guards to them. In practice, we run `Enumerate-Valid-Subconfigurations` to find the minimal subconfigurations, and can then find all ways to add $h - j$ guards using other combinatorial algorithms. Note that for each value of $s = h - j$ we need only generate the ways to place these additional guards once, as they do not depend on the configuration considered.

6.2.5 Generating Neighbouring Configurations

In Algorithm 4 we construct the connected components of the $(h, \mathcal{P})$ reconfiguration graph by generating their neighbours directly. This is in opposition to the idea of generating all valid h-configurations and testing if they can reconfigure to each other, which is very slow in practice given the very large number of valid configurations. In this section we describe the algorithm we use to generate these neighbours.

Recall that Lemma 1.14 shows that the problem of reconfiguration is equivalent to finding a perfect matching in a bipartite graph. Construct a new bipartite graph $B = (V, E)$ a follows: begin by adding a vertex for each guard in c. For each guard g, let v_g be the vertex that guard is on, and then add copies of the vertices in $N_G(v_g)$ adjacent to g in B. By a similar proof to Lemma 1.14, a matching in this graph corresponds to a set of moves one can make with guards, and thus a set to which c can reconfigure.

Thus, matchings with at most m edges corresponds to moves lists with at most m moves. By generating all such matchings, we can generate all possible neighbouring configurations of guards. Some of these may be invalid, but we can simply check if the new configuration satisfies I_γ. The pseudocode which generates all such reconfigurations is shown in Algorithm 7; we now show its correctness.

Theorem 6.7. *Algorithm 7 finds all valid configurations to which some initial configuration c can reconfigure.*

Proof. As mentioned above, each possible reconfiguration from c can be represented as a matching in B. In these matchings, each edge corresponds to exactly one move. As we check all matchings with at most m edges, we will check all possible reconfigurations, and clearly only keep the ones which are valid. □

6.3 Running Times

In general, analyzing the running times of the algorithms given above is difficult. When applied to a specific instance, there often many optimizations which can be made and cannot be accounted for in any worst case analysis.

Focusing on Algorithm 2, note that on the worst case we make $O(\log n)$ calls to find $(h, \mathcal{P})$ strategies. Let $\texttt{hP}(h, \mathcal{P})$ denote the time it takes to find such a strategy. Then this algorithm runs in $O(\texttt{hP}(n, \mathcal{P}) \cdot \log n)$ time.

To find an $(h, \mathcal{P})$ strategy note that in the worst case we need to generate every valid configuration of guards. How many of these must be investigated depends heavily on any properties we impose on the configurations. When no properties must be satisfied, then there are n^h possible configurations.

Additionally, we must account for the fact that we regularly check if there is an $(h, \mathcal{P})$ strategy. It is difficult to say how many time such a check will be done, as this is determined by the existence of troublemaker configurations. Each such check potentially removes each vertex from the current subgraph of $\mathcal{R}$, again potentially checking every valid configuration. Thus for a fixed h, it takes $O(n^h)$ time to check if there is an $(h, \mathcal{P})$ strategy.

Thus, in the absolute worst case, Algorithm 2 runs in $O(n^n \log n)$ time. The author wishes to stress that in practice this algorithm is much faster for a few reasons:

- Oftentimes the true answer is much less than n, and as such we will never generate all configurations with n guards.
- Requiring that the configurations of guards satisfy an appropriate property (for example, be dominating sets to comply with Lemma 1.3) can massively reduce the number of configurations which need to be checked.
- Generating valid configurations can take significant additional time, but this can be done in advance of running the algorithm. This technique is particularly useful when investigating multiple similar instances on the same graph.

6.4 Applying the General Algorithm

In this last section we give example applications of Algorithm 2 to a few variants of eternal domination. By one count, there are at least 80 existing choices for I_γ, so

enumerating every possible application of this algorithm is impractical (See sections 3.8 and 3.9 of [10]). However, we aim to give sufficiently detailed examples in this section such that adapting this general purpose algorithm is not too difficult. In most cases, all that is needed is an algorithm for determining if a configuration is valid.

6.4.1 The Eternal Domination Number

We begin with simply computing the eternal domination number as presented in Chapter 1. The first definition given corresponds to the instance $\mathcal{P}_0 = (G, U, 1, \infty, \infty)$ where U is the universal set property that accepts all sets. This instance can be used in Algorithm 2 without additional modification.

Doing so however will be extremely slow as there are a large number of configurations to consider, many of which cannot be in any $(h, \mathcal{P}_0)$ strategy. To speed this process up, we can use Lemma 1.3, which tells us that we need only consider configurations which are also dominating sets. To do so requires us to have an algorithm which determines if a configuration is valid. Algorithm 8 does so.

Before we prove the correctness of Algorithm 8, note that this in fact can be applied to both the 1 guard moves model and the all guards move model, and so covers both of the classic models of eternal domination. This algorithm proceeds by checking whether guards are assigned to the endpoints of each edge; if there are, then both vertices are added to a set of defended vertices. If all vertices are considered defended by the time each edge has been checked, then we have a dominating set. Note that this algorithm only works for graphs without isolated vertices, but extending it to cover these is trivial.

Theorem 6.8. *For a given configuration c, Algorithm 8 determines if the guarded vertices form a dominating set.*

Proof. When considering a graph without isolated vertices, the definition of a dominating set can be restated as follows: a set D is a dominating set if for each vertex x there is a neighbouring vertex y such that at least one of x or y is in D. Algorithm 8 clearly tests for this property. □

6.4.2 The Eternal Roman Domination Number

Similar to the eternal domination number, the eternal Roman domination number simply needs an algorithm for determining if a configuration of guards is valid. Algorithm 9 does so, and functions similarly to Algorithm 8.

Theorem 6.9. *For a given configuration c, Algorithm 8 determines if the guarded vertices form a Roman dominating set.*

Proof. The proof is similar to that of Theorem 6.8. □

6.4.3 The Maneuver, Invasion, and Stacking Numbers

In addition to computing the number of guards needed for different instances $\mathcal{P}$, we can use Algorithm 2 to compute the three new parameters we have presented. We first determine the baseline number of guards needed, which is $D(G, I_\gamma, 1, \infty, \infty)$ in all cases. We can then search through the possible values of m, a, or s respectively and check if the number of guards needed has changed. By Theorem 2.7, Theorem 3.2, and Theorem 4.2 we can conduct each of these searches using binary search.

In practice it is faster to use a linear search for invasion numbers however, as we do not even know if there are even graphs with invasion number 3. Algorithms 10, 11, and 12 show pseudocode for computing the maneuver, invasion, and stacking numbers respectively.

Theorem 6.10. *Algorithms 10, 11, and 12 compute the maneuver, invasion, and stacking numbers respectively.*

Proof. Each of these proofs are similar, and each follows from the definitions of the parameters. □

6.5 Pseudocode

The pseudocode for each algorithm mentioned in this chapter can be found in this section.

Algorithm 2 An algorithm for determining $D(\mathcal{P})$ for any instance $\mathcal{P}$.

```
procedure ETERNAL-DOMINATION-NUMBER(P)
    Let d_min be the minimum value found.
    while binary search can continue do
        Let h be the next value to check from binary search.
        Run Find-Strategy(h, P).
        if an (h, P) strategy is found then
            Update d_min accordingly.
            Update binary search parameters accordingly.
        else if No strategy was found then
            Update binary search parameters accordingly.
        end if
    end while
    return d_min
end procedure
```

Algorithm 3 An algorithm for finding an $(h, \mathcal{P})$ strategy.

procedure FIND-STRATEGY(h, $\mathcal{P}$)
  Let S be the set of configurations who we have already seen.
  for each c found by `Enumerate-Valid-Configurations`$(h, \mathcal{P})$ **do**
    if $c \in S$ **then**
      Continue on to the next set.
    end if
    Run `Generate-Component`$(c, \mathcal{P})$.
    if an $(h, \mathcal{P})$ strategy is found **then**
      return the strategy found.
    end if
    Add all sets seen to S
  end for
  return that no strategy was found.
end procedure

Algorithm 4 An algorithm which generates the connected component of $\mathcal{R}_{\mathcal{P}}$ and checks if it contains an $(h, \mathcal{P})$ strategy.

procedure GENERATE-COMPONENT(i, $\mathcal{P}$)
  Let H be a graph with one vertex, i.
  Let $T = \{i\}$ be our set of troublemaker configurations.
  Let U be our set of useless configurations.
  Let S be the set of configurations we have already checked.
  while there are configurations to check in T **do**
    Remove some configuration d from T.
    Run `Find-Neighbours`$(d, \mathcal{P})$.
    Add all neighbouring vertices found to H.
    Add d to S.
    if T is empty **then**
      Run `Prune-Troublemakers`$(H, \mathcal{P})$
      if An $(h, \mathcal{P})$ strategy is found **then**
        Return the strategy found and S.
      end if
      Let T' be the initial set of troublemakers found while pruning.
      Remove all configurations in $S \cap T'$ from H and add them to U.
      Add all configurations in $T' \setminus U$ to T.
    end if
  end while
  return that there is no $(h, \mathcal{P})$ strategy in this component.
end procedure

Algorithm 5 An algorithm which repeatedly prunes troublemaker configurations in order to find an $(h, \mathcal{P})$ strategy.

procedure PRUNE-TROUBLEMAKERS(r, $\mathcal{P}$)
  for each configuration c in r **do**
    Generate b_c and m_c.
  end for
  Let T_0 be our initial set of trouble make configurations.
  let Q be a queue initialized to contain each configuration in T_0.
  while there are configurations in Q **do**
    Let c be some configuration popped off of Q.
    Delete c from r.
    for each neighbouring configuration $c' \in N(c)$ **do**
      Let $m_{c'} = m_{c'} - b_c$.
      if $m_{c'}$ now contains a zero element **then**
        Add c' to Q.
      end if
    end for
  end while
  if r is non-empty **then**
    return that there is an $(h, \mathcal{P})$ strategy (namely, the remaining graph).
  else if r is empty **then**
    return that there is no $(h, \mathcal{P})$ strategy in this component.
  end if
end procedure

Algorithm 6 An algorithm which enumerates all valid subconfigurations of some configuration i.

procedure ENUMERATE-VALID-SUBCONFIGURATIONS(i, $\mathcal{P}$)
  for each vertex $v \in V(G)$ **do**
    Let i_v be the subconfiguration where a guard has been removed from v.
    if i_v is a valid configuration **then**
      Run `Enumerate-Valid-Subconfigurations`$(i_v, \mathcal{P})$
      Yield i_v
    end if
  end for
end procedure

Algorithm 7 An algorithm which enumerates all configurations neighbouring an initial configuration c.

```
procedure FIND-NEIGHBOURS(c, 𝒫)
    Let B be the bipartite graph defined above.
    for each matching M in B do
        if |M| ≤ m then
            Let c′ be obtained by moving the guards according to M.
            if c′ satisfies I_γ then
                Yield c′
            end if
        end if
    end for
end procedure
```

Algorithm 8 An algorithm for checking if the vertices guarded by a configuration c form a dominating set on the graph G.

```
procedure RECOGNIZE-DOMINATING-CONFIGURATION(c, 𝒫)
    Let D = ∅ be the set of defended vertices.
    for each edge uv ∈ E(G) do
        if |c(u)| ≥ 1 or |c(v)| ≥ 1 then
            Add u and v to D.
        end if
    end for
    if |D| = |V(G)| then
        return true
    end if
    return false
end procedure
```

Algorithm 9 An algorithm for recognizing if the guards in a configuration c form a Roman dominating set on the graph G.

procedure RECOGNIZE-ROMAN-DOMINATING-CONFIGURATION(c, $\mathcal{P}$)
  Let $D = \emptyset$ be the set of defended vertices.
  for each edge $uv \in E(G)$ **do**
    if $|c(u)| \geq 2$ or $|c(v)| \geq 2$ **then**
      Add u and v to D.
    end if
    if $|c(u)| = 1$ **then**
      Add u to D.
    end if
    if $|c(v)| = 1$ **then**
      Add v to D.
    end if
  end for
  if $|D| = |V(G)|$ **then**
    return true
  end if
  return false
end procedure

Algorithm 10 An algorithm for calculating the maneuver number $M(G)$ of a graph

procedure MANEUVER-NUMBER($\mathcal{P}$)
  Let $m_{min} = \infty$ be the minimum value for m found so far.
  Compute $D_M(G; \infty)$ by running `Eternal-Domination`$(G, I_\gamma, 1, \infty, \infty)$.
  Let $m_l = 1$ and $m_u = D_M(G; \infty)$.
  while $M(G)$ has not been determined **do**
    Let m be the next number to check as determined by binary search.
    Compute $D_M(G; m)$ by running `Eternal-Domination`$(G, I_\gamma, 1, m, \infty)$.
    Update the parameters of the binary search appropriately.
    if $D_M(G; m) = D_M(G; \infty)$ and $m < m_{min}$ **then**
      Update m_{min} to m.
    end if
  end while
  return m_{min}.
end procedure

Algorithm 11 An algorithm for calculating the invasion number $I(G)$ of a graph

procedure INVASION-NUMBER($\mathcal{P}$)
  Let a_{max} be the maximum value for a found so far.
  Compute $D_A(G;\infty)$ by running `Eternal-Domination`$(G, I_\gamma, 1, \infty, \infty)$.
  Let $a = 2$.
  while $D_A(G;a) \neq D_A(G;1)$ **do**
    Compute $D_A(G;a)$ by running `Eternal-Domination`$(G, I_\gamma, a, \infty, \infty)$.
    Let $a = a + 1$.
    if $D_A(G;a) = D_A(G;1)$ **then** Let $a_{max} = a$.
    end if
  end while
  return a_{max}.
end procedure

Algorithm 12 An algorithm for calculating the stacking number $S(G)$ of a graph

procedure STACKING-NUMBER($\mathcal{P}$)
  Let $s_{min} = \infty$ be the minimum value for m found so far.
  Compute $D_S(G;\infty)$ by running `Eternal-Domination`$(G, I_\gamma, 1, \infty, \infty)$.
  Let $s_l = 1$ and $s_u = D_M(G;\infty)$.
  while $S(G)$ has not been determined **do**
    Let s be the next number to check as determined by binary search.
    Compute $D_S(G;s)$ by running `Eternal-Domination`$(G, I_\gamma, 1, \infty, s)$.
    Update the parameters of the binary search appropriately.
    if $D_S(G;s) = D_S(G;\infty)$ and $s < s_{min}$ **then**
      Update s_{min} to s.
    end if
  end while
  return s_{min}.
end procedure

Chapter 7
Conclusion & Open Questions

We end the thesis by presenting a range of open problems related to the eternal domination problem, as well as the variants we have introduced.

7.1 Maneuver Numbers

We begin by presenting a handful of problems related to maneuver numbers and other topics related to reconfiguration. Aside from the questions below, one can of course ask about the maneuver number for specific classes of graphs:

Problem 7.1. Let $\mathcal{C}$ be a class of graphs, and let $G \in \mathcal{C}$ be some arbitrary graph in $\mathcal{C}$. Do there exist bounds on $M(G)$? Is it possible to determine the maneuver number exactly in polynomial time?

7.1.1 Maneuver Loading

We will call a graph **maneuver loaded** if there exists k such that

$$D(G; k) = k = D(G; \infty)$$

Or in other words, a graph is maneuver loaded if the optimal strategy for defending the graph moves *all* guards simultaneously in some state transition. It is not hard to see that a few classes of graphs are maneuver loaded. However, it is very likely that this is not a complete list.

Theorem 7.2. *Let G be a graph. Then G is maneuver loaded if it belongs to any of the following classes of graphs:*

- *Cycles*
- *Complete multipartite graphs.*

- *Any tree where an optimal locally finest neocolonization contains only a single block.*

Proof. The result is obvious for cycles and complete multipartite graphs, and follows immediately from Theorem 2.18 for the third case. □

Problem 7.3. Which classes of graphs are maneuver loaded?

7.1.2 Lower Maneuver Numbers

We defined the maneuver number as the minimum value of k such that $D_M(G;k) = D(G;\infty)$, i.e the minimum number of guards allowed to move so that the total number of guards needed is minimum. At the other end of this spectrum, we can ask what the maximum number of guards we can allow to move while still not being able to use fewer guards. In other words, define the **lower maneuver number** as the maximum value of m such that $D_M(G;m) = D_M(G;1)$. We denote this as $M^-(G)$.

Given the wealth of literature on the one guard moves model and the lower number of possibilities to consider, this problem may be more tractable than the standard maneuver number. The most obvious problem for this problem is simply determining the lower maneuver number for some classes of graphs:

Problem 7.4. Let $\mathcal{C}$ be a class of graphs and $G \in \mathcal{C}$ an arbitrary graph in $\mathcal{C}$. Is it possible to bound $M^-(G)$? Can this number be determined in polynomial time?

7.1.3 Required Moves

Let G be a graph and $m \leq D_M(G;\infty)$ some integer. Consider the variant of the eternal domination problem where we require that *at least* m guards are moved in each state transition. We call this the m **required moves model**, and denote the minimum number of guards needed for a winning strategy with $D_{M^-}(G;m)$. We can then define another parameter analogous with the maneuver number, called the **shifting number**, as the maximum value of m such that $D_{M^-}(G;m) = D_M(G;\infty)$.

We note that it it is most interesting to prevent guards from swapping positions in this model, as these move allow one to move many guards without potentially disrupting an arrangement. In fact, in the case where swaps are banned, observe that cliques require more guards to be defended:

Theorem 7.5. $D_{M^-}(K_n;1) = 2 > D_M(G;\infty)$

Proof. It is clear that we can defend K_n with 2 guards when 1 guard is required to move each round. Let v be a vertex of K_n and suppose for contradiction that we can defend $G = K_n$ with just one guard in this model. Consider attacking v twice. After the first attack, the guard must be present on v. After the second, the guard must have left v, but this leaves it undefended. Clearly 2 guards suffice to defend. □

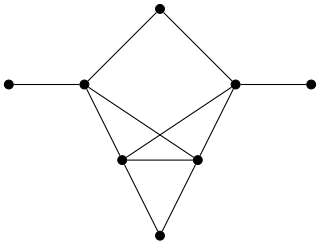


Figure 7.1: An example of a graph with invasion number 2 which does not belong to a class we have characterized.

Problem 7.6. Are there graphs for which $D_{M^-}(G;m) - D_M(G;\infty)$ is arbitrarily large for small values of m? How does this change if swaps are allowed or disallowed?

Problem 7.7. What are the shifting numbers for some classes of graphs? When can they be computed in polynomial time?

7.2 Invasion Numbers

In this section we present a open questions related to multiple attacks and invasion numbers. The two most pressing questions to the author are whether or not there are any graphs with invasion number 3, and whether or not there is a general upper bound on the number of guards needed to defend against a attacks.

Problem 7.8. Are there are any graphs with invasion number 3 (or higher)?

Problem 7.9. Is it true that for any a and G, we have that $D_A(G;a) \leq \left\lceil \frac{a}{a+1}|V(G)| \right\rceil$?

Additionally, we have only considered the all guards move model for a-attacks. In analogy to the 1 guard moves model, one can consider the a guards move model for a attacks. Let $D_{AM}(G;a,b)$ denote the minimum number number of guards needed when a vertices are attacked in each round, and at most b guards are allowed to move to defend.

Problem 7.10. What are some bounds for $D_{AM}(G;a,a)$?

In Chapter 3 we presented a number of classes of graphs with invasion number 2. As may be suspected, this was not a complete characterization, as the graph in Figure 7.1 has invasion number 2 and does not belong to any class we have characterized.

Problem 7.11. Which other classes of graphs have invasion number 2?

Problem 7.12. Which graphs have invasion number 2 and are minimal with respect to edge deletions?

Problem 7.13. Conversely, which invasion number 2 graphs are maximal with respect to adding edges?

Corollary 3.14 shows that there is no forbidden subgraph characterization for invasion number 2 graphs. With some difficulty, we were able to show that there are still forbidden structures for graphs with high invasion number. Can these results be extended?

Problem 7.14. Are there forbidden structures for graphs with a particular invasion number?

Finally, we showed that there were two classes of graphs constructed through particular operations which either increased or preserved the invasion number. Do other such operations exist?

Problem 7.15. Is there an operation on graphs which preserves invasion number?

Problem 7.16. Is there an operation on graphs which increases the invasion number?

7.2.1 Upper Invasion Numbers

Let G be a graph and define the **upper invasion number** as the maximum value of k such that $D_A(G; n-k) = n-k$, denoted as $I^+(G)$. Observe that by Lemma 3.5, every graph has upper invasion number at least 1.

Problem 7.17. Which classes of graphs have upper invasion number greater than 1? Can this value be computed in polynomial time?

7.2.2 Higher Tolerance

Let the **tolerance-c invasion number**, $I_c(G)$, be the maximum value of a such that $D(G; a) \leq D(G; 1) + c$. The standard invasion number is then the tolerance-0 invasion number, but this is clearly a very tight condition. By relaxing c, we can obtain a wide range of open questions:

Problem 7.18. For a fixed value of c, which classes of graphs have high tolerance-c invasion number?

Problem 7.19. For a fixed value of c, what is the maximum tolerance-c invasion number? Which graphs achieve this number?

Problem 7.20. Is there a value of c such that the proportion of graphs with $I_c(G) \geq 2$ is positive? How small can this c be, and in particular, is it a constant?

Problem 7.21. Let $f \in \omega(1)$ be some function which tends to infinity. Is there some c such that every graph on n vertices has tolerance-c invasion number at least $f(n)$?

7.3 Stacking Numbers

In Chapter 4 we presented the stacking number, answering an open question of Finbow et al. [9]. We did so by constructing a specific class of graphs with the desired properties. However, we have left open many of the more general results regarding this parameter. Of course the most natural question to ask is whether there are other constructions which yield graphs with arbitrary stacking number, and what the stacking number of other classes of graphs is:

Problem 7.22. Find other constructions which give graphs G with arbitrarily high values for $S(G)$?

Problem 7.23. Can the stacking number be bounded for classes of graphs other than trees?

7.3.1 Stacking Proportions

Let G be a graph and v a vertex. We call v **s-stacked** if for every optimal eternal domination strategy Λ, there is some state which places at least s guards on v. For a fixed s let $\sigma_s(G)$ denote the set of s stacked vertices in G. Call the proportion $\frac{\sigma_s(G)}{n}$ the s **stacking proportion** and let $S^+(s)$ denote the maximum s stacking proportion when considered over all graphs. Our definition of the airport graph gives us a lower bound on this value:

Corollary 7.24. *For any s we have that $S^+(s) \geq \frac{1}{2s^2+s+1}$.*

Proof. Let $G = G(k, (s), (2s+1))$ and note that G has the following number of vertices:

$$(2s+1)(s(k+1)+1)+k = k(2s^2+s+1)+(2s+1)^2,$$

The stacking number of G is s, and k vertices are required have s guards stacked on them in any optimal strategy. Therefore we get that

$$S^+(s) \geq \frac{k}{k(2s^2+s+1)+(2s+1)^2}$$

If we take the limit as k goes to infinity the result follows. □

Problem 7.25. Is it the case that $S^+(s) \geq f(s)$ for some $f \in \omega(\frac{1}{s^2})$?

Problem 7.26. Are there graphs G and integers $s \geq 2$ for which $\sigma_s(G) = n$? Note that this would imply that $S^+(s) = 1$.

7.4 Eternal Roman Domination

In Chapter 5 we presented an upper bound for the eternal Roman domination number on trees and showed that it was best possible. However, that was but one variant of eternal Roman domination (known as free eternal Roman domination). In this variant, guards can move freely, and in particular can travel away from vertices in different directions.

Problem 7.27. Let $\mathcal{C}$ be a class of graphs and G an arbitrary graph in that class. Are there bounds on $D_R(G)$? Can it be computed exactly in polynomial time?

Another variant of eternal Roman domination which has been investigated is that of **strong eternal Roman domination** [13]. In this variant, guards themselves have a strength of 1 or 2, and a guard cannot be split. We use $D_{SR}(G)$ to denote the minimum weight of a winning strategy. There is an easy upper bound to this value:

Theorem 7.28. *Let G be a graph. Then $D_{SR}(G) \leq 2D(G)$.*

Proof. This follows directly from replacing each guard in an eternal dominating strategy with one of strength 2 to obtain an eternal Roman dominating strategy. □

Despite the simplicity of this bound, it may be best possible in the case of trees. This would imply that there is no non-trivial upper bound on $D_{SR}(G)$ for all graphs, but such a bound may still exist for other classes of graphs:

Problem 7.29. Does there exist a tree T with $D_{SR}(T) < 2D(T)$?

Problem 7.30. Let $\mathcal{C}$ be a class of graphs and G an arbitrary graph in that class. Do there exist bounds on $D_{SR}(G)$? Can it be computed exactly in polynomial time?

Problem 7.31. Let G be a graph. Are there characterizations for when $D_{SR}(G) < 2D(G)$?

7.5 Algorithms

In Chapter 6 we presented a general algorithm for computing the eternal domination number as well as ones for the maneuver, invasion, and stacking numbers. These algorithms run in exponential time. There are faster algorithms known for specific classes of graphs, such as trees and split graphs [15, 1]. In the one guard moves model, the computational complexity is known to be NP-complete[24]. However for general instances $\mathcal{P}$, the computational complexity of the eternal domination problem is not known. It is not even clear that this problem belongs to NP, as any strategy found by an algorithm may have exponential size.

Problem 7.32. What is the computational complexity of the eternal domination problem, for an arbitrary instance $\mathcal{P}$?

Problem 7.33. Are there faster algorithms than ours for computing the eternal domination number for a general instance of the problem? We are sure there are, but what are they?

Our algorithm relied on the notion of useless configurations, those configurations which could not be part of any eternal dominating strategy. Characterizing these configurations would be useful not just in potentially improving our algorithm, but also in proving results directly.

Problem 7.34. Are there characterizations of situations where a configuration is useless?

7.5.1 Further Applications of the General Algorithm

Throughout this chapter we present many new parameters and problems related to the eternal domination number. It bears mentioning that anyone who wishes to investigate these parameters can be compute the value of these parameters with simple modifications to Algorithm 2 (or the other algorithms which it calls). We do not present these algorithms, and leave these modifications as an exercise for the intrepid researcher.

7.6 Sequences

Analogous to our definition of the stacking sequence in Chapter 4, we can define the *maneuver sequence* and *invasion sequence* Specifically, let G be a graph and define the **maneuver sequence** (m_i) as $m_i = D_M(G; i)$ and the **invasion sequence** (a_i) as $a_i = D_A(G; i)$.

Using these sequence we can redefine the maneuver number as the last index where the value changes, and the invasion number as the first index where this happens. By Theorems 2.7 and 3.2 the maneuver sequence is decreasing, and the invasion sequence is increasing. The most obvious open problems for these sequences is determining which sequences are achievable, and by which graphs? Note that Theorem 4.5 provides a description of graphs achieving some stacking sequences, but we do not believe that this description is complete.

Problem 7.35. For a sequence M, is there a graph G which has M as its maneuver sequence?

Problem 7.36. For a sequence A, is there a graph G which has A as its invasion sequence?

Problem 7.37. For a sequence S, is there a graph G which has S as its stacking sequence?

7.7 Other Variants

We finish with an assortment of other open problems and variants of eternal domination which are not as closely related to the ones we have investigated ourselves.

7.7.1 Reconfiguration Variants

In all of the variants presented so far, we have dealt with the notion of reconfiguration where guards move to only adjacent vertices. This is closely related to the notion of Token Sliding in more standard reconfiguration problems. There are however, two other more commonly investigated variants of reconfiguration: *token addition/removal* and *token jumping*.

Under **token addition/removal** rules instead of moving guards to adjacent vertices, the player can choose to *either* add or remove them from a vertex. We use $D_{TAR}(G;k)$ to denote the maximum weight of a state in an eternal dominating strategy where at most k guards can be added or removed. It is not hard to see that this variant is fairly trivial:

Theorem 7.38. *For any graph G and integer k, $D_{TAR}(G;k) = n$.*

Proof. One can force the defender to place a guard on every vertex by successively attacking vertices without guards on them. The defender must always choose to add a guard, and this guard must be on the attacked vertex. The result follows. □

Under **token jumping** rules, instead of moving to adjacent vertices, guards are allowed to move to any vertex. Let $D_{TJ}(G;k)$ denote the minimum number of guards needed to eternally dominate G where at most k guards are allowed to jump in each round. We can restrict the value of $D_{TJ}(G;k)$ to two values:

Theorem 7.39. *For any graph G and integer k we have that*

$$\gamma(G) \leq D_{TJ}(G;k) \leq \gamma(G) + 1$$

Proof. The lower bound is trivial. A strategy for the upper bound can be constructed by selecting some minimum dominating set D of G, placing guards on each vertex of D, and then having one extra guard which jumps to each attacked vertex. □

When we allow all guards to jump between rounds, then there is a simple dichotomy theorem to determine if $D_{TJ}(G;\infty) = \gamma(G)$:

Theorem 7.40. *Let G be a graph. Then $D_{TJ}(G;\infty) = \gamma(G)$ if and only if for each vertex $x \in V(G)$ there is a minimum dominating set D containing x.*

Proof. Label the vertices of G with $[n]$ and let $D_1, \ldots, D_n$ be the minimum dominating sets containing each vertex i respectively. If such a set exists, then clearly whenever some vertex i one can reconfigure to D_i to defend the attack. This is an eternal dominating strategy, and so $D_{TJ}(G;\infty) = \gamma(G)$ in this case.

If $D_{TJ}(G;\infty) = \gamma(G)$ then there exists an eternal dominating strategy for G which uses $\gamma(G)$ guards. The states of this strategy then must necessarily be a set of minimum dominating sets where each vertex v is contained in some set. □

Theorem 7.41. *Let G be a graph and x a vertex of G which is adjacent to at least 2 leaves. Then $D_{TJ}(G;\infty) = \gamma(G) + 1$.*

Proof. Suppose for contradiction that there exists a graph G with vertex x such that x is adjacent to at least 2 leaves but $D_{TJ}(G;\infty) = \gamma(G)$. Let L be the set of leaves adjacent to x and let $l \in L$ be some leaf in L. By Theorem 7.40 there is a minimum dominating set D containing l. If $x \in D$, then $D' = D \setminus \{l\}$ is a smaller dominating set, which is a contradiction. So assume $x \notin D$. This implies that each leaf adjacent to x is in D, as otherwise we would not have a dominating set. But then $D \setminus L \cup \{x\}$ is yet another smaller dominating set, which is again a contradiction. □

Additionally, note that it is possible to define analogs of the maneuver and invasion number for the token jumping model; in the case of a attacks, the upper bound in Theorem 7.39 fails but can be replaced with $\gamma + a$. We leave these problems open:

Problem 7.42. For which graphs G and integers k do we have $D_{TJ}(G;k) = \gamma(G)$?

Problem 7.43. Given a graph which has $D_{TJ}(G;\infty) = \gamma(G)$, what is the minimum value of k such that $D_{TJ}(G;k) = \gamma(G)$?

Problem 7.44. For a graph G what is the maximum number of simultaneous attacks which can be defended against under the token jumping model without needing additional guards?

7.7.2 Finite Defense

Requiring that a strategy can defend against attacks forever is a very strong condition. In many cases it may make sense to relax this notion to only require survival against a finite sequence of attacks. We refer to the minimum number of guards needed to defend against a sequence of k attacks as the **finite eternal domination number**, denoted $D_{FIN}(G;k)$. The following bounds are trivial:

Theorem 7.45. *For any graph G and integer k we have $\gamma(G) \leq D_{FIN}(G;k) \leq D(G)$.*

Proof. Clearly we still must maintain a dominating set in all steps (except possibly the last), as otherwise there is a vertex which could not be defended. Additionally, an eternal dominating set is obviously sufficient to defend against any finite sequence of attacks. □

Note that for graphs which have $\gamma(G) = D(G)$ we have that $D_{FIN}(G)$ is completely determined. A few cases of this problem have already been investigated: when $k = 1$, the problem is equivalent to the standard dominating set problem; then case for $k = 2$ has previously been investigated under the name of "secure domination"[6, 8].

There are a number of obvious open problems for this variant:

Problem 7.46. Let $\mathcal{C}$ be a class of graphs and G an arbitrary graph from $\mathcal{C}$. For some value of k, are there bounds on $D_{FIN}(G; k)$? Can the exact value be computed in polynomial time?

Problem 7.47. Let G be a graph. Is there an integer k for which $D_{FIN}(G; k) = D(G)$? How small can this value of k be?

7.7.3 Offline Defense

The eternal domination problem when defined as a game is an online problem, and one in which the input is adversarial. We now present a variant of the game which is offline, i.e. the input is given fully ahead of time. Let G be a graph and S some (possibly infinite) sequence of vertices of G. We use $D_{OFF}(G; S)$ to denote the minimum number of guards needed to protect against attacks on the vertices in S.

This problem has been studied under the name of *secure dominating sets*, particularly in the case $k = 1$ [16]. The variation presented here differs however, as the proof of Lemma 1.3 fails for the offline variant. As such we note that it is not necessary to always maintain a dominating set in this offline variant. We first present two simple upper bounds which apply for any sequence of attacks:

Theorem 7.48. *For any graph G and sequence S, we have that $D_{OFF}(G; S) \leq D(G)$.*

Theorem 7.49. *Let G be a graph and S a sequence of vertices. If S can be partitioned into k walks in G then $D_{OFF}(G) \leq k$.*

Proof. Let $W_1, \ldots, W_k$ be such a partition. By placing one guard on each walk and moving it along when the next vertex is attacked, we have found an offline defense strategy. □

By Theorem 7.49 it is clear that some sequence of attacks are trivially handled. We let $P(G)$ denote the maximum value of $D_{OFF}(G; S)$ taken over all sequences S, and call this the **preparation number** of G. We call a sequence which achieves this value a **disaster sequence**. Note that by Theorem 7.48 we have that $P(G) \leq D(G)$.

The **burning number** of a graph is the minimum number k such that a graph can be covered by balls of radius $1, \ldots, k$ such that the each ball does not contain the centre of a ball of higher radius (more commonly this number is phrased in terms of a game). This number is denoted as $b(G)$ and was introduced by Bonato et al. [3]. We can use this parameter to lower bound the value of the preparation number:

Theorem 7.50. *For any graph G, $b(g) \leq P(G)$.*

Proof. Let $v_1, \ldots, v_k$ be the centres of the balls of radius $1, \ldots, k$ respectively. Consider the sequence of attacks $S = (v_k, \ldots, v_1)$. After the j-th round, we have that for any guard on some vertex v_t can have travelled at most $j - t$ vertices away from where it began. By the definition of the burning number we know that $d(v_i, v_j) > j - i$, and so we must have needed an additional guard to defend against this attack. The result follows. □

We end this section with some open problems.

Problem 7.51. Let G be a graph. What do the disaster sequences of G look like?

Problem 7.52. Let G be a graph. What is the preparation number of G?

Problem 7.53. Let G be a graph. Can the preparation number of G be bounded based on the structure of G?

Problem 7.54. Do there exist graphs for which $P(G) > b(G)$? Are there any for which $P(G) = D(G)$?

7.7.4 Directed Graphs

We add this section to note that it is very easy to define the eternal domination number for directed graphs. We naturally assume that guards must move in the direction of arcs. As with eternal domination we by default assume that there are no restrictions on the configurations formed, but note that by a similar proof to Lemma 1.3 we must have that for each unguarded vertex, there must be an in-neighbour with a guard. If not then an attack on such a vertex cannot be defended.

We use $D_{DIR}(G)$ to denote the minimum number of guards needed to defend G eternally in this variant. Additionally, if we let G be an undirected graph, we can consider orienting its edges so as to maximize $D_{DIR}(G)$. Accordingly, let $C(G)$ be the **compass number** of G, the maximum value of $D_{DIR}(G)$ when taken over all orientations of G.

We present a simple lower bound on $D_{DIR}(G)$ before some open problems:

Theorem 7.55. *Let G be a directed graph, and let k be the number of source vertices in G. Then $D_{DIR}(G) > k$.*

Proof. Note that it is impossible to move a guard to a source vertex, and so we must begin with a guard on each. The fact that no two sources can be adjacent to each other and these guards cannot leave their vertices gives us the strictness of this bound. □

Problem 7.56. Let $\mathcal{C}$ be a class of directed graphs and $G \in \mathcal{C}$ an arbitrary directed graph from this class. Are there bounds on $D_{DIR}(G)$? Can this number be computed in polynomial time?

Problem 7.57. Let G be a directed graph. What is the compass number of G, and which orientations achieve it? Can these orientations be characterized?

7.7.5 The Base Set

Throughout this thesis we presented the maneuver, invasion, and stacking numbers of a graph. Each of these relied on tweaking a particular assumption made about the rules of the eternal domination game while leaving the others in their default state. However, it is natural to ask how these new parameters can depend on each other when other assumptions are not at their default state. This creates a whole host of (admittedly somewhat arbitrary) new parameters for a graph.

To try and wrangle this mess, we present the concept of the *base set* in relation to the eternal domination problem. Let $D(G; a, m, s)$ denote the minimum number of guards needed to defend against G when there are a simultaneous attacks, at most m guards can move, and at most s guards can occupy the same vertex. The **base set** is then the set of all triples (a, m, s) such that $D(G : a, m, s,) = D(G; 1, \infty, \infty)$.[1]

Problem 7.58. For a given graph G, what does the base set of G look like?

[1] Thanks for reading this far. Now go touch grass.

Glossary

adjacent blocks of a neocolonization Let $N = (V_1, \ldots, V_k)$ be a neocolonization of a graph G. The blocks V_i and V_j are said to be adjacent if there are vertices $v \in V_i$ and $u \in V_j$ such that $uv \in E(G)$.

bipartite movement graph The bipartite movement graph for two configurations c_1 and c_2 is a graph which encodes whether or not c_1 can reconfigure to c_2.

block adjacency graph Let $N = (V_1, \ldots, V_k)$ be a neocolonization of a graph G. The block-adjacency graph of G with respect to N is the graph with vertices for each block of N and edges between adjacent blocks.

clique covering A clique covering of a graph is a partition of its vertices so that each part induces a complete graph.

clique covering number The clique covering number of a graph is the minimum size of a clique covering, denoted $\theta(G)$.

clique expansion An operation applied to a clique C which adds 2 vertices: one, x, adjacent to every vertex in C, and another, l, adjacent to only x. Denoted $CE(G, C)$.

closed neighbourhood The set of vertices adjacent to a vertex or set as well the vertex or set itself.

configuration property A property which every configuration of guards in the eternal domination game must satisfy. Usually stated in terms of the set of vertices occupied by guards.

connected dominating set A connected dominating set is a dominating set whose vertices induce a connected graph.

connected domination number The minimum size of a connected dominating set on a graph G, denoted $\gamma_c(G)$.

decoy vertices Vertices added during a reduction in order to force particular configurations of guards to appear in the reduced graph.

derived strategy Let T be a tree with neocolonization N. The strategy derived from N, Λ_N, is an eternal dominating strategy with weight $w(N)$ which has $m(\Lambda_N) = \max_{B_i \in N} diam(T[B_i])$..

distinguished corona An operation applied to a graph relative to a vertex x. Adds a leaf adjacent to every vertex other than x. Denoted $DC(G, x)$.

dominating set A set S of vertices is called a dominating set if every vertex is either in S or adjacent to a vertex in S.

domination number The minimum size of a dominating set on a graph G, denoted $\gamma(G)$.

eternal dominating strategy Any winning strategy for Alice (the defender) in the eternal domination game is referred to as an eternal dominating strategy.

eternal domination game A two player game where guards are moved around the vertices of a graph to defend against attacks while maintaining a dominating set.

eternal domination number The minimum number of guards needed for Alice (the defender) to win the eternal domination game. Denoted $D(G)$.

eternal Roman domination number The eternal Roman domination number of a graph is the least k such that there is an eternal Roman dominating strategy where all states have weight k.

instance An instance of the eternal domination problem is a 5-tuple (G, I_γ, a, m, s) where G is a graph; I_γ is a property each configuration must have; and a, m, and s are integers corresponding to the number of attacks, guards allowed to move, and guards allowed to occupy a vertex respectively.

invasion number The invasion number of a graph is the maximum number of simultaneous attacks which can be eternally defended against without requiring additional guards. In other words, it is the maximum value of k for which $D_A(G; k) = D_A(G; 1)$.

$(k, \mathcal{P})$ **reconfiguration graph** The graph whose vertices consist of all k configuration satisfying the constraints of $\mathcal{P}$, with edges between configurations which can reconfigure to each other.

$(k, \mathcal{P})$ **strategy** A subgraph of the $(k, \mathcal{P})$ reconfiguration graph which corresponds to a winning strategy for Alice in the eternal domination game.

k**-configuration** A k configuration is an assignment of k mobile guards to the vertices of a graph G, or formally, is a function $c : [k] \to \mathcal{P}(V(G))$ such that $\{c(v) : v \in V(G)\}$ is a partition of $V(G)$.

locally finest neocolonization A neocolonization is said to be locally finest if it cannot be refined.

maneuver load The maneuver load of an eternal dominating strategy Λ is the maximum number of guards which move in some transition between states. Denoted $m(\Lambda)$.

maneuver number The maneuver number of a graph is the minimum maneuver load over all optimal strategies of G, denoted $M(G)$.

maneuver optimal A strategy Λ for a graph G is said to be maneuver optimal if $m(\Lambda) = M(G)$.

moderate vertex Let G be a tree with leaves L and let $G' = G - L$. Then x is said to be moderate if $diam(G' - x) = diam(G')$..

movement digraph The movement digraph of a graph G with respect to some moves list M is a directed multigraph where each arc corresponds to a move in M.

moves list A moves list for a configuration c on a graph G is a list of pairs (u, v) where $u, v \in V(G)$, and for each $x \in V(G)$ we have that x appears at most $c(x)$ times as the first coordinate.

neocolonization A neocolonization of a graph is a partition of the vertices of the graph such that each part induces a connected graph. The sets in this partition are referred to as the blocks of the neocolonization.

neocolonization refinement A neocolonization N can be refined if there exists another neocolonization N' such that $w(N') \leq w(N)$ and each block of N' is a subset of a block in N.

open neighbourhood The set of vertices adjacent to a vertex or set, not including the vertex or set itself.

R1 One of the reductions we can apply to a tree to form a neocolonization. Consists of deleting the leaves adjacent to a vertex which is adjacent to exactly one internal vertex, and adding one guard.

R2 One of the reductions we can apply to a tree to form a neocolonization. Consists of deleting a vertex of degree 2 which is adjacent to exactly one leaf and adding one guard.

reconfiguration of k configurations Two k configuration c_1 and c_2 are said to be able to reconfigure to one another if it is possible to move the guards in c_1 to adjacent vertices and form c_2..

Roman dominating configuration A roman dominating configuration is a k configuration whose guards form a Roman dominating set.

Roman dominating function Let $G = (V, E)$ be a graph and $r : V \to \{0, 1, 2\}$ be some function. We say that r is a Roman dominating function if for each vertex u such that $r(u) = 0$ there exists a vertex v adjacent to u such that $r(v) = 2$. In other words, r is a Roman dominating function if every vertex which is assigned 0 is adjacent to one which is assigned 2.

Roman domination number The Roman domination number of a graph G is the minimum weight of a Roman dominating set of G, denoted $\gamma_R(G)$.

stacking load For a graph G with eternal dominating strategy Λ the stacking load of Λ is the maximum number of guards which occupy a vertex in any state of Λ.

stacking number The stacking number of a graph is the minimum k such that there exists an optimal eternal dominating strategy where at most k guards are allowed to occupy a vertex.

stacking sequence The stacking sequence of a graph is sequence defined as $D_S(G; i)$ for $i \geq 1$.

subconfiguration A configuration c' is a subconfiguration of another configuration c if for all v we have that $c'(v) \subseteq c(v)$.

subtree rooted at x with respect to y For a tree T with adjacent vertices x and y, the subtree rooted at x with respect to y is the subtree obtained by rooting T at y and then considering the rooted subtree rooted at x.

superconfiguration A configuration c' is a superconfiguration of another configuration c if for all v we have that $c(v) \subseteq c'(v)$.

troublemaker configuration A configuration is deemed a troublemaker relative to a subgraph of the reconfiguration graph if it cannot be part of any strategy contained in that subgraph.

universal property The trivial configuration property which accepts any configuration.

upset A set S of configurations is an upset if for all $c \in S$ we have that all superconfigurations c' of c are in S.

useless configuration A configuration is deemed useless relative to a subgraph r of the reconfiguration graph if it is both a troublemaker, and r contains all edges in its neighbourhood in the full reconfiguration graph.

valid configuration For a given instance $\mathcal{P}$ a configuration c is valid if every vertex is assigned at most s guards, and c satisfies I_γ.

weight of a block Let $N = (V_1, \ldots, V_k)$ be a neocolonization. The *weight* of a block $V_i \in N$ is defined to be 1 if $G[V_i]$ induces a clique, and $\gamma_c(G[V_i]) + 1$ otherwise.

weight of a neocolonization The weight of the neocolonization N is the sum of the weights of each part in N. We denote this with $w(N)$.

Bibliography

[1] S. Bard, C. Duffy, M. Edwards, G. Macgillivray, and F. Yang. Eternal domination in split graphs. *J. Comb. Math. Comb. Comput*, 101:121–130, 2017.

[2] V. Blažej, J. M. Křišťan, and T. Valla. Efficient attack sequences in m-eternal domination, 2022.

[3] A. Bonato, J. Janssen, and E. Roshanbin. Burning a graph as a model of social contagion. In *Algorithms and Models for the Web Graph*, volume 8882 of *Lecture Notes in Computer Science*, pages 13–22. Springer International Publishing AG, Switzerland, 2014.

[4] A. Bonato and G. MacGillivray. Characterizations and algorithms for generalized cops and robbers games. *arXiv.org*, 2017.

[5] A. P. Burger, E. J. Cockayne, W. R. Gründlingh, C. M. Mynhardt, J. H. van Vuuren, and W. Winterbach. Infinite order domination in graphs. *J. Combin. Math. Combin. Comput.*, 50:179–194, 2003.

[6] A. P. Burger, E. J. Cockayne, W. R. Gründlingh, C. M. Mynhardt, J. H. van Vuuren, and W. Winterbach. Finite order domination in graphs. *J. Combin. Math. Combin. Comput.*, 49:159–175, 2004.

[7] G. Chartrand, L. Lesniak, and P. Zhang. *Graphs and digraphs*. Textbooks in mathematics. Chapman and Hall/CRC, an imprint of Taylor and Francis, Boca Raton, FL, sixth edition. edition, 2015.

[8] E. J. Cockayne, O. Favaron, and C. M. Mynhardt. Secure domination, weak Roman domination and forbidden subgraphs. *Bull. Inst. Combin. Appl.*, 39:87–100, 2003.

[9] S. Finbow, S. Gaspers, M.-E. Messinger, and P. Ottaway. A note on the eternal dominating set problem. *International journal of game theory*, 47(2):543–555, 2018.

[10] R. Gera, T. W. Haynes, S. T. Hedetniemi, and M. A. Henning. *An Annotated Glossary of Graph Theory Parameters, with Conjectures*, pages 177–281. Springer International Publishing, 2018.

[11] W. Goddard, S. Hedetniemi, and S. Hedetniemi. Eternal security in graphs. *J. Combin. Math. Combin. Comput.*, 52, 01 2005.

[12] B. Hartnell and C. M. Mynhardt. Independent protection in graphs. *Discrete Mathematics*, 335:100–109, 11 2014.

[13] J. I. Hoepner. Eternal roman domination, 2020.

[14] W. Klostermeyer, M. Lawrence, and G. MacGillivray. Dynamic dominating sets: The eviction model for eternal domination. *J. Combin. Math. Combin. Comput.*, 97:247–269, 05 2016.

[15] W. Klostermeyer and G. MacGillivray. Eternal dominating sets in graphs. *J. Combin. Math. Combin. Comput.*, 68:97–111, 02 2009.

[16] W. Klostermeyer and C. Mynhardt. Secure domination and secure total domination in graphs. *Discussiones Mathematicae Graph Theory*, 28(2):267–284, 2008.

[17] W. Klostermeyer and C. M. Mynhardt. Edge protection in graphs. *Australas. J. Combin.*, 45, 10 2009.

[18] W. Klostermeyer and C. M. Mynhardt. Eternal total domination in graphs. *Ars Combinatoria*, 68:472–492, 10 2012.

[19] W. Klostermeyer and C. M. Mynhardt. Protecting a graph with mobile guards. *Appl. Anal. Discrete Math.*, 10, 07 2014.

[20] K. Kurita, K. Wasa, H. Arimura, and T. Uno. Efficient enumeration of dominating sets for sparse graphs. *Discret. Appl. Math.*, 303:283–295, 2021.

[21] E. Moss. Eternal paired domination in trees, 2022.

[22] C. S. ReVelle and K. E. Rosing. Defendens imperium romanum: A classical problem in military strategy. *The American Mathematical Monthly*, 107(7):585–594, 2000.

[23] I. Stewart. Defend the roman empire. *Scientific American*, 281(6):136–138, 1999.

[24] V. Virgelot. *A Study of Eternal Dominating Sets in Graphs*. PhD thesis, University of Victoria, 2023.