

Forbidden Minors for a Pursuit Game on Graphs

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Abhinanda Bhattacharyya

Annandale-on-Hudson, New York
May, 2013

Abstract

We study a pursuit game involving one or more pursuers and an invisible target on finite, connected topological graphs. The winning condition for this game leads to a sequence of minor closed properties, and one of our main goals is to characterize the forbidden minors for these properties. We also study simple graphs, multigraphs and topological graphs in the context of this game, and explore the complicated relationship between the forbidden graph characterizations for each. We conclude with some results on how the minor closed properties we are studying are related to pathwidth, one of the main concepts underlying the proof of the Forbidden Minor Theorem. As part of our research, we created a computer program on *Processing*, an open source programming language, which enables us to interactively play the game on a computer. The code for this program is given in the concluding chapter of this project.

Contents

Abstract	1
Dedication	7
Acknowledgments	8
1 Graph Theory Background	13
1.1 Simple Graphs	13
1.2 Multigraphs	21
1.3 Topological Graphs	30
2 The Robots-Target Game	39
2.1 Defining the Game	39
2.2 Robot Number	44
3 Forbidden Subquotients and Forbidden Minors	50
3.1 The Complete Set of Forbidden Subquotients for $r(G) = 1$	50
3.2 Net Graph, Triple Edge and the $(3, 2)$ -Tree	55
3.3 The Complete Set of Forbidden Subquotients for $r(G) = 2$	61
3.4 Forbidden Subquotients for $r(G) = 3$	64
3.5 Forbidden Minors	69
3.5.1 The Forbidden Multigraph Minors for $r(G) = 1$	69
3.5.2 The Forbidden Multigraph Minors for $r(G) = 2$	71
3.5.3 Forbidden Simple Graph Minors	82
4 Robot Number and Pathwidth	83
4.1 Path Decomposition and Pathwidth	84
4.2 The Relationship Between Pathwidth and the Robots-Target Game	88

<i>Contents</i>	3
5 Simulating the Robots–Target Game on a Computer	93
Bibliography	103

List of Figures

1.1.1	Some simple graphs.	14
1.1.2	We obtain G' by deleting an edge from G	14
1.1.3	We obtain G' by contracting an edge in G	15
1.1.4	M_1 , M_2 and M_3 are simple graph minors of G	16
1.1.5	M_1 , M_2 and M_3 are simple graph minors of G	17
1.1.6	$K_{3,3}$ and K_5 are the only forbidden minors for planar graphs.	20
1.1.7	These are planar drawings of K'_5 and K''_5	20
1.1.8	These are planar drawings of K'_3 and $K''_{3,3}$	21
1.2.1	Some multigraphs.	22
1.2.2	G' is a simplification of G	22
1.2.3	G' , G'' and G''' are obtained from G by edge deletion.	23
1.2.4	G'' and G' are obtained from edge contractions of G	24
1.2.5	Multigraph edge contraction and simple graph edge contraction can yield different graphs.	24
1.2.6	M_1 , M_2 and M_3 are multigraph minors of G	25
1.2.7	(a) This is the forbidden simple graph minor set for the set of simple graph pseudotrees. (b) This is the forbidden multigraph minor set for the set of multigraph pseudotrees.	27
1.3.1	Subdividing edge e_1	30
1.3.2	Some topological graphs.	31
1.3.3	G' is the graph obtained from a topological edge contraction in G	33
1.3.4	Cutting edge e_1	34
1.3.5	The forbidden subquotient set for the set of topological graphs with at most one vertex having degree 3 or greater.	37
1.3.6	The forbidden multigraph minor set for the set of multigraphs with at most one vertex having degree 3 or greater.	38

2.1.1	A game with one robot and the target on the claw graph.	42
2.1.2	A game with two robots and the target on the claw graph.	43
2.1.3	Two robots clearing a graph.	44
2.1.4	Two robots clearing a graph.	45
2.2.1	The wheel graph has robot number 4.	48
2.2.2	Five robots win on the bipyramid graph.	49
3.1.1	These are the forbidden subquotients for $r(G) = 1$	51
3.1.2	We mark the edges of the claw graph.	52
3.1.3	Loop with one edge cut.	52
3.2.1	The net graph, triple edge and the $(3, 2)$ -tree are forbidden subquotients for $r(G) = 2$	55
3.2.2	The net graph is divided into three subspaces.	58
3.2.3	We show spines in N' , N'' and N'''	58
3.2.4	We denote the edges of the triple edge as 1, 2 and 3. The subspaces of the graph which are shown to be thickened are shelters.	59
3.2.5	We show spines in TE' and TE''	59
3.2.6	The $(3, 2)$ -tree is divided into three subspaces.	60
3.2.7	We show spines in T' and T''	60
3.4.1	These are the maximal minors of G obtained in Theorem 3.4.1.	66
3.4.2	We divide K_4 into four subspaces.	67
3.4.3	Three robots have a winning strategy on K_4'	68
3.4.4	Three robots have a winning strategy on K_4''	68
3.4.5	This is a partial list of forbidden subquotients for three robots.	70
3.5.1	This is the graph obtained from attaching two pendant edges of the claw graph at their degree 1 vertices.	71
3.5.2	This is the graph obtained from attaching a degree 1 vertex of a pendant edge in the claw graph to the central vertex.	71
3.5.3	The triple edge is a forbidden multigraph minor.	72
3.5.4	This is the graph obtained from attaching a degree 1 vertex of a pendant edge of the net graph to a vertex on the triangle other than the one from which the edge originates.	73
3.5.5	This is the graph obtained from attaching two pendant edges of the net graph to each other at their degree 1 vertices.	73
3.5.6	These are all the forbidden multigraph minors for $r(G) = 2$ that are obtained from linking pendant edges of the net graph.	74
3.5.7	This is the graph obtained from attaching the degree 1 vertex of a pendant edge in the $(3, 2)$ -tree to its origin vertex.	75
3.5.8	This is the graph obtained from attaching the degree 1 vertex of a pendant edge in the $(3, 2)$ -tree to some degree 3 vertex other than its origin vertex and the central vertex.	76
3.5.9	This is the graph obtained from attaching the degree 1 vertex of a pendant edge in the $(3, 2)$ -tree to the degree 1 vertex of a pendant edge with which it does not share an endpoint.	76

3.5.10 These are all the forbidden multigraph minors for $r(G) = 2$ that are obtained from linking pendant edges of the $(3, 2)$ -tree. 77

3.5.11 Spines in the maximal minors of N_1 77

3.5.12 Spines in the maximal minors of N_2 78

3.5.13 Spines in the maximal minors of N_3 78

3.5.14 Spines in the maximal minors of G_1 79

3.5.15 Spines in the maximal minors of G_2 79

3.5.16 Spines in the maximal minors of G_3 80

3.5.17 Spines in the maximal minors of G_4 80

3.5.18 Spines in the maximal minors of G_5 81

3.5.19 Spines in the maximal minors of G_6 81

3.5.20 This is the complete set of forbidden simple graph minors for one robot. . . 82

3.5.21 This is the complete set of forbidden simple graph minors for two robots. . 82

4.1.1 A path decomposition of G 85

4.1.2 W is a representation of a path decomposition in G 85

4.1.3 A path decomposition of G 86

4.1.4 A path decomposition of the $(3, 2)$ -tree. 87

4.1.5 A path decomposition P of the net graph. 87

4.1.6 A different representation of path decomposition P of the net graph. . . . 88

4.2.1 Relating robot number and pathwidth. 90

4.2.2 This is a path decomposition of the $(3, 2)$ -tree which has pathwidth 3. . . . 92

5.0.1 Stage 1: drawing a graph in the sketch. 94

5.0.2 Stage 2: placing the robots and the target on the graph. 95

5.0.3 Stage 3: selecting and moving the robots. 96

5.0.4 Stage 4: moving the target. 97

Dedication

I would like to dedicate this project to my family.

Acknowledgments

I would like to thank my adviser, Jim Belk, for his brilliant guidance and for making Senior Project the incredibly rewarding experience that it has been. I owe my interest in graph theory first to Sam Hsiao for making it sound so intriguing when he mentioned it in Combinatorics class a few semesters ago, and then to Maria Belk, who gave me the foundation I needed to write this project. I would like to thank Keith O'Hara for his guidance on the coding part of this project, and for getting me interested in computer science. I am also very grateful to Sanjaya DeSilva for all his advice this year.

My family for their love and support, and for giving me the best education someone could ask for.

And finally, to my friends, for the philosophy, the dancing and the absurd humor. Thank you doesn't even say it.

Introduction

Pursuit–evasion games are problems in which one group of players tries to capture another under certain specified conditions in some given environment. Many pursuit–evasion games, or simply pursuit games as they are often called, are played on graphs, and the research in this project focuses on one such game. Researchers are interested in pursuit games on graphs for a number of reasons, and one among these reasons is the connection that can often be drawn between concepts in graph theory and the concepts that pertain to a specific game. In particular, certain pursuit games on graphs are related to treewidth and pathwidth [9] [6] [2], which are some of the fundamental concepts used in Robertson and Seymour’s proof of the Forbidden Minor Theorem [7].

The Forbidden Minor Theorem is a fundamental result in graph theory. A minor of a graph is a graph obtained by a sequence of edge deletions and edge contractions in the original graph in any order. A set S of graphs is said to be minor closed if for any graph in S , any minor of that graph is also in S . The Forbidden Minor Theorem states that for any minor closed set S , there exists a finite list M of forbidden minors which are graphs that may not be contained in S . The Forbidden Minor Theorem gives a way of

characterizing any minor closed set of graphs. In the past few years, complete forbidden minors sets have been determined for a number of minor closed properties, including planarity, trees, treewidth at most k and pathwidth at most k .

In this project, we define a new pursuit game on graphs called the robots–target game, where the winning condition gives rise to a sequence of minor closed properties. The robots–target game is played on finite, connected topological graphs and involves two players, the robots (the pursuers) and the target (the pursued). The robots are unaware of the target’s location on a graph, and must search the entire graph until they catch the target, which is the winning condition for them. The target wins if it has a strategy to indefinitely evade the robots on a given graph.

In a previous senior project, Tina Zhang studied forbidden minor theory as it pertained to a pursuit game which was a slightly modified version of the robots–target game [10]. Unlike in the robots–target game, the pursuers, or the cops in Zhang’s game are aware of the location of the pursued, or the robber at any given time. Zhang’s project was the motivation for studying forbidden minor theory in the context of the robots–target game.

We define robot number, a fundamental concept used throughout this project, which is the minimum number of robots required to win against the target on a graph. We show that for each robot number k , the graphs on which k robots can win against the target form a minor closed set. One of our primary goals was to give forbidden minors for a sequence of robot numbers, and we were able to find a complete list of such forbidden minors for robot numbers 1 and 2 and a partial list for robot number 3. There are exactly two forbidden minors for robot number 1 and three for robot number 2. The partial list of forbidden minors obtained for robot number 3 contains 21 graphs at this time.

We also studied the relationship of forbidden graph characterizations across simple graphs, multigraphs and topological graphs in the context of the robots–target game. In general, it is possible to study forbidden minor theory for simple graphs, multigraphs

and topological graphs, but we found it easiest to work within the setting of topological graphs when investigating robot number. The results that we found for topological graphs enabled us to derive forbidden multigraph minors and subsequently forbidden simple graph minors. A portion of the research in this project involved finding a way to establish the relationship between forbidden graph characterizations for simple graphs, multigraphs and topological graphs. We gave algorithms for obtaining the forbidden graph characterization of one type of graph based on that of another, and found that the relationships between these characterizations may be complicated.

In her senior project, Zhang defines a minor closed property called cop number, which is analogous to robot number, and gives bounds for her cop number using a graph property known as treewidth. The set of graphs with treewidth at most k is minor closed and the forbidden minors have been determined for $k \leq 3$ [1]. Zhang was able to use the forbidden minor characterization of treewidth to help her in her research of forbidden minors for cop number.

Our game does not seem to be related to treewidth. It is, however, closely related to another graph property called pathwidth, which is analogous to treewidth. As in treewidth, the set of graphs with pathwidth at most k is known to be minor closed. Barát explores pathwidth in the context of a pursuit game which is similar to ours [2], and we used the results from his paper to guide our research on pathwidth in relation to robot number. Like Zhang, we were able to bound robot number within a certain pathwidth range, and we give a conjecture that this bound is in fact even narrower than what we show.

Lastly, as part of this research, we created a simulation of the robots-target game on *Processing*, an open source programming language which is built on Java. The program allows the user to interactively play robots-target on a computer in the form of a turn-taking game, though it should be noted that the game is not originally defined to be one

as such. We used this program as a platform to play our game, and it was particularly useful in our search for forbidden minors for different robot numbers.

This project is organized as follows. Chapter 1 introduces basic concepts in graph theory. In this chapter, we distinguish between simple graphs, multigraphs and topological graphs and also show how these graphs are related to one another. In Chapter 2, we rigorously define the robots-target game and give some basic results on robot number. In Chapter 3 we give a complete list of forbidden graphs for robot numbers 1 and 2 and a partial list for robot number 3. Chapter 4 explores the relationship between pathwidth and robot number. Finally, in Chapter 5, we describe our computer program and give the code to run it.

1

Graph Theory Background

1.1 Simple Graphs

Although the research in this project is largely based on topological graphs, we will begin this chapter with a discussion of simple graphs. One of the key concepts introduced in this section is that of a simple graph minor. We give the definition of a simple graph minor and show some examples in this section, and will make use of this concept once again in Chapter 3.

The discussion of simple graph minors leads us to the statement of the Forbidden Minor Theorem, which is discussed at length at the end of this section. This theorem was first proven by Robertson and Seymour for simple graphs. In the following sections, we will extend the statement of this theorem to multigraphs and topological graphs.

Finally, in Chapter 4, we will go on to discuss the relationship between the robots-target game and pathwidth. Pathwidth, which will be defined and explained later on, pertains only to simple graphs. This is a motivating reason for discussing simple graphs in detail.

Recall first the definition of a simple graph.

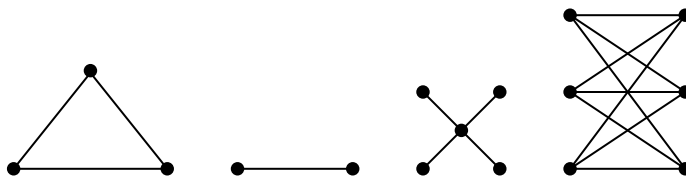
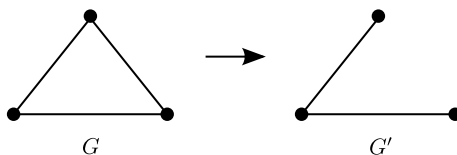


Figure 1.1.1: Some simple graphs.

Figure 1.1.2: We obtain G' by deleting an edge from G .

Definition 1.1.1. A **simple graph** is a graph having no loops and in which any two distinct vertices have at most one edge connecting them. \triangle

We show some simple graphs in Figure 1.1.1.

We say that a graph is **connected** if there is a path between any two vertices in the graph. Note that we will use the following convention throughout this project.

Convention 1.1.2. All graphs in this project are **assumed to be connected** and we assume that any operations described will not disconnect a graph.

We will now define two fundamental operations on simple graphs.

Definition 1.1.3. Let $G = (V, E)$ be a graph. Given any edge $e \in E$, we can **delete** e to obtain the graph $G' = (V, E')$ where $E' = E - \{e\}$. \triangle

An edge deletion simply removes the edge connecting two vertices in a simple graph. In Figure 1.1.2, for example, we obtain G' by deleting an edge in G .

Note that since we are only concerned with connected graphs in this research, we do not allow edge deletions that will disconnect a graph.

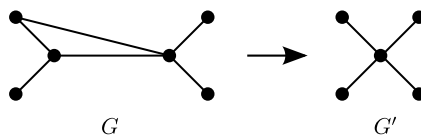


Figure 1.1.3: We obtain G' by contracting an edge in G .

Definition 1.1.4. Let $G = (V, E)$ be a graph. Let $e \in E$ and let $u, v \in V$ be the endpoints of e . The **contraction of e** is an operation that creates a new graph $G' = (V', E')$ which is defined as follows:

1. The vertex set V' consists of all the vertices in V other than u and v , together with a new vertex w .
2. If $x, y \in V'$ and $x, y \neq w$, then $\{x, y\} \in E'$ if and only if $\{x, y\} \in E$.
3. If $x \in V'$ and $x \neq w$, then $\{x, w\} \in E'$ if and only if $\{x, u\} \in E$ or $\{x, v\} \in E$. \triangle

The contraction of edge e in G in the above definition results in the endpoints u and v of e merging to form a single vertex w in G' . Any edge that was incident on either of u or v in G becomes incident on w in G' , and any edge that was incident on vertices other than u or v in G is incident on the same vertices in G' .

Example 1.1.5. Figure 1.1.3 shows an edge contraction of the graph G , where we obtain G' by contracting the middle edge of G . Observe that the two edges in the upper left side of G are combined to form a single edge in G' . \diamond

We will now give a definition for a simple graph minor using the notions of edge deletions and edge contractions.

Definition 1.1.6. Let G be a simple graph. A **simple graph minor** H of G is a simple graph obtained from G by a sequence of edge deletions and edge contractions in any order. \triangle

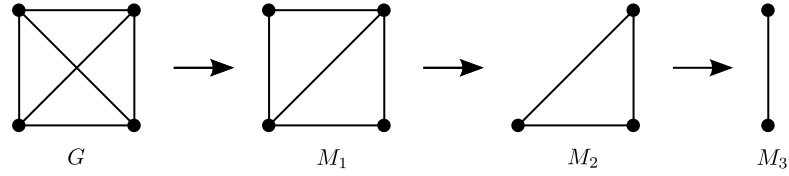


Figure 1.1.4: M_1 , M_2 and M_3 are simple graph minors of G .

Example 1.1.7. In Figures 1.1.4 and 1.1.5, we show a sequence of minors of a given graph G . In Figure 1.1.4, we obtain M_1 by deleting one of the diagonal edges of G . M_2 is obtained from M_1 by contracting the top edge of M_1 . Note that the diagonal edge and the edge on the left side of M_1 are identified in this contraction. Finally, M_3 is obtained from M_2 by contracting the bottom edge of M_2 . The diagonal edge and the edge on the right side of M_2 are identified in this contraction.

In Figure 1.1.5, we obtain M_1 from G by contracting the middle edge of G . Observe that the top left and the top right edges are identified in this contraction, as are the bottom left and right side edges. We obtain M_2 from M_1 by contracting the top edge of M_1 . Note that we could not have obtained M_2 from M_1 by deleting the top edge of M_1 , as this operation would have left an isolated vertex. In fact, as a consequence of Convention 1.1.2, we are not allowed to delete any edge in M_1 . Similarly, M_3 is obtained from M_2 by contracting an edge in M_2 . \diamond

Example 1.1.8. Let G be a graph. Then any connected subgraph G' of G is a minor of G , though this may require some edge contractions if the subgraph has fewer vertices than the original graph. Correspondingly, every minor can be obtained by contracting some of the edges of some connected subgraph. \diamond

We will now discuss the notion of a minor closed set of graphs.

Definition 1.1.9. Let S be a set of simple graphs and let G be a graph. S is **minor closed** if for all $G \in S$, any minor G' of G is also in S . \triangle

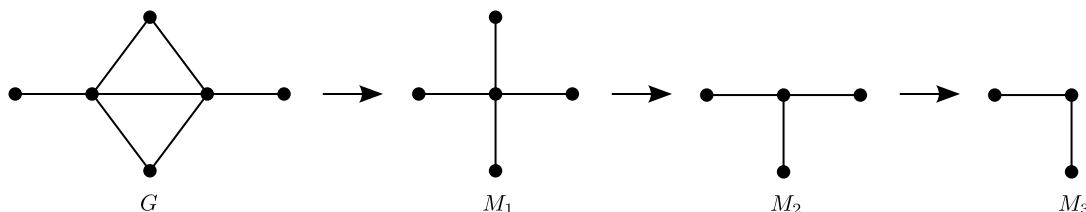


Figure 1.1.5: M_1 , M_2 and M_3 are simple graph minors of G .

We present some examples of minor closed properties in simple graphs below.

Example 1.1.10. Let $n \in \mathbb{N}$ and let S be the set of graphs having at most n vertices. Then S is minor closed. Let G be a graph on at most n vertices. Any minor of G also has at most n vertices.

Similarly, let S be the set of graphs having at most n edges. Then any minor of a graph in S will also have at most n edges, and we say that S is minor closed. \diamond

Example 1.1.11. The set T of trees is minor closed. In particular, it is not possible to delete an edge in a tree, and if we contract an edge in a tree, the result is always another tree. \diamond

Example 1.1.12. Recall that a graph G is **planar** if and only if G can be drawn on a plane such that all of the edges in G meet only at the endpoints of the edges. Let P be the set of all planar graphs. It is not hard to see that P is minor closed.

Similarly, for any surface (such as a torus or projective plane), the set of graphs that can be drawn on that surface without edge crossings is minor closed. \diamond

The notion of minor closed sets of graphs allows us to state the Forbidden Minor Theorem for simple graphs.

Theorem 1.1.13 (The Forbidden Minor Theorem for Simple Graphs). *Let S be a minor closed set of graphs. Then there exists a finite number of graphs M_1, \dots, M_k such that for any graph G , we have $G \in S$ if and only if G does not contain any M_i as a minor.*

Note that the set of graphs M_1, \dots, M_k is not uniquely determined. This means that we are able to include any number of graphs in the set M_1, \dots, M_k . However, there is always some set of minimal graphs which is required to be forbidden from a given minor closed set. This set of graphs is called a forbidden minor set, and we give a rigorous definition of forbidden minors below.

Definition 1.1.14. We say that G is a **forbidden minor** for S if G is not in S , and every minor of G other than itself is in S . △

We will now state a more precise version of the Forbidden Minor Theorem.

Theorem 1.1.15. *Let S be a minor closed set of graphs and let G be a graph. Then:*

1. S has finitely many forbidden minors M_1, \dots, M_k .

2. $G \in S$ if and only if G does not contain any M_i as a minor. □

The following theorem gives us a simple test to determine whether a given graph is a forbidden minor for some minor closed set.

Theorem 1.1.16. *Let S be a minor closed set of graphs. Then G is a forbidden minor for S if and only if for every edge e of G that can be deleted, the graph G' resulting from deleting e is in S , and for every edge e of G , the graph G' resulting from the contraction of e is in S , and $G \notin S$.*

Theorem 1.1.16 gives us an algorithm for checking minimality for a certain forbidden graph for a minor closed set. Checking minimality for forbidden simple graph minors, therefore, involves two steps:

1. Delete one of every possible non-alike edge in a graph, and check if the resulting graph is in the minor closed set being characterized.

2. Contract one of every possible non-alike edge in a graph, and check if the resulting graph is in the minor closed set being characterized.

In the following examples, we will give some minor closed sets along with the forbidden minors characterizing them, and then we will briefly describe minimality for the forbidden minors.

Example 1.1.17. Let $n \in \mathbb{N}$ and let S be the set of graphs having at most n edges. Then the forbidden minors for S are all connected graphs with $n + 1$ edges. In particular, if G is any graph with $n + 1$ edges, then G is not an element of S , but contracting or deleting any edge in G results in a graph which is an element of S . \diamond

Example 1.1.18. Let T be the set of trees. Clearly, the triangle is a forbidden minor for T . We claim that this is the only forbidden minor. To prove this, we will show that any graph not in T has the triangle as a minor. Let G be a graph which is not in T . Then G must not be a tree, which means it must be a cycle. We know that we can then obtain a triangle from G by a sequence of edge contractions. We can conclude that a tree must not have any triangle as a minor, and the triangle is the only forbidden minor for the set of trees. \diamond

Example 1.1.19. Let P be the set of planar graphs. Let K_5 and $K_{3,3}$ be the graphs shown in Figure 1.1.6. Though the proof of this statement is difficult, it is known that K_5 and $K_{3,3}$ are non-planar. We claim that K_5 and $K_{3,3}$ are forbidden minors for the set P .

We shall prove this using Theorem 1.1.16. Figure 1.1.7 shows planar drawings of K'_5 and K''_5 respectively, where K'_5 is obtained from K_5 by deleting an edge and K''_5 is obtained from K_5 by contracting an edge. Similarly, Figure 1.1.8 shows planar drawings of $K'_{3,3}$ and $K''_{3,3}$ respectively, where $K'_{3,3}$ is obtained from $K_{3,3}$ by deleting an edge and $K''_{3,3}$ is obtained from $K_{3,3}$ by contracting an edge. These drawings suffice to show that K_5 and $K_{3,3}$ are forbidden minors for planar graphs.

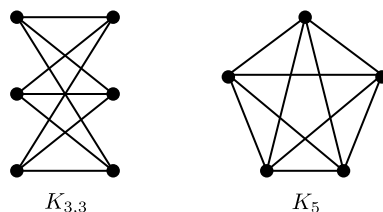


Figure 1.1.6: $K_{3,3}$ and K_5 are the only forbidden minors for planar graphs.

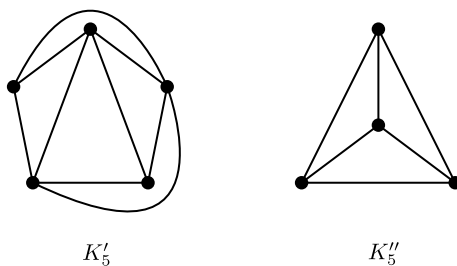


Figure 1.1.7: These are planar drawings of K'_5 and K''_5 .

A well known theorem called **Kuratowski's Theorem** states that K_5 and $K_{3,3}$ are the only forbidden minors for planarity [8]. Note that while our argument shows that these graphs are indeed forbidden minors, it is much harder to show that makes up the complete list of forbidden minors. Kuratowski's Theorem does precisely that.

Kuratowski's Theorem was the motivating result that caused graph theoreticians to conjecture that any minor closed property could be characterized by some finite set of forbidden minors. \diamond

In general, it is much easier to find forbidden minors than it is to prove that a set of graphs forms a complete set of forbidden minors. In Chapter 3, we give two extensive arguments to show a complete set of forbidden minors, and we will observe some of the complexity involved in finding a complete list.

The Forbidden Minor Theorem, formerly known as Wagner's Conjecture [7], was proved by Neil Robertson and Paul D. Seymour in 2004 after they had worked on it for a span

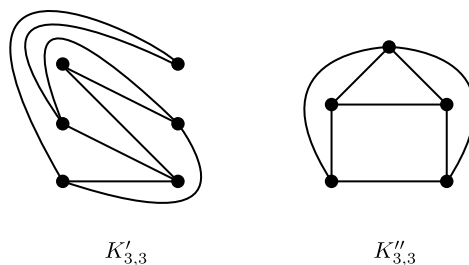


Figure 1.1.8: These are planar drawings of K'_3 and $K''_{3,3}$.

of several years. The proof of the theorem is long and involved and uses concepts such as treewidth and pathwidth, which will be discussed later on in this project.

The Forbidden Minor Theorem is a fundamental theorem in graph theory because it gives a way of characterizing any minor closed set of graphs. However, there is no known algorithm for determining what the forbidden minors are for a minor closed set of graphs. One of the goals of this project is to find an approach to determine what the forbidden minors are for a certain minor closed property.

1.2 Multigraphs

In this section we will introduce the concept of multigraphs. Multigraphs provide an alternative context for studying the Forbidden Minor Theorem, which is the motivation for our research in Chapter 3.

Definition 1.2.1. A **multigraph** is a graph in which loops and multiple edges between vertices are allowed. △

We give some examples of multigraphs in Figure 1.2.1.

Note that every simple graph is a multigraph, but not every multigraph is necessarily a simple graph. However, we can use the following operation to obtain a simple graph from any given multigraph.

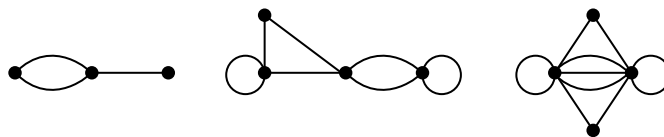
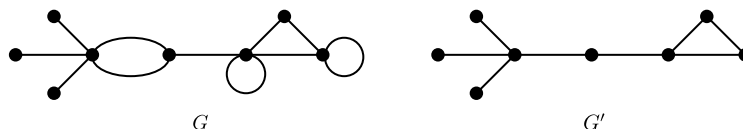


Figure 1.2.1: Some multigraphs.

Figure 1.2.2: G' is a simplification of G .

Definition 1.2.2. Let G be a multigraph. The **simplification** of G is the simple graph obtained from G as follows:

1. For every pair of vertices in G connected by multiple edges, remove all but one of these edges.
2. Remove all loops in G . △

In Figure 1.2.2 we show a simplification G' of a multigraph G .

We will now describe forbidden minor theory for multigraphs. Like for simple graphs, we have two basic operations for multigraphs: edge deletion and edge contraction. Edge deletion in multigraphs is analogous to edge deletion in simple graphs, which was defined in Definition 1.1.3. We illustrate the notion of edge deletion in multigraphs with the following example.

Example 1.2.3. Consider Figure 1.2.3 where we obtain G' , G'' and G''' from G by a sequence of edge deletions. First we delete the edge which forms the loop inside the double edge. Then we go on to delete one of the middle edges of the resulting graph G' . Finally, to obtain G''' , we delete the top left edge. ◇

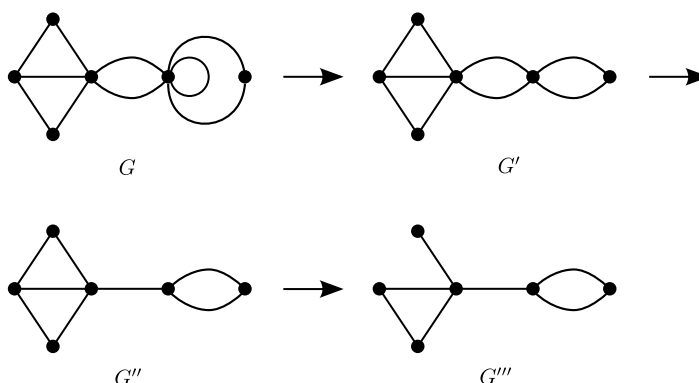


Figure 1.2.3: G' , G'' and G''' are obtained from G by edge deletion.

Edge contraction in multigraphs, however, is defined differently than for simple graphs. We give the definition below.

Definition 1.2.4. Let $G = (V, E)$ be a multigraph, and let $e_0 \in E$ with endpoints $u, v \in V$. The **contraction of** e_0 is an operation that creates a new graph $G' = (V', E')$ as defined below:

1. V' consists of all the vertices in V other than u and v , together with a new vertex w .
2. Any edge $e \in E$ which was incident on either or both of u or v in G becomes incident to $w \in V'$.
3. Any edge $e \in E$ which was not incident on either of u or v in G corresponds to some edge e' in G' which is incident on the same vertices as e . \triangle

Note that an edge contraction in a multigraph can result in the creation of loops or multiple edges. We give two examples below to illustrate edge contraction in multigraphs. The second example will help to demonstrate how edge contraction can be different in multigraphs and simple graphs.

Example 1.2.5. Consider Figure 1.2.4. We contract the middle edge of G to obtain G' . We can see that the two triangles in the original graph G now become two double edges

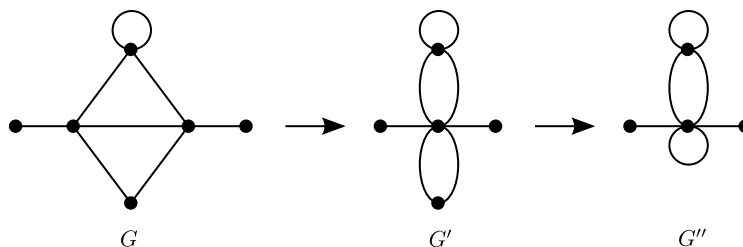
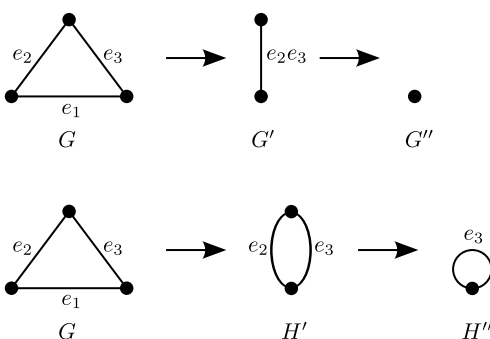
Figure 1.2.4: G'' and G' are obtained from edge contractions of G .

Figure 1.2.5: Multigraph edge contraction and simple graph edge contraction can yield different graphs.

in G' . We contract one of the edges of one of the double edges in G' to obtain G'' . This double edge becomes a loop. \diamond

Example 1.2.6. Consider Figure 1.2.5, where we contract the edges of a triangle. The first row indicates simple graph edge contraction, where contracting edge e_1 in G gives us edge e_1e_2 in G' . Contracting edge e_1e_2 gives us the single vertex graph, G'' . Now, consider the second row. A multigraph edge contraction of the same edge e_1 in G gives us a double edged graph H' with edges e_2 and e_3 which are both incident on the same vertices. Contracting edge e_2 gives us a graph H'' which is simply a loop with edge e_3 . Observe that simple graph edge contraction and multigraph edge contraction of the same original graph G give us different resulting graphs. \diamond

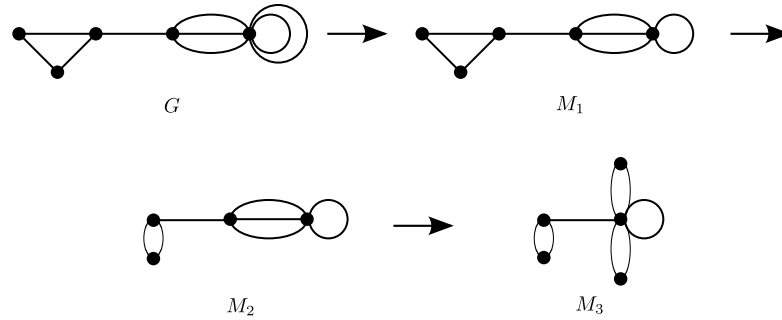


Figure 1.2.6: M_1 , M_2 and M_3 are multigraph minors of G .

In the previous section, we had discussed the notion of minors in simple graphs. An analogous notion exists for multigraphs, which we define below.

Definition 1.2.7. Let G be a multigraph. A **multigraph minor** H of G is a multigraph obtained from G by a sequence of edge deletions and edge contractions in any order. \triangle

Example 1.2.8. In Figure 1.2.6, we illustrate how we may obtain multigraph minors for some given graph G . We obtain the minor M_1 by deleting a loop in G . We obtain M_2 by contracting an edge in the triangle in M_1 . Finally, M_3 is obtained by contracting an edge in the triple edge in M_2 . \diamond

In the previous section, we defined what it meant for a set of simple graphs to be minor closed. A similar notion exists for multigraphs.

Definition 1.2.9. Let S be a set of multigraphs. S is said to be **minor closed** if any graph which is a graph minor of some graph in S is also in S . \triangle

Though the Forbidden Minor Theorem was originally stated for simple graphs, the concept of forbidden graph characterization can also be extended to multigraphs. We give the analogous statement of Theorem 1.1.13 for multigraphs below [7].

Theorem 1.2.10 (The Forbidden Minor Theorem for Multigraphs). *Let S be a minor closed set of multigraphs. Then there exists a finite number of graphs M_1, \dots, M_k such that*

for any graph G , we have $G \in S$ if and only if G does not contain any M_i as a multigraph minor.

Forbidden minors for any minor closed set of multigraphs are defined exactly as in Definition 1.1.15 in the previous section. The concept of minimality which we had introduced in the previous section with reference to forbidden simple graph minors pertains to forbidden multigraph minors as well, and we can use the algorithm defined in Theorem 1.1.16 as our test for checking minimality for forbidden multigraph minors.

However, since edge contraction in multigraphs and edge contraction in simple graphs are defined differently, in general, we find that it is often the case that forbidden multigraph minor sets and forbidden simple graph minor sets for equivalent minor closed properties are different. The examples below will help clarify this point.

Example 1.2.11. Let T be the set of trees. Note that T is a minor closed set. In Example 1.1.18 in the previous section, we showed that the only forbidden simple graph minor for the T is the triangle. However, the triangle is not a forbidden multigraph minor for T : in the context of multigraphs, contracting an edge in a triangle produces a double edge, which is not in T . Instead, the only forbidden multigraph minor for T is a single vertex with a loop. Minimality is straightforward and similar to the proof of minimality in Example 1.1.18. \diamond

Example 1.2.12. Forbidden minors for multigraphs and simple graphs are not always different. Let P be the set of planar multigraphs. P is a minor closed set. Recall from Example 1.1.19 that K_5 and $K_{3,3}$, shown in Figure 1.1.6, are the forbidden minors for planar simple graphs. These are also the only forbidden minors for planar multigraphs. \diamond

Example 1.2.13. A **pseudotree** is a connected graph with at most one cycle. It is not hard to see that the set S of all pseudotrees is a minor closed set. The only forbidden multigraph minor for S is a double loop, which is shown in Figure 1.2.7.

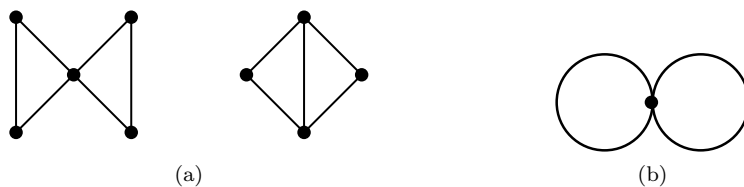


Figure 1.2.7: (a) This is the forbidden simple graph minor set for the set of simple graph pseudotrees. (b) This is the forbidden multigraph minor set for the set of multigraph pseudotrees.

The definition of a pseudotree applies just as well to simple graphs, and the set S' of simple pseudotrees is a minor closed set of simple graphs. S' is smaller than S because there are pseudotrees that are multigraphs but not simple graphs. S' has two forbidden minors which are shown in Figure 1.2.7. Note that in the context of multigraphs, both these graphs have the double loop as a multigraph minor. \diamond

Example 1.2.13 raises the question of how forbidden multigraph minors are related to forbidden simple graph minors. In the following discussion, we will show that every minor closed set of multigraphs has a corresponding minor closed set of simple graphs, and we can derive the forbidden simple graph minors from the forbidden multigraph minors by applying a basic algorithm.

Proposition 1.2.14. *Let G and G' be simple graphs. Then G' is a multigraph minor of G if and only if G' is a simple graph minor of G .*

Proof. Suppose first that G' is a simple graph minor of G . If G' is obtained from G by a sequence of simple graph edge contractions, then G' can be obtained from G by a sequence of multigraph edge contractions followed by the deletion of any multiple edges that result. We can see that if G' can be obtained by simple graph edge contractions and deletions, then G' can be obtained by multigraph edge contractions and deletions.

For the other direction, suppose that G' is a multigraph minor of G . We claim that G' is a simple graph minor of G . Since G' is a multigraph minor of G , it follows that G' can

be obtained by some connected subgraph H of G by contracting edges. Clearly, H is a simple graph minor of G . Since G' does not have multiple edges, all the edge contractions to get from H to G' are simple graph contractions. \square

Corollary 1.2.15. *Let S be a minor closed set of multigraphs, and let S' be the simple graphs in S . Then S' is a minor closed set of simple graphs.*

We will now begin our discussion on how to obtain forbidden simple graph minors from forbidden multigraph minors by defining what it means for an edge contraction to be unsimplifying [3]. Note that in this definition, the number of multiple edges refers to the number of extra edges in a graph, in that each double edge counts as one, each triple edge counts as two and so on.

Definition 1.2.16. Let G be a multigraph and let e be an edge in G . We say that the **contraction of e is unsimplifying** if it increases the number of loops or if it increases the number of multiple edges in G . \triangle

Note that contracting one of several parallel edges in a multigraph decreases the number of multiple edges but increases the number of loops, and therefore such a contraction counts as unsimplifying.

The following theorem gives us a simple way of obtaining a forbidden simple graph minor from some given forbidden multigraph minor. We also give the sketch of a proof for this theorem.

Theorem 1.2.17. *Let S be a minor closed set of multigraphs, and let S' be the set of all the simple graphs in S . Then, for every forbidden simple graph minor G' of S' , there exists a forbidden multigraph minor G of S such that G can be obtained from G' by a sequence of unsimplifying edge contractions.*

Sketch of a Proof for Theorem 1.2.17. Let G' be a forbidden simple graph minor for S' . Then $G' \in S$, so there exists some forbidden minor G for S such that G is a minor of G' . Then, it is possible to show that G can be obtained from G' in the following manner:

1. Delete some edges, then
2. Perform simple edge contractions, then
3. Perform unsimplifying contractions

[This holds true in general for any multigraph minors].

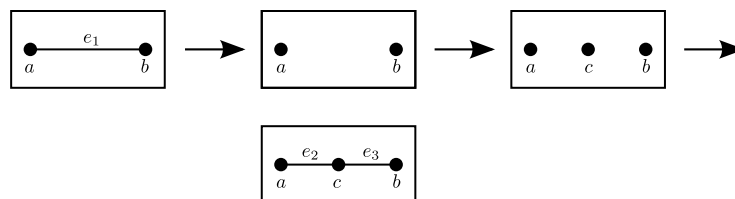
Since G' is minimal, we could not have deleted any edges in Step 1, nor could we have performed any simple edge contractions in Step 2. Therefore G is obtained from G' using a sequence of unsimplifying edge contractions. \square

Given a forbidden multigraph minor G for some minor closed property S , it is not hard to enumerate all possible simple graphs G' from which we can obtain G by a sequence of unsimplifying edge contractions. At the end of Chapter 3, we will use the algorithm from Theorem 1.2.17 to obtain the forbidden simple graph minor set from the forbidden multigraph minor set for the minor closed property that we are studying.

We conclude this section by giving the following theorems, which we will use in Chapter 3 [5].

Theorem 1.2.18. *Let G and H be graphs. Then H is a minor of G if and only if there exists a subgraph G' of G such that H is a contraction of G' .*

Theorem 1.2.19. *Let G and H be graphs. Suppose every vertex of H has degree less than or equal to 3. Then H is a contraction of G if and only if G is a subdivision of H .*

Figure 1.3.1: Subdividing edge e_1 .

1.3 Topological Graphs

This section introduces the concept of topological graphs. Topological graphs are a variation on multigraphs, and are the class of graphs that we will mainly deal with in Chapter 3.

Before explaining what it means for a graph to be topological, we will begin by discussing the topology of multigraphs.

Definition 1.3.1. Consider an edge e_1 with endpoints a and b as shown in Figure 1.3.1.

We **subdivide** e_1 as follows:

1. Remove e_1 , leaving isolated vertices a and b .
2. Add a vertex c .
3. Draw edge e_2 between vertices a and c and draw edge e_3 between vertices c and b .

We call this process an **edge subdivision** of e_1 . A **subdivision** G' of a graph G is obtained by a sequence of edge subdivisions. \triangle

When an edge in a graph is subdivided, the graph becomes combinatorially different but remains the same topologically. This notion is captured by the following definition.

Definition 1.3.2. Let G and H be multigraphs. We say that G and H are **homeomorphic** if some subdivision G' of G is isomorphic to some subdivision H' of H . \triangle

It is often helpful to think of two homeomorphic multigraphs as being the same graph. The following definition will give us an instance of when this is the case.

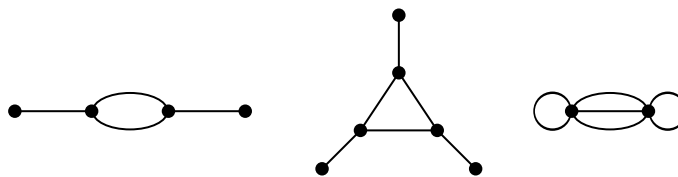


Figure 1.3.2: Some topological graphs.

Definition 1.3.3. Let S be a minor closed set of multigraphs. We say that S is **homeomorphism invariant** if for all G and H that are homeomorphic, $G \in S$ if and only if $H \in S$. \triangle

Some examples of homeomorphism invariant minor closed sets are the set of trees, the set of pseudotrees and the set of planar multigraphs. However, the minor closed set of multigraphs on n or fewer vertices and the minor closed set of multigraphs with n or fewer edges are not homeomorphism invariant.

When studying any homeomorphism invariant property, it is particularly useful to think of homeomorphic graphs as being the same. The property that we will study in the following chapters is homeomorphism invariant, which is the motivation for studying topological graphs. We define topological graphs below.

Definition 1.3.4. A **topological graph** is a multigraph in which no vertex has degree 2. \triangle

Note that though the vertex in a loop has degree 2, we include the single loop on a vertex in the set of topological graphs. We give some examples of topological graphs in Figure 1.3.2.

The following is an interesting result related to topological graphs and gives some insight into why this project focuses primarily on the study of topological graphs.

Theorem 1.3.5. *Every graph other than a cycle may be obtained by subdividing a topological graph.*

This theorem is true for every graph other than a cycle because a cycle is a subdivision of a vertex-free loop, which is not a graph.

We now define the notion of a subspace in a topological graph.

Definition 1.3.6. Let G and H be topological graphs. We say that H is a **subspace** of G if H is homeomorphic to some subgraph of some subdivision of G . \triangle

As we pointed out in Convention 1.1.2, we only consider connected graphs in this project. As a result, a topological graph G has only finitely many different graphs as subspaces.

The concept of a subspace in topological graphs is analogous to that of subgraphs for multigraphs or simple graphs. The definition of a subspace in topological graphs allows us to give a rigorous definition for a subquotient.

Definition 1.3.7. A **subquotient** G' of a graph G is a contraction of any graph which is a subspace of G . \triangle

The concept of a subquotient is similar to that of a minor for multigraphs and simple graphs, and just as we are able to define a minor closed property for multigraphs and simple graphs, we can define a subquotient closed property for topological graphs.

Definition 1.3.8. Consider some set S of topological graphs. S is said to be **subquotient closed** if any graph which is a subquotient of some graph in S is also in S . \triangle

Note the following proposition.

Proposition 1.3.9. *Let S be a minor closed set of multigraphs which is homeomorphism invariant. Then the set of all topological graphs in S is subquotient invariant.*

Subquotient closed properties work in much the same way as minor closed properties of simple graphs and multigraphs. However, once again, we will find a distinction between forbidden graph characterization of multigraphs and topological graphs, and we will go on

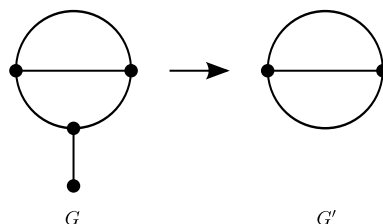


Figure 1.3.3: G' is the graph obtained from a topological edge contraction in G .

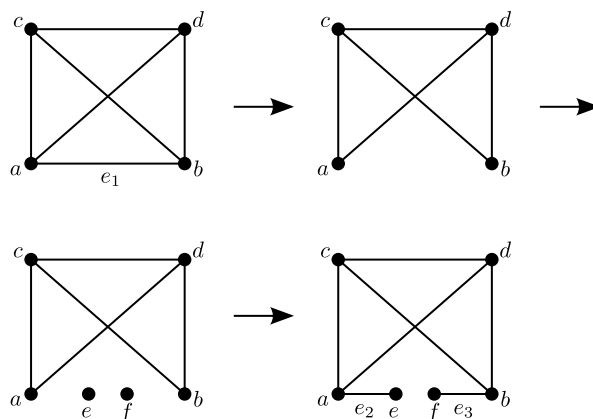
to describe this distinction in the following pages. First we state the Forbidden Subquotient Theorem for Topological Graphs [3].

Theorem 1.3.10 (The Forbidden Subquotient Theorem for Topological Graphs). *Let S be a subquotient closed set of topological graphs. Then there exists a finite number of graphs M_1, \dots, M_k such that for any graph G , we have $G \in S$ if and only if G does not contain any M_i as a subquotient.*

As with the analogous statement for simple graphs and multigraphs, we have a notion, then, of forbidden subquotients in relation to any subquotient closed property in topological graphs. However, before we define a forbidden subquotient, we will first give the basic operations to obtain a subquotient from some given topological graph.

The first operation **edge contraction** is defined exactly as in Definition 1.2.4, where we define edge contraction for multigraphs. However, it is important to note that edge contraction in topological graphs may involve the additional step of removing any bivalent vertices that result from an edge contraction of a given topological graph. Consider Figure 1.3.3, where we contract an edge from G to obtain G' . Observe that we have removed the edge subdivision of the lowest edge of G' which resulted from the edge contraction in G . The removal of this subdivision makes G' topological.

We now define an edge cut, which differs from the notion of edge deletion that we had defined for simple graph and multigraph minors.

Figure 1.3.4: Cutting edge e_1 .

Definition 1.3.11. Consider edge e_1 in Figure 1.3.4. We cut edge e_1 by the following steps:

1. Remove edge e_1 .
2. Add vertices e and f .
3. Draw edge e_2 connecting vertices a and e and draw edge e_3 connecting vertices f and b .
4. Remove any bivalent vertices that result from the above steps.

This gives us an **edge cut** of e_1 . △

Note that the graph obtained by a sequence of edge cuts and edge contractions of a topological graph is a subspace of the original graph. The definition of edge contractions and edge cuts enables us to state the following theorem, which gives an algorithm to obtain a subquotient of some given topological graph [3].

Theorem 1.3.12. *Let G be a topological graph. Then H is a subquotient of G if and only if H can be obtained from G by a sequence of edge contractions and edge cuts.*

As we had noted earlier, the concept of a subquotient is similar to the concept of a minor for multigraphs and simple graphs. However, since the graph operations for obtaining a subquotient of a given topological graph are different from those used for obtaining multigraph and simple graph minors, forbidden graph characterization for a given subquotient closed set also varies from forbidden graph characterization in simple graph or multigraph closed sets. We now give a definition for a forbidden subquotient for a given subquotient closed property, and will follow the definition with some basic definitions and a discussion on minimality in forbidden subquotients.

Definition 1.3.13. Let S be a subquotient closed set of graphs. We say that G is a **forbidden subquotient** for S if G is not in S , and every subquotient of G other than itself is in S . △

As with forbidden multigraph minors and forbidden simple graph minors, we also have the notion of minimality in forbidden subquotients. The following theorem gives us a simple test for checking for minimality in forbidden subquotients [3].

Theorem 1.3.14. *Let S be a subquotient closed set of topological graphs. Then G is a forbidden subquotient for S if and only if for every edge e of G that can be cut, the graph G' resulting from an edge cut of e is in S , and for any edge e that can be contracted, the graph G' resulting from the contraction of e is in S .*

We now give an example of a subquotient closed property and its corresponding forbidden subquotient set.

Example 1.3.15. Let S be the set of graphs with at most one vertex having degree 3 or greater. S is subquotient closed. The forbidden subquotient set for S is given in Figure 1.3.5 ◇

In Example 1.2.13 from the previous section, we had shown forbidden minor sets for multigraph pseudotrees and simple graph pseudotrees, and we had obtained different resulting sets of graphs. Similarly, there is a complicated relationship between forbidden multigraph minors and forbidden subquotients for equivalent closed properties, and our goal in the following discussion is to describe a way of finding this relationship.

In Chapter 3 of this project, we will explore a subquotient closed property in the context of topological graphs. At the end of the Chapter, we will explore this same property in the context of multigraphs. The following discussion will help guide our research.

First we define a **pendant edge** to be an edge in a graph which has at least one vertex with degree 1. We now give the definition of a pendant edge link [3].

Definition 1.3.16. Let G be a topological graph with some pendant edge e . We obtain a topological graph G' from G by a **pendant edge link** which is defined as follows:

1. Attach the degree 1 vertex of e to another vertex in G .
2. If the above operation results in the creation of a bivalent vertex, then eliminate this vertex. \triangle

The following theorem gives us an algorithm for obtaining a forbidden multigraph minor from a given forbidden subquotient for some equivalent closed property [3].

Theorem 1.3.17. *Let S be a minor closed set of multigraphs which is homeomorphism invariant. Every forbidden multigraph minor for the set can be obtained from a forbidden subquotient by a sequence of pendant edge links.*

Below, we explain how Theorem 1.3.17 is used to obtain forbidden multigraph minors from forbidden subquotients.

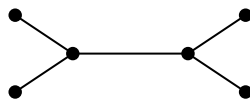


Figure 1.3.5: The forbidden subquotient set for the set of topological graphs with at most one vertex having degree 3 or greater.

Let S be a set of minor closed graphs which is homeomorphism invariant, and let F be the set of forbidden subquotients for S . Let $G_1, \dots, G_k \in F$. Then we can obtain the forbidden multigraph minor set for S' as follows:

1. Obtain a set of graphs $M' = G'_1, \dots, G'_k$ by applying all possible pendant edge links to each G_i .
2. Obtain a minimal set M from M' by deleting any graph in M' which is a minor of another.
3. Delete any bivalent vertices that occur in any graphs in M . △

Example 1.3.18. Consider once again the subquotient closed set S from Example 1.3.15. S is also minor closed, and we give the complete forbidden multigraph minor set for S in Figure 1.3.6. We obtained this set by applying the algorithm from Theorem 1.3.17 to the graph in Figure 1.3.5. ◇

Note that in this example, the set of forbidden multigraph minors was larger than the set of forbidden subquotients. For a certain case of the minor closed property that we study in this project, the forbidden multigraph minor set was more complicated than the forbidden subquotient set, and this was one of the motivations for us to focus largely on topological graphs. At the end of Chapter 3, we derive lists of forbidden multigraph minors from the forbidden subquotients we find in the first few sections of the chapter.

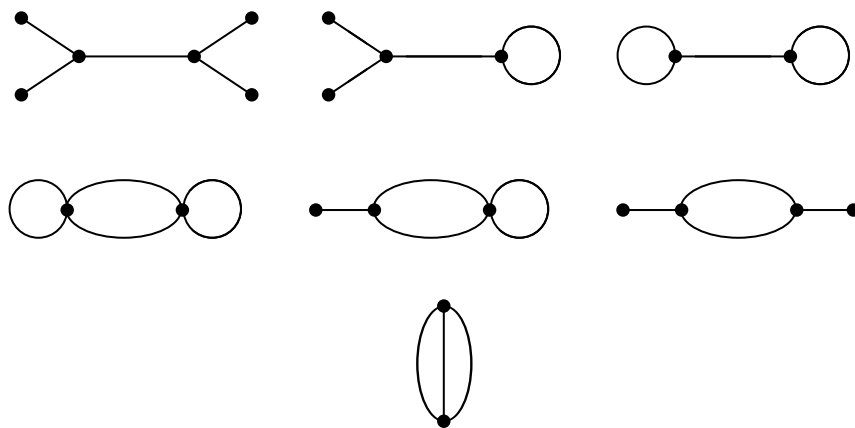


Figure 1.3.6: The forbidden multigraph minor set for the set of multigraphs with at most one vertex having degree 3 or greater.

2

The Robots-Target Game

The robots-target game is a pursuit game played on a finite, connected topological graph. In Section 2.1, we will define the game and demonstrate with examples the winning conditions for the robots and the targets. In Section 2.2 we will define **robot number**, which is a fundamental concept that we will use throughout the project. We conclude with some interesting results on complete graphs, the wheel graph and the bipyramid graph.

2.1 Defining the Game

In this section, we rigorously define the fundamental concepts of this game, which include the motions of players on the game, and what it means for either the robots or the target to win on a given graph.

We first define a continuous motion on a graph.

Definition 2.1.1. A **continuous motion** on a graph G is any continuous function $F: [0, \infty) \rightarrow G$. \triangle

The game is played by k robots and one target, which move continuously on the graph.

Definition 2.1.2. Let G be a graph. Then:

1. A **k-robot motion** on G is a tuple (R_1, R_2, \dots, R_k) of continuous motions on G .
2. A **target motion** T on G is a continuous motion on G . △

Unlike in many other pursuit games, there are no restrictions on the speed of the players.

The following definition explains how a continuous motion of robots or the target can result in either the robots catching the target or the target escaping the robots.

Definition 2.1.3. Let G be a graph and let (R_1, R_2, \dots, R_k) be a k -robot motion and T a target motion on G .

1. We say that the robots **catch** the target if $R_i(t) = T(t)$ for some $t \in [0, \infty)$ and some $i \in \{1, \dots, k\}$.
2. We say that the target **escapes** the robots if $R_i(t) \neq T(t)$ for all $t \in [0, \infty)$ and all $i \in \{1, \dots, k\}$. △

We use Definition 2.1.3 to define winning conditions for the robots and the target respectively. We give these definitions below.

Definition 2.1.4. Consider a graph G . The **robots win** on G if there exists some k -robot motion (R_1, R_2, \dots, R_k) such that for all target motions T on G , the robots catch the target. △

Definition 2.1.5. Consider a graph G . The **target wins** on G , if for all k -robot motions (R_1, R_2, \dots, R_k) , there exists some target motion T such that the target escapes the robots. △

By the above definitions, it is clear that the target wins on a graph if there is some target motion that evades the robots. The robot strategy on a graph is predetermined and the target is allowed to know the robot strategy beforehand. As a result of the robot

strategy being predetermined, it is convenient to think of the target as being invisible to the robots.

We now present some examples of the game which will illustrate how this game is played on certain graphs.

Example 2.1.6. Consider the game shown in Figure 2.1.1. In this game, we have one robot and one target on a claw graph. In the top half of this figure, we represent robot motions, and in the bottom half we add the motions of the target. Note that the figure shows integer time, but k -robot motion and target motion are continuous. This figure shows the first 11 units of time of robot motion and target motion on a graph but $t \in [0, \infty)$, which means that we are only capturing **snapshots** of robot motion and target motion in this time.

The robots start at the central vertex at $t = 0$, and move continuously along the top left edge at $t = 1$. The robot arrives at the top left vertex at $t = 2$, and then moves continuously back along the top left edge at $t = 3$. At $t = 4$, the robot arrives at the central vertex, and then moves continuously along the bottom edge at $t = 5$. At $t = 6$, the robot reaches the bottom vertex, and at $t = 7$, it moves once again continuously along the bottom edge. At $t = 8$, the robot reaches the central vertex, and at $t = 9$, it moves continuously along the top right edge. At $t = 10$, it reaches the top right vertex, and at $t = 11$, it moves continuously back onto the top right edge towards the central vertex.

The target motion, shown at the bottom, can be described as follows: the target starts at $t = 0$ at the bottom vertex of the graph, and remains there at $t = 1$. At $t = 2$, it moves continuously along the bottom edge, and at $t = 3$, it arrives at the top right vertex. It remains at this vertex at $t = 4$, and then moves continuously along the top right edge at $t = 5$. At $t = 6$, it moves continuously along the top left edge, and remains moving continuously along this edge at $t = 7$. At $t = 8$, it arrives at the top left vertex, and remains

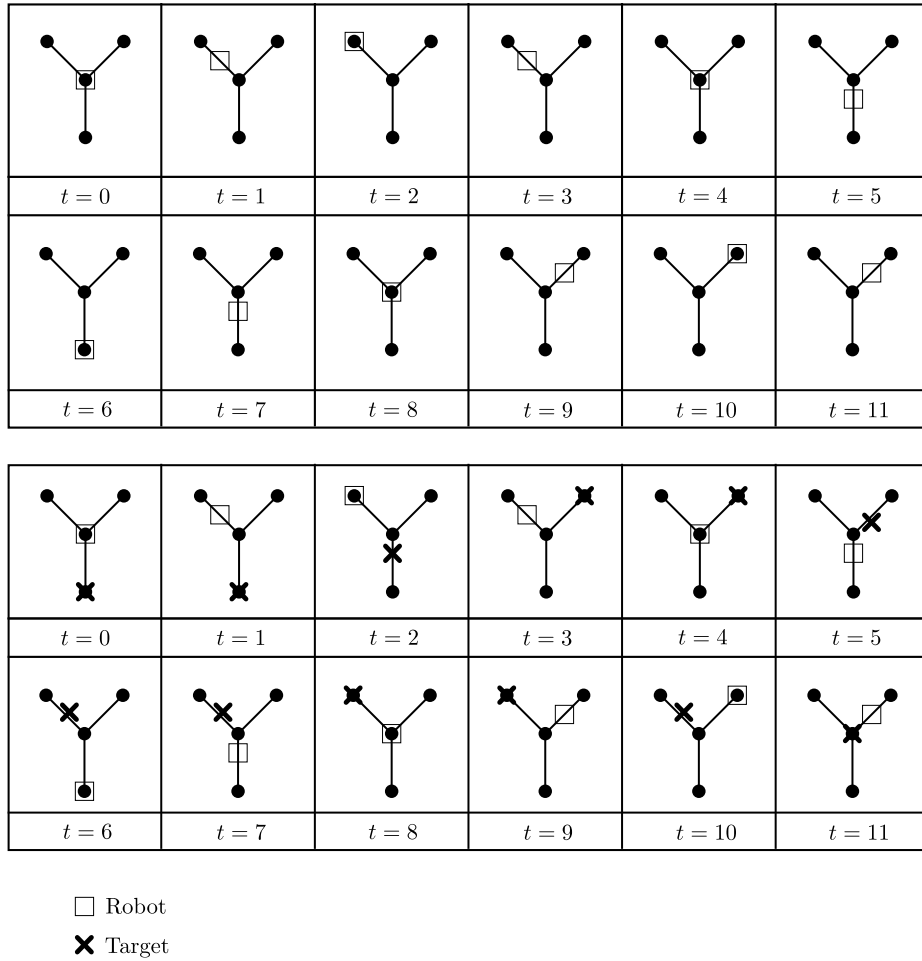


Figure 2.1.1: A game with one robot and the target on the claw graph.

there at $t = 9$. It moves continuously once again along the top left edge at $t = 10$, and arrives at the central vertex at $t = 11$. ◇

Note that this game does not end at $t = 11$, but we know that the robots do not win between $t = 0$ and $t = 11$. However, in Section 3.1 of the following chapter, we will rigorously prove that one robot does not catch the target on a claw graph, and we will give the winning strategy for the target.

The game shown in Figure 2.1.2 is organized in the same way as the game in Figure 2.1.1, but in this game, we have two robots on the claw graph instead of one. In this case, the

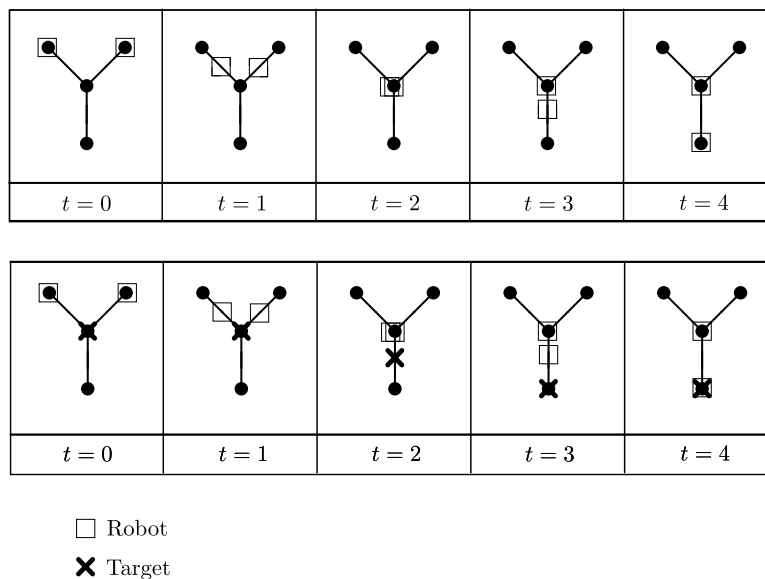


Figure 2.1.2: A game with two robots and the target on the claw graph.

robot catches the target and wins by time $t = 4$ and we can see clearly that the robot has a winning strategy.

We will now discuss the notion of cleared space, which is a recurring concept in this project. We begin by defining an uncleared space.

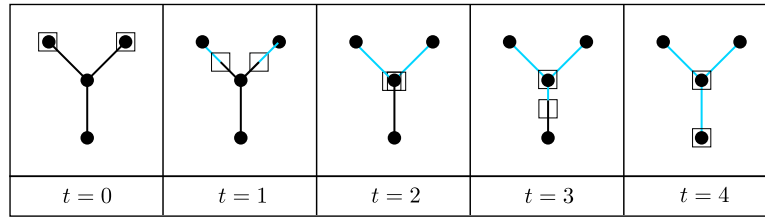
Definition 2.1.7. Let G be a graph and let (R_1, \dots, R_k) be a k -robot and $t \in [0, \infty)$.

Uncleared space at time t is the set of all possible places on G where we can find an uncaught target. △

We now define a cleared space on a graph.

Definition 2.1.8. The complement of an uncleared space on a graph G at time t is called a **cleared space** of G . △

A subspace S of some graph is said to have been cleared by the robots if S was traversed by the robots in such a way that once traversed, S is no longer accessible to the target,



□ Robot

Figure 2.1.3: Two robots clearing a graph.

that is, the target cannot enter S through any vertices or edges. Therefore, a cleared space and an uncleared space must always be separated by a robot in order for the cleared space to remain cleared.

Note that the robots win on a graph G if and only if they can clear all of G .

We illustrate the notions of cleared space in the following example.

Example 2.1.9. In Figure 2.1.3, we re-visit the claw graph game from Example 2.1.2. Cleared space in this graph is shown in blue. At $t = 0$, the robots start at the top right and the top left vertices respectively, and the entire graph is uncleared. At $t = 1$, each of the robots moves continuously on one of the top edges, clearing these edges in the process. At $t = 2$, they reach the central vertex, and all of the top two edges become cleared. One of the robots then begins to move continuously down the bottom edge at $t = 3$, and the entire graph becomes cleared at $t = 4$. \diamond

Figure 2.1.4 shows another example of how two robots clear a graph.

2.2 Robot Number

A fundamental concept in this project is that of robot number. We rigorously define robot number in this section and show that it is minor closed. At the end of this section, we will

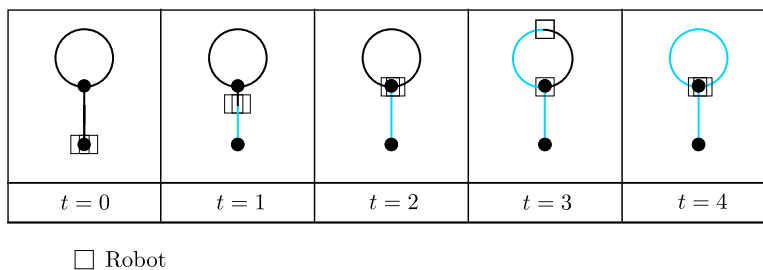


Figure 2.1.4: Two robots clearing a graph.

show some results relating robot number to the complete graphs, the wheel graph and the bipyramid graph.

Definition 2.2.1. Let G be a graph. The **robot number** $r(G)$ is the minimum number of robots required to catch the target on G . △

If a graph G has $r(G) = m$, then G is a forbidden minor for a graph with $r(G) = m - 1$.

Below we give a proposition related to robot number.

Proposition 2.2.2. *Let G be a finite graph on n vertices. Then $r(G)$ is finite.*

Proposition 2.2.2 is fairly straightforward. If we have some graph G on n vertices, then $n + 1$ robots can catch the target. The robots win by having all but one of the robots sit at a vertex on the graph. Then have the additional robot travel from one vertex to another, until it clears the last uncleared edge. This implies that the robot number of a graph must be finite.

We will now introduce forbidden minor theory in the context of the robots–target game by showing that robot number is minor closed.

Theorem 2.2.3. *Let S be the set of graphs on which k robots win against the target. S is multigraph minor closed.*

Proof. Let G be a graph on which k robots can catch the target. We are able to obtain a multigraph minor of G by contracting and deleting edges. We will argue that the statement holds true in both these cases.

Consider two vertices u and v in a graph with some edge E between them. Suppose a robot is traversing edge E . If we contract edge E , we obtain vertex uv . The robot that was previously on edge E will now position itself at vertex uv . This ensures that the winning strategy on graph G which included edge E is preserved when an edge is contracted since the target's inability to traverse vertices u and v is sustained since the robot now occupies vertex uv , and clearly edge E no longer exists.

Suppose now that instead of contracting edge E , we delete it and suppose also that doing so does not disconnect G . We are left with two vertices u and v with no edge between them. Since the robot can no longer travel from u to v directly, the robot will instead choose the shortest path from u to v . This strategy preserves the robot's ability to win because the target, who was previously unable to traverse edge E continues to be unable to do so since edge E no longer exists. At the same time, the robots continue to shrink the target's navigable space within the graph by taking the next shortest path from u to v .

We can conclude that any edge contractions or edge deletions on a graph G where the robots can catch the target will preserve the robot's ability to win. \square

Note that since $r(G)$ is a homeomorphism invariant property, it follows that it is also subquotient closed by Proposition 1.3.9. Note also that by Proposition 1.2.14, it is minor closed for simple graphs.

Note the following convention, which we will use throughout this project.

Convention 2.2.4. We will use the terminology “forbidden subquotients for k robots” to mean the forbidden subquotients for the set of topological graphs on which k robots win.

We show a basic result on robot number below.

Theorem 2.2.5. *Let graph G be a path. Then $r(G) = 1$.*

Proof. We first show that one robot has a winning strategy on G . Have the robot start at one end of the path and travel to the other end. This motion clears all of G .

It is trivially true that $r(G) \neq 0$ for G , since 0 robots cannot have a winning strategy on any graph. □

We now give a result on robot number for the complete graph K_n .

Theorem 2.2.6. *Let K_n be a complete graph. Then n robots can win on K_n .*

Proof. Consider a complete graph on n vertices. We will show that n robots can win on K_n . We will first show that n robots have a winning strategy on K_n . Have all n robots start at some vertex v_1 . Then have one of the n robots sit at v_1 and have the remaining $n - 1$ robots travel to the $n - 1$ adjacent vertices to v_1 along the edges connecting these $n - 1$ vertices to v_1 . With all paths leading to v_1 cleared and blocked from access by the target, the robot that remained on v_1 is now free to travel. Therefore, this robot can clear the graph and traverse all the edges of the graph.

This shows that n robots have a winning strategy on K_n . □

Note that we have only shown an upper bound for the robot number of K_n and in order to show a specific robot number, we would need to prove minimality.

A **wheel graph** W_n is defined as a graph on n vertices consisting of an $(n - 1)$ -cycle and a central vertex to which every vertex in the $(n - 1)$ -cycle is connected. We give the robot number for W_n in the following theorem.

Theorem 2.2.7. *Let W_n be a wheel graph. Then $r(W_n) = 4$.*

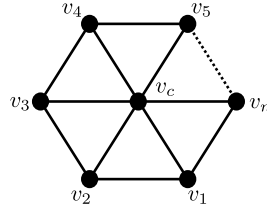


Figure 2.2.1: The wheel graph has robot number 4.

Proof. First we will show that four robots have a winning strategy on W_n . Consider W_n as shown in Figure 2.2.1.

We have vertices $\{v_1, v_2, \dots, v_n\}$ in W_n . We will refer to our four robots as Robot 1, Robot 2, Robot 3 and Robot 4. Have all four robots begin at some outer vertex v_1 . Have Robot 1 remain at v_1 throughout the game. Have Robot 2 travel to the central vertex v_c . Then have Robot 3 and Robot 4 travel to v_2 . Have Robot 3 remain at v_2 , and have Robot 4 travel to v_c , clearing edge v_2v_c . Then have Robot 3 move to v_3 , clearing edge v_2v_3 , and have Robot 4 also travel to v_3 , clearing edge v_cv_3 . The robots will follow the same strategy for each consecutive v_i until they have cleared edges v_nv_1 and v_cv_1 and returned to vertex v_1 . This gives a winning strategy for the robots on W_n .

Now we will show minimality. We know that W_n has K_4 as a subquotient. By Theorem 3.4.3, which will be given in Chapter 3, we know that 3 robots cannot catch the target on K_4 . This shows minimality.

We can conclude that $r(W_n) = 4$. □

Note that the strategy used in the proof of Theorem 2.2.7 is not the fastest strategy for the robots to win on the wheel graph.

A **bipyramid graph** B_n is defined as a graph on n vertices consisting of an $(n-2)$ -cycle and two additional vertices, each of which is adjacent to every vertex in the $(n-2)$ -cycle. We give an upper bound for robot number for the bipyramid graph.

Theorem 2.2.8. *Let B_n be a bipyramid graph. Then five robots can win on B_n .*

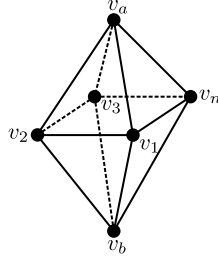


Figure 2.2.2: Five robots win on the bipyramid graph.

Proof. Consider the bipyramid graph shown in Figure 2.2.2, with cycle vertices $\{v_1, \dots, v_n\}$ and additional vertices v_a and v_b . We will show a winning strategy for the robots on B_n . Let us refer to the robots as Robot 1, Robot 2, Robot 3, Robot 4 and Robot 5. Have Robot 1 start at vertex v_a , and have Robot 2 start at vertex v_b . These robots will remain at these vertices throughout the game. Have Robot 3, Robot 4 and Robot 5 start at vertex v_1 . Have Robot 3 remain at vertex v_1 throughout the game. Have Robot 4 travel to vertex v_a and then back to vertex v_1 , clearing the edge between these two vertices. Then have Robot 4 travel to vertex v_b and then back to vertex v_1 , clearing the edge between these two vertices. Then have Robot 4 and Robot 5 travel together to vertex v_2 , clearing edge $v_1 v_2$. Then have Robot 5 remain at vertex v_2 , and once again have Robot 4 travel to vertices v_a and v_b , clearing the edges between these vertices and v_2 . Robot 4 and Robot 5 proceed along the $(n - 2)$ -cycle in B_n in this way, with Robot 4 clearing the edges between the vertices of the $(n - 2)$ -cycle and vertices v_a and v_b , until they have cleared edges $v_n v_a$, $v_n v_b$ and $v_n v_1$ and have returned once again to vertex v_1 . This strategy clears all of B_n .

We can conclude that five robots win on the bipyramid graph.

□

3

Forbidden Subquotients and Forbidden Minors

In this chapter, we give some of the main results of this project. These results are related to the concept of forbidden graph characterization that we had introduced in Chapter 1.

In the previous chapter, we showed that robot number is a subquotient closed property. This result is fundamental to the research in this chapter, and allows us to characterize the forbidden graph sets for each of robot number 1, robot number 2 and robot number 3. In this chapter, we will go on to sequentially give forbidden graphs for each of these robot numbers. We give a complete set of forbidden graphs for $r(G) = 1$ and $r(G) = 2$, and a partial set for $r(G) = 3$.

We had briefly discussed the relationship between forbidden subquotients and forbidden minors in Chapter 1, and will extend the research in this chapter to exploring the relationship of these characterizations in the context of robot number.

3.1 The Complete Set of Forbidden Subquotients for $r(G) = 1$

In this section, we study the forbidden subquotients for robot number 1. There are exactly two such subquotients, the claw graph and the loop, which are shown below in Figure 3.1.1.

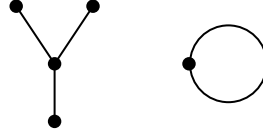


Figure 3.1.1: These are the forbidden subquotients for $r(G) = 1$.

We first prove that the claw graph and the loop are indeed forbidden subquotients for $r(G) = 1$, and in Theorem 3.1.3, we will conclude that these are the only forbidden subquotients for $r(G) = 1$.

We begin by proving that the claw graph is a forbidden subquotient for robot number 1.

Theorem 3.1.1. *Let G be a graph. The claw graph is a forbidden subquotient for one robot.*

Proof. We will first show that the target has a winning strategy on the claw graph. The claw graph consists of three edges attached at a vertex. We mark each edge of the claw graph as 1, 2 and 3 as shown in Figure 3.1.2. By Theorem 2.2.5, we know that one robot is necessary to clear an edge, so the robot on the claw graph will check one edge of the graph at a time. Let $e_1, e_2, e_3, \dots \in \{1, 2, 3\}$ be the sequence of edges visited by the robot, where $e_i \neq e_{i+1}$ for each i . Then the target strategy to win is as follows:

1. Start in any end other than e_1 .
2. Subsequently, when the robot arrives at e_i , the target moves to an edge other than e_i or e_{i+1} .

Note that the target is able to move between any edge in the claw graph by traveling through the central vertex of the graph when a robot is checking an edge. Also note that when a robot remains on some edge, the target remains in the edge that it is currently in. This shows a winning strategy for the target.

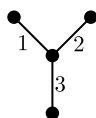


Figure 3.1.2: We mark the edges of the claw graph.



Figure 3.1.3: Loop with one edge cut.

Now, we will show that this graph is minimal. When we contract one of the pendant edges from the claw graph, the graph that remains is a path. By Theorem 2.2.5, we can see that the robot will catch the target on such a graph.

We can see that any edge cut on this graph will disconnect the graph, so we do not consider this operation to check minimality.

We can conclude that the claw graph is a forbidden subquotient for one robot. \square

We will now show that the loop is a forbidden subquotient for one robot.

Theorem 3.1.2. *Let G be a graph. The loop is a forbidden subquotient for one robot winning on G .*

Proof. We will first show that the target has a winning strategy against one robot on this graph. The target wins simply by traveling in the same direction as the robot, at a distance of half the circumference from the robot.

We will now show that G is minimal. Contracting an edge in the loop produces a single vertex. It is trivially true that one robot will catch the target on a vertex.

Now, cut an edge in the loop. This gives us a path as shown in Figure 3.1.3. By Theorem 2.2.5, the robot catches the target.

We can conclude that the loop is a forbidden subquotient for one robot. \square

We will now show that the complete set of forbidden subquotients for $r(G) = 1$ consists of the claw graph and the loop.

Theorem 3.1.3 (The loop and the claw graph are the only forbidden subquotients for $r(G) = 1$). *Let G be a graph. Then the following statements are equivalent.*

1. *One robot wins on G .*
2. *G does not contain either the claw graph or the loop as a subquotient.*
3. *G is a path.*

We begin by proving Lemma 3.1.4 and Lemma 3.1.5.

Lemma 3.1.4. *A graph does not have the claw graph as a subquotient if and only if all its vertices have degree less than or equal to 2.*

Proof. Consider some graph X where every vertex has degree less than or equal to 2. We will show that X must not contain the claw graph as a subquotient. Suppose that X contains the claw graph as a subquotient. Then by Theorem 1.2.18 there must exist some subgraph X' of X such that the claw graph is a contraction of X' . Then, by Theorem 1.2.19 X' must be a subdivision of the claw graph. However since the claw graph has a single vertex which has degree 3, that means that there must be at least one vertex in X' which has degree 3. Since X' is a subgraph of X , it must be the case that X must also contain at least one vertex with degree 3 or greater. This is a contradiction.

Now for the other direction, consider a graph Y which does not contain the claw graph as a subquotient. We will show that every vertex in Y must have degree less than or equal to 2. Suppose that there exists a vertex in Y with degree 3. Then, Y must contain the claw graph as a subquotient. This is a contradiction.

We can conclude that a graph does not have the claw graph as a subquotient if and only if all its vertices have degree less than or equal to 2. □

Lemma 3.1.5. *A graph does not have the loop as a subquotient if and only if it is a tree.*

Proof. First, consider some tree T . We will show that T does not have the loop as a subquotient. Suppose that T has the loop as a subquotient. Then by Theorem 1.2.18 there must exist some subgraph T' of T such that the loop is a contraction of T' . By Theorem 1.2.19, this must mean that the loop is topologically the same graph as T' . This implies that T' is a simple cycle. Since T' is a subgraph of T , we get that T must contain a simple cycle. By definition, this is a contradiction.

Now, for the other direction, consider some graph S which does not have the loop as a subquotient. We will show that S is a tree. Suppose that S is not a tree. Then S must contain a simple cycle. This is a contradiction.

We can conclude that a graph does not have the loop as a subquotient if and only if it is a tree. □

We now go on to prove Theorem 3.1.3

Proof of Theorem 3.1.3. (1) \implies (2) From Theorem 3.1.1 and Theorem 3.1.2, we have that if G contains the claw graph or the loop as a subquotient, then one robot does not win on G .

We can conclude that (1) \implies (2).

(2) \implies (3) Consider first the claw graph. Then, by Lemma 3.1.4, the subquotient closed property characterizing the claw graph is that all vertices have degree less than or equal to 2. The set satisfying this property is the set of paths and polygons. Now consider the loop. By Lemma 3.1.5, the subquotient closed set characterizing the loop is all trees. If a graph G does not contain either the claw graph or the loop, then it must satisfy the intersection of the subquotient closed properties characterizing the claw graph and the loop. This intersection is all paths.

We can conclude that (2) \implies (3).

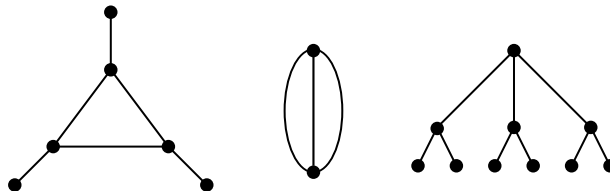


Figure 3.2.1: The net graph, triple edge and the $(3, 2)$ -tree are forbidden subquotients for $r(G) = 2$.

(3) \implies **(1)** One robot wins on a path by Theorem 2.2.5.

Therefore **(3)** \implies **(1)**.

This proves our theorem and we can conclude that the only forbidden subquotients for one robot against the target are the claw graph and the loop. \square

We will use the main results from this section extensively in our study of the forbidden subquotients for $r(G) = 2$. The characterization of the claw graph and the loop as the only forbidden subquotients for $r(G) = 1$ imply that at any instance of either of these graphs in another larger graph, the number of robots required to clear the graph would be at least 2. The following section will capture this idea in more detail.

In addition, it is worth noting the procedure for obtaining the forbidden subquotients for a given robot number and subsequently the approach used for showing a complete set.

3.2 Net Graph, Triple Edge and the $(3, 2)$ -Tree

In this section, we introduce three graphs, the net graph, triple edge and the $(3, 2)$ -tree. These graphs are shown in Figure 3.2.1. We will show in the following section that these graphs are the only forbidden subquotients for $r(G) = 2$.

Before we go on to describe these three graphs in detail, we give an important lemma. Recall that we prove that a given graph is a forbidden subquotient for some $r(G)$ by first

showing a winning strategy for the target on G and then checking for minimality. In this section, we will use the test given in the lemma below to check for minimality.

Lemma 3.2.1. *Let G a topological graph which is obtained as follows:*

1. *Start with a path.*
2. *Make some edges double.*
3. *Attach some pendant edges and loops.*

Then two robots have a winning strategy on G .

Proof. Consider some topological graph G which is obtained as specified in the statement of the lemma. We will show that two robots can win on G . The robots will start at one end of G , and move together towards the other end of G , progressively clearing the graph as they do so. On any part of this graph representing a path with no pendant edges, loops or double edges, the robots can simply clear the space between vertices by traveling along edges. When the robots encounter a vertex v from which arises one or all of a loop, several loops, a pendant edge or several pendant edges, one robot sits at the vertex v , while the other robot checks and clears each of the loops and pendant edges arising from v . This ensures that previously cleared space in the graph remains cleared while new space is being cleared. Once this new space is cleared, the robots move further along the graph. When the robots encounter a vertex u from which arises a double edge to the next vertex w , they will split up at u and travel one along each edge until they both converge at w . When they reach vertex w , all the previously cleared space on G will remain cleared.

The robots can employ this strategy to travel along and clear every space in G until they have traveled from one end of the graph to another. □

Lemma 3.2.1 gives us a simple test for checking for minimality based on whether a given graph has the form described in the statement of the lemma.

Note that we will refer to the starting path in a graph G which is obtained as described in Lemma 3.2.1 as a **spine**.

We now discuss the forbidden subquotients for robot number 2.

Theorem 3.2.2. *Let G be a graph. The net graph is a forbidden subquotient for two robots.*

Proof. First we will show that the target has a winning strategy on the net graph. We divide the graph into three subspaces as shown in Figure 3.2.2, and we mark these subspaces as 1, 2 and 3. Note that each of these subspaces is a claw graph, and each claw is attached to another at the end of a pendant edge. By Theorem 3.1.1, we know that each of these subspaces is a forbidden subquotient for one robot. Clearly, then, two robots will be needed to clear any of these three subspaces. Let $c_1, c_2, c_3, \dots \in \{1, 2, 3\}$ be the sequence of claws visited by two robots, where each $c_i \neq c_{i+1}$. The target strategy is as follows:

1. The target starts in any claw other than c_1 .
2. Subsequently, when the robots arrive at c_i , the target moves to a claw other than c_i or c_{i+1} .
3. If the target is in a claw with a robot, then the target would use the strategy described in Theorem 3.1.1.

Note that since each claw is connected to another at one place, the target is always able to move between claws. We can conclude that the target has a winning strategy on the net graph.

We will now show minimality. We have three maximal subquotients for the net graph, N' , which has the only edge cut possible on the net graph, N'' , which has one of the edges of the triangle of the net graph contracted, and N''' , which is obtained from contracting

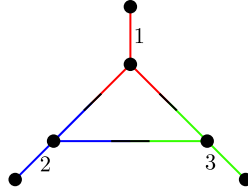


Figure 3.2.2: The net graph is divided into three subspaces.

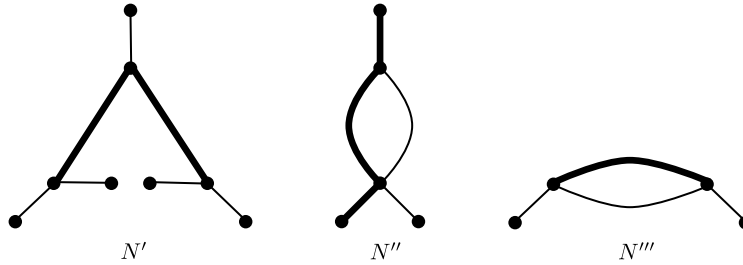


Figure 3.2.3: We show spines in N' , N'' and N''' .

one of the pendant edges of the net graph. In Figure 3.2.3, we show a spine in each of these graphs. By Lemma 3.2.1, these pictures suffice to show minimality.

We can conclude that the net graph is a forbidden subquotient for two robots. \square

Theorem 3.2.3. *Let G be a graph. The triple edge is a forbidden subquotient for two robots.*

Proof. We will first show that the target has a winning strategy against two robots on the triple edge. First note that we say that **an edge has two robots in it** either if both robots are in the interior of some edge or if one of the robots is in the interior of some edge and the other is at an endpoint of that edge. We mark the edges of the triple edge as 1, 2 and 3 as shown in Figure 3.2.4. Let $\{e_1, e_2, e_3, \dots\} \in \{1, 2, 3\}$ be the sequence of edges with two robots in it. The target is aware of what this sequence is and its strategy is to always stay in the edge other than the next two edges that will have two robots in them. We define a **shelter** on some edge to be a subspace of an edge. Each edge has two shelters, as shown in Figure 3.2.4. The target is always in either one of these shelters on an edge.



Figure 3.2.4: We denote the edges of the triple edge as 1, 2 and 3. The subspaces of the graph which are shown to be thickened are shelters.

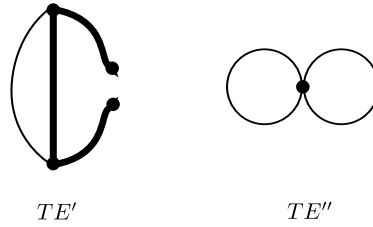
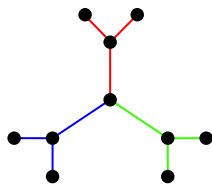
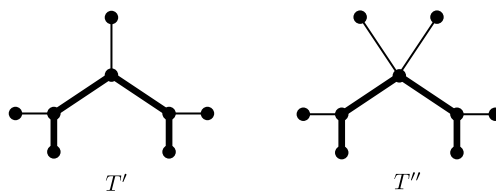


Figure 3.2.5: We show spines in TE' and TE'' .

When one robot enters an edge from the side that is closer to the shelter that the target is in, then the target moves to the other shelter on the edge. This is possible because the target always chooses to be in an edge which does not have two robots in it, so the path between one shelter and another on an edge that has the target in it is always robot-free. When the robot which has entered the edge on which the target is in moves toward the further shelter that the target has moved to, the target moves to the edge other than the next two edges which will have two robots in them. This is the winning strategy for the target to indefinitely evade two robots on a triple edge.

We will now show minimality. Figure 3.2.5 shows the two maximal forbidden subquotients that we can obtain from the triple edge, TE' and TE'' . We show a spine in TE' , and the spine in TE'' is simply a vertex. By Lemma 3.2.1, we can infer that the triple edge is indeed minimal.

We can conclude that the triple edge is a forbidden subquotient for two robots. \square

Figure 3.2.6: The $(3, 2)$ -tree is divided into three subspaces.Figure 3.2.7: We show spines in T' and T'' .

Theorem 3.2.4. *Let G be a graph. The $(3, 2)$ -tree is a forbidden subquotient for two robots winning on G .*

Proof. We will first show that the target has a winning strategy on the $(3, 2)$ -tree. We divide the $(3, 2)$ -tree into three subspaces as shown in Figure 3.2.6. It is not hard to see that the $(3, 2)$ -tree consists of three claw graphs attached to one another at a vertex each. By Theorem 3.1.1, we know that two robots are required to clear a claw graph at a time. The winning strategy for the target on this graph is to always be in a claw which is not currently being cleared by two robots or the claw which will be cleared by the robots next. The target is able to move from one claw to another at any time by traveling through the central vertex. This shows that the target has a winning strategy on the $(3, 2)$ -tree.

We now show minimality. Consider the graphs T' and T'' in Figure 3.2.7. These are the maximal forbidden subquotients of the $(3, 2)$ -tree, and we show spines in these graphs in the figure. By Lemma 3.2.1, we have minimality.

We can conclude that the $(3, 2)$ -tree is a forbidden subquotient for two robots. \square

3.3 The Complete Set of Forbidden Subquotients for $r(G) = 2$

In Section 3.2, we had introduced the net graph, triple edge and the $(3, 2)$ -tree and showed that these are forbidden subquotients for robot number 2. In this section, we will go on to show that these are indeed the only forbidden subquotients for robot number 2.

The following theorem summarizes this result.

Theorem 3.3.1 (The net graph, triple edge and $(3, 2)$ -tree are the only forbidden subquotients for $r(G) = 2$). *Let G be a topological graph. Then the following statements are equivalent.*

1. *Two robots can win on G .*
2. *G does not have the net graph, triple edge or the $(3, 2)$ -tree as a subquotient.*
3. *G can be obtained as follows:*
 - *Start with a path*
 - *Make some edge double*
 - *Attach some pendant edges and loops*
 - *Subdivide some edges*

Before proving Theorem 3.3.1, we need to prove the following lemmas.

Lemma 3.3.2. *Let G be a topological graph. Then G does not have the net graph as a subquotient if and only if G can be obtained as follows:*

1. *Start with a tree.*
2. *Add loops and multiple edges.*
3. *Subdivide.*

Proof. First, consider some graph G which does not have the net graph as a subquotient.

Suppose first that G is a cycle. G may be obtained by subdividing an edge.

Suppose now that G is not a cycle. Unsubdivide G to obtain H . This removes all bivalent vertices. Now simplify H to obtain T . It will suffice to show that T is a tree.

Suppose, to the contrary that T has a cycle. Let v be a vertex in the cycle. We claim that the degree of v inside G is at least 3. Note that the degree of v inside T is at least 2 since v is in a cycle. Since there are no bivalent vertices in H , the degree of v in H must be at least 3. This implies that the degree of v inside G must be at least 3, and we can conclude that G must have the net graph as a subquotient. This is a contradiction.

Now, for the other direction, let G be a graph which is obtained as in the enumerated list from the statement of the lemma. We will show that G must not contain the net graph as a subquotient. Suppose that G contains the net graph as a subquotient. Then by Theorem 1.2.18, there must exist some subgraph G' of G such that the net graph is a contraction of G' . Then, by Theorem 1.2.19 the net graph must be homeomorphic to G' . We know that every simple cycle in G must have at most two vertices with degree 2 or greater. The net graph has three such vertices. This is a contradiction. \square

Lemma 3.3.3. *Let G be a graph. Then G does not have the net graph or triple edge as a subquotient if and only if G can be obtained as follows:*

1. *Start with a tree.*
2. *Add loops and double edges.*
3. *Subdivide.*

Proof. Consider some graph G . Suppose that G does not contain the net graph or the triple edge as a subquotient. Suppose that G is not a cycle. Since we have that the net graph is not a subquotient, there exists a tree T and a graph H so that H is obtained from T by adding multiple edges and loops and G is a subdivision of H . Then, if H had the

triple edge, then this would mean that G , a subdivision of H would also have the triple edge, a contradiction. \square

Lemma 3.3.4. *Let G be a graph. Then G does not have the $(3,2)$ -tree as a subquotient if and only if G can be obtained as follows:*

1. *Start with a path.*
2. *Add loops and double edges.*
3. *Add pendant edges.*
4. *Subdivide.*

Proof. Consider some graph G which is not a cycle. Unsubdivide G to obtain H , removing all bivalent vertices. Now remove all pendant edges to obtain K . By Lemma 3.3.3, there exists some tree T so that K is obtained from T by adding double edges and loops. Let v be a vertex in T . Suppose that the degree of v inside T is at least 3. Now pick some vertex w which is adjacent to v . Then, the degree of w inside H must be at least 2, since if it were 1, then it would be removed in K . However, we know all subdivisions are removed in H , which gives us that the degree of w inside H must be at least 3. We know that the degree of w inside H is the same as the degree of w inside G . We can conclude that the degree of w inside G is at least 3 and that G contains the $(3,2)$ -tree. \square

We now prove Theorem 3.3.1

Proof of Theorem 3.3.1. (1) \implies (2) Let G be a graph. Suppose that two robots win on G . By Theorem 3.2.2, Theorem 3.2.3 and Theorem 3.2.4 we have that the net graph, triple edge and $(3,2)$ -tree are forbidden subquotients for two robots winning. Clearly, then, G must not contain the net graph, triple edge or the $(3,2)$ -tree as a subquotient. We can conclude that (1) \implies (2).

(2) \implies (3) Consider a graph G which does not contain the net graph, the triple edge or the $(3, 2)$ -tree as a subquotient. By Lemma 3.3.2, Lemma 3.3.3 and Lemma 3.3.4, we can conclude that (2) \implies (3).

(3) \implies (1) Let G be a graph which is obtained as specified in the statement of the theorem. It follows from Lemma 3.2.1 that two robots have a winning strategy on G .

We conclude that the only forbidden subquotients for $r(G) = 2$ are the net graph, triple edge and the $(3, 2)$ -tree. \square

3.4 Forbidden Subquotients for $r(G) = 3$

In this section, we give a partial list of forbidden subquotients for three robots. So far, there are 21 graphs in this list. We give proofs for five of these graphs, and show all 21 graphs that we have found in Figure 3.4.5.

We begin by showing that a graph obtained by attaching three $(3, 2)$ -trees in a certain configuration gives us a set of forbidden subquotients for three robots. The proof of the theorem is followed by a related conjecture.

Theorem 3.4.1. *Let G be a graph which is obtained as follows:*

1. *Start with a claw.*
2. *At the degree 1 vertex of each pendant edge in the claw, do either one of the following:*
 - *Attach the $(3, 2)$ -tree at a degree 1 vertex of a pendant edge of the $(3, 2)$ -tree.*
 - *Attach the $(3, 2)$ -tree at its central vertex.*

Then G is a forbidden subquotient for three robots.

Proof. We will first show a winning strategy for the target to escape the robots on a graph G which is obtained as specified in the theorem. By Theorem 3.2.4 we know that the $(3, 2)$ -tree is a forbidden subquotient for two robots. This means that at least three

robots are required to clear any $(3, 2)$ -tree that was attached to the claw in Step 2 of the statement of the theorem. Let us call the $(3, 2)$ -trees 1, 2 and 3. Let $t_1, t_2, t_3, \dots \in \{1, 2, 3\}$ be the sequence of the $(3, 2)$ -trees that three robots will visit at any time. Suppose $t_i \neq t_{i+1}$ for each i . The target is aware of what this sequence is, and escapes the robots by always being in the $(3, 2)$ -tree other t_i and t_{i+1} for every i . Because the central vertex of the claw in G is robot-free every time three robots clear a $(3, 2)$ -tree, the target is able to move freely from one $(3, 2)$ -tree to another. This gives us the target strategy to evade three robots on G .

We show all the maximal minors of G in Figure 3.2.4. Minimality for each of these graphs was checked and the proof of minimality is omitted here.

We can conclude that G is a forbidden subquotient for three robots.

□

We give the following conjecture related to graph G obtained in Theorem 3.4.1

Conjecture 3.4.2. *The set of all graphs M obtained in the fashion described in Theorem 3.4.1 is the set of all forbidden subquotients for three robots that are trees.*

We will now show that K_4 is a forbidden subquotient for three robots.

Theorem 3.4.3. *K_4 is a forbidden subquotient for three robots.*

Proof. We will first show that the target has a winning strategy against three robots on K_4 . Divide K_4 into four subspaces, 1, 2, 3 and 4, as shown in Figure 3.4.2. Observe that each of these subspaces is homeomorphic to the claw graph, and that each pair of subspaces is incident at exactly one place. Now, by Theorem 3.1.1, we know that a pair of robots will need to be on any one claw at a time to clear it. We have three robots on this graph, so a pair of robots will check a claw at a time, while the other robot will be in another claw. Let $(C_1, K_1), (C_2, K_2), \dots \in \{1, 2, 3, 4\}^2$ be a sequence of ordered pairs,

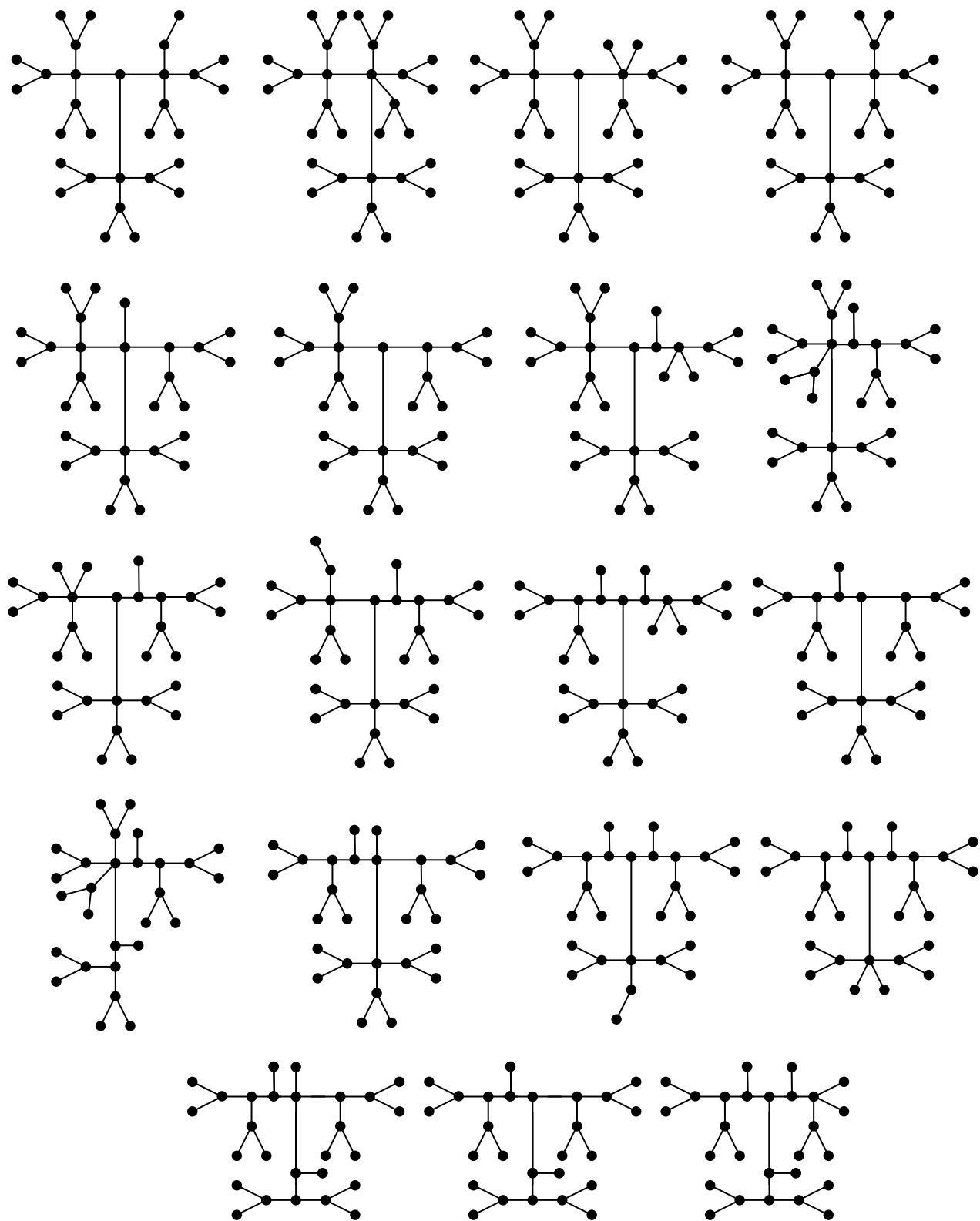


Figure 3.4.1: These are the maximal minors of G obtained in Theorem 3.4.1.

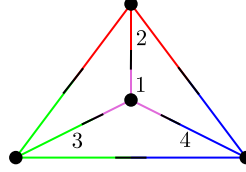


Figure 3.4.2: We divide K_4 into four subspaces.

where each C_i is the claw that a pair of robots check at the same time and each K_i is the claw that the third robot is in at the time. We know that the target can evade one robot in a claw, and that it is only in danger in the situation where two robots are in a claw. Therefore, we note that in this sequence of ordered pairs, we do not consider any situations where each robot is on a different claw. Assume that $(C_i, K_i) \neq (C_{i+1}, K_{i+1})$ for each i . The target strategy is as follows:

1. The target starts at some claw other than C_1 and K_1 .
2. Subsequently, when two robots arrive at (C_i, K_i) , there are two possible cases:
 - Case 1:** If (C_i, K_i) and (C_{i+1}, K_{i+1}) consist of three claws, then the target will move to the claw which does not have a robot in it.
 - Case 2:** If (C_i, K_i) and (C_{i+1}, K_{i+1}) consists of all four claws, then the target moves to K_i and starts evading one robot on the claw using the strategy prescribed in Theorem 3.1.1. The target is easily able to evade one robot on K_i in this situation because there is only one robot on the claw and the target has access to any two branches at a time.

We will now show minimality. For simplicity, we will refer to the three robots as Robot 1, Robot 2 and Robot 3.

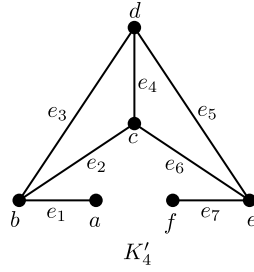


Figure 3.4.3: Three robots have a winning strategy on K'_4 .

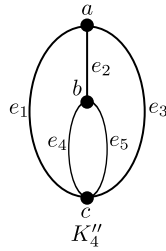


Figure 3.4.4: Three robots have a winning strategy on K''_4 .

Consider K'_4 in Figure 3.4.3 with one of the edges of K_4 cut. We will show a winning strategy for three robots on K'_4 . Have all three robots start at a . Then have them travel together to b , clearing e_1 . Then have Robot 1 and Robot 2 travel to d , clearing e_3 , and have Robot 3 travel to c , clearing edge e_2 . Have Robot 1 travel to vertex c , clearing edge e_4 , and have Robot 2 travel to vertex e , clearing edge e_5 . Have Robot 3 travel to vertex e , clearing edge e_6 . Then have Robot 3 travel to vertex f , clearing edge e_7 . This strategy clears all of K'_4 .

Now consider K''_4 shown in Figure 3.4.4, where we have contracted one of the outer edges of K_4 . Have all three robots start at vertex a . Have Robot 1 travel to vertex c , clearing edge e_1 . Have Robot 2 travel to b , clearing edge e_2 . Have Robot 3 travel to c , clearing e_3 . Have Robot 1 travel to b , clearing e_4 and have Robot 3 travel to b , clearing e_5 . This strategy clears the whole graph.

We can conclude that K_4 is a forbidden subquotient for $r(G) = 3$. □

We conclude this section with Figure 3.4.5, which shows the 21 forbidden subquotients found so far for three robots. A winning strategy for the target has been found for each of these graphs except for the graph which we have highlighted with a box. We conjecture that such a strategy exists for this graph, but do not have a rigorous proof to show this. Minimality has been checked for each of these graphs.

3.5 Forbidden Minors

In Chapter 1, we introduced the relationship of forbidden graph characterization in multigraphs, simple graphs and topological graphs. In this chapter, we explore this relationship as it pertains to robot number. In Subsections 3.5.1 and 3.5.2, we derive the complete list of forbidden multigraph minors for robot numbers 1 and 2 from the forbidden subquotients that we found in Section 3.1 and Section 3.3. In Subsection 3.5.3, we derive the complete list of the forbidden simple graph minors for robot numbers 1 and 2 based on the forbidden multigraph minors we obtained.

3.5.1 The Forbidden Multigraph Minors for $r(G) = 1$

In this subsection, we will give the forbidden multigraph minors for $r(G) = 1$ and $r(G) = 2$ based on our findings in Sections 3.3 and 3.3.

We will apply Theorem 1.3.17 which was stated in Section 1.3 to obtain forbidden multigraph minors from forbidden subquotients.

We state the result of this subsection as a theorem below.

Theorem 3.5.1. *Let G be a multigraph. The forbidden multigraph minors for one robot winning on G are the claw graph and the loop.*

Proof. Recall from Section 3.1 that the only forbidden subquotients for $r(G) = 1$ are the loop and the claw graph. Clearly there are no pendant edges in the loop, which means that the only forbidden multigraph minor to be derived from the loop is itself.

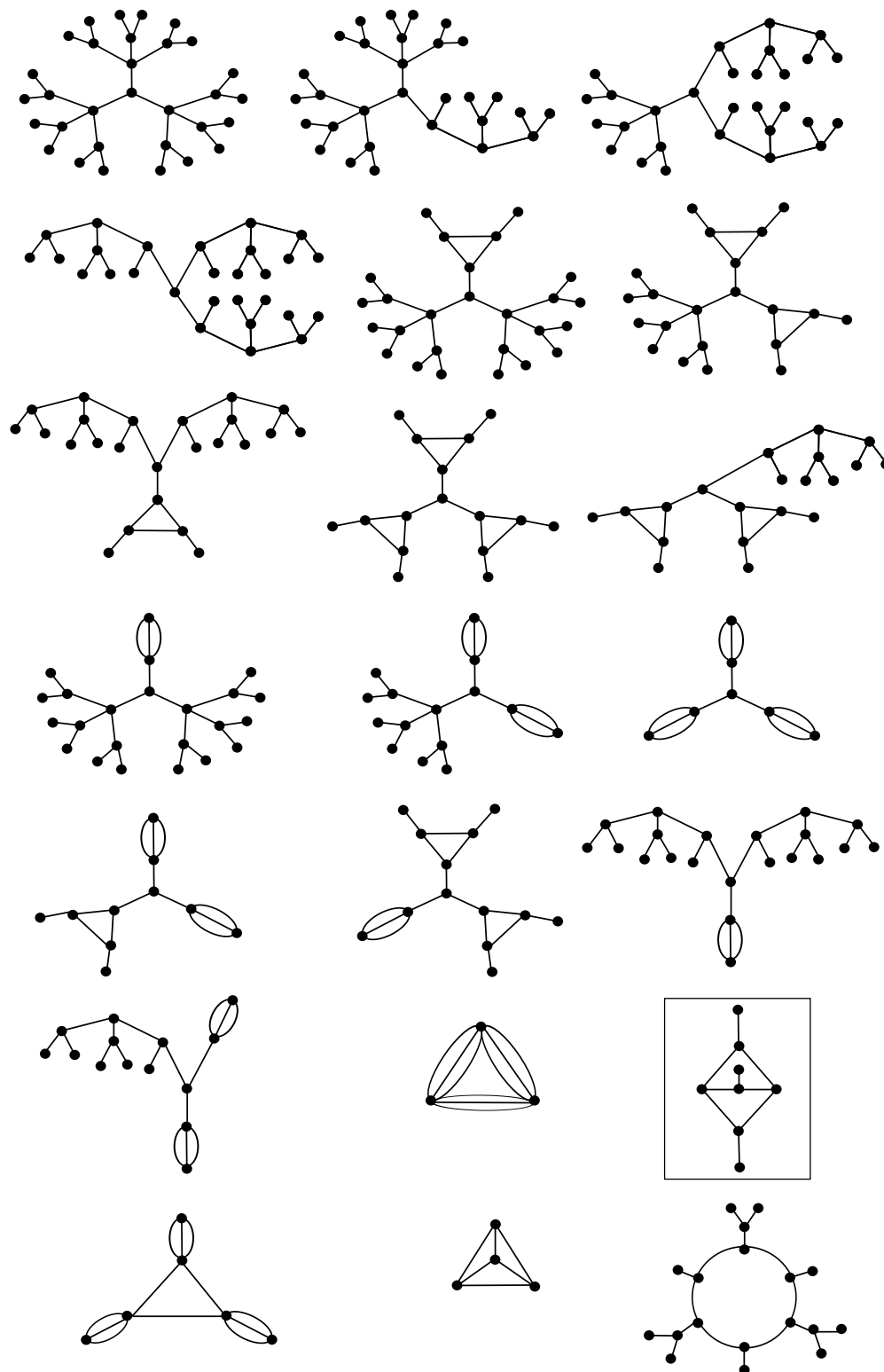


Figure 3.4.5: This is a partial list of forbidden subquotients for three robots.



Figure 3.5.1: This is the graph obtained from attaching two pendant edges of the claw graph at their degree 1 vertices.



Figure 3.5.2: This is the graph obtained from attaching a degree 1 vertex of a pendant edge in the claw graph to the central vertex.

Now, for the claw graph, we have three pendant edges attached at a vertex. In Figure 3.5.1, we show the result of a pendant edge link of the degree 1 vertex of a pendant edge to a degree 1 vertex of another pendant edge in the claw graph. Clearly, this creates a subdivided loop, or a double edge, which has the loop as a minor. This means that the graph in the figure is not minimal.

The only other pendant edge link possible is shown in the graph in Figure 3.5.2, where we attach the degree 1 vertex of one of the pendant edges to the central vertex of the graph. This creates a loop. Clearly, the graph in Figure 3.5.2 contains the loop as a minor, and is not minimal.

We can conclude that the loop and the claw graph are the only forbidden multigraph minors for $r(G) = 1$. Note that the forbidden multigraph minor set and the forbidden subquotient set coincide in this case, and we do not need to show minimality. \square

3.5.2 The Forbidden Multigraph Minors for $r(G) = 2$

In this subsection, we give the forbidden multigraph minors for $r(G) = 2$. Once again, we will use Theorem 1.3.17 as our algorithm for obtaining the required graphs. Recall from

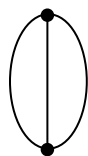


Figure 3.5.3: The triple edge is a forbidden multigraph minor.

Section 3.3 that the only forbidden subquotients for $r(G) = 2$ are the triple edge, net graph and the $(3, 2)$ -tree. Because forbidden multigraph minor characterization for these graphs is more complicated than for the graphs from the previous section, we will give an argument for each as a sequence of lemmas. We begin with the triple edge.

Theorem 3.5.2. *Let G be a multigraph. The forbidden multigraph minor set for two robots winning on G is the union of the sets of graphs in Figure 3.5.3, Figure 3.5.6, and Figure 3.5.10.*

Lemma 3.5.3. *The triple edge is a forbidden multigraph minor for two robots.*

Proof. Since there are no pendant edges in the triple edge, it is fairly straightforward to see that the only forbidden multigraph minor that can be derived from it is itself. Minimality was shown in Theorem 3.2.3.

We show the triple edge once again in Figure 3.5.3

□

We will now discuss the net graph.

Lemma 3.5.4. *The graphs show in Figure 3.5.6 are all of the forbidden multigraph minors for two robots that can be obtained from the net graph.*

Proof. Consider first the net graph. Clearly, one forbidden multigraph minor of the net graph is itself. Note that minimality was shown in Theorem 3.2.2 and we will not repeat the proof in this section. The net graph is shown as N_0 in Figure 3.5.6.

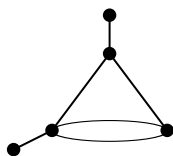


Figure 3.5.4: This is the graph obtained from attaching a degree 1 vertex of a pendant edge of the net graph to a vertex on the triangle other than the one from which the edge originates.

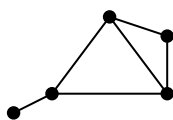


Figure 3.5.5: This is the graph obtained from attaching two pendant edges of the net graph to each other at their degree 1 vertices.

We will use Theorem 1.3.17 to proceed.

Observe that if we attach a degree 1 vertex of any of the pendant edges of the net graph to any of the vertices on the triangle other than the vertex from which that edge originates, then we obtain the graph shown in Figure 3.5.4. Clearly, this graph contains the triple edge as a minor, and we need not consider any further pendant edge links on it, as every subsequent link would contain the triple edge as a minor.

If we attach the degree 1 vertex of a pendant edge in the net graph to the degree 1 vertex of another pendant edge, we obtain a graph as shown in Figure 3.5.5. This graph also contains the triple edge as a minor, and we need not consider any further edge links on this graph as any subsequent edge link will produce a graph which continues to contain the triple edge as a minor.

Finally, we are able to attach the degree 1 vertex of each of the pendant edges in the net graph to the vertex of the edge's origin, forming a loop at the vertices of the triangle. There are three ways that we can have such a pendant edge link: we can have two pendant edges arising from each of two vertices on the triangle and a loop from the last, or we

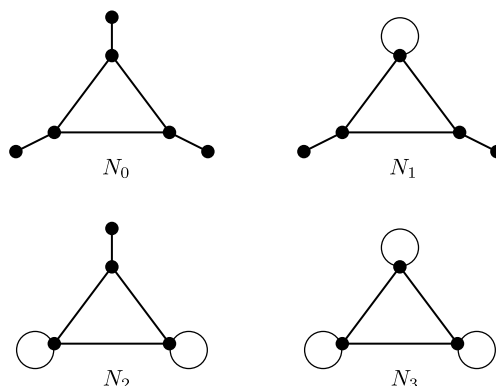


Figure 3.5.6: These are all the forbidden multigraph minors for $r(G) = 2$ that are obtained from linking pendant edges of the net graph.

can have two loops arising from each of two vertices on the triangle and a pendant edge from the last, or lastly, we can have a loop arising from each of the three vertices of the triangle. Two robots cannot win on a graph obtained in such a manner. The graphs N_1 , N_2 and N_3 in Figure 3.5.6 are of this form.

We can conclude that the graphs shown in Figure 3.5.6 are all the forbidden multigraph minors for two robots that can be obtained from the net graph.

□

Lastly, we consider the $(3, 2)$ -tree.

Lemma 3.5.5. *The graphs shown in Figure 3.5.10 are all of the forbidden multigraph minors for two robots that can be obtained from the $(3, 2)$ -tree.*

Proof. There are six pendant edges in the $(3, 2)$ -tree and 10 vertices, which means we need to consider a number of pendant edge links in this graph to obtain a minimal forbidden multigraph minor set for $r(G) = 2$.

Clearly, the $(3, 2)$ -tree must be a forbidden multigraph minor for two robots. Minimality for this statement was shown in Theorem 3.2.4 and we do not repeat it in this section.

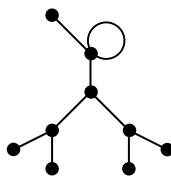


Figure 3.5.7: This is the graph obtained from attaching the degree 1 vertex of a pendant edge in the $(3, 2)$ -tree to its origin vertex.

Now, we try attaching the degree 1 vertices of two adjacent pendant edges and remove the subdivision from the resulting loop. We obtain graph G_4 which is shown in Figure 3.5.10. G_4 is a forbidden multigraph minor for two robots. We could attach two pairs of pendant edges of the $(3, 2)$ -tree, to obtain another forbidden multigraph minor, G_4 . Lastly, we could have all three pairs of pendant edges attached in this way, giving us G_6 , which is also a forbidden multigraph minor.

Next, we attach the degree 1 vertex of a pendant edge to its origin vertex, as shown in Figure 3.5.7. It is not hard to see that this graph contains the graph G_4 in Figure 3.5.10 as a minor. We need not consider any other pendant edge links on this graph.

Next, attach the degree 1 vertex of a pendant edge to the central vertex of the $(3, 2)$ -tree. This gives us graph G_1 shown in Figure 3.5.10. This graph is a forbidden multigraph minor for two robots. We can also attach the degree 1 vertex of another pendant edge to obtain G_2 , and we can attach the third pendant edge to the central vertex to obtain G_3 . Both of these are forbidden multigraph minors for two robots.

Next, we try to attach the degree 1 vertex of a pendant edge to one of the degree 3 vertices in the graph other than the central vertex or the origin vertex of the pendant edge. The resulting graph is shown in Figure 3.5.8, and it clearly has the net graph as a minor. We do not consider any further pendant edge links on this graph.

Lastly, consider the graph shown in Figure 3.5.9, which we obtained by attaching the degree 1 vertices of two pendant edges together that do not share an endpoint. It is easy

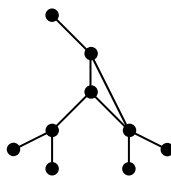


Figure 3.5.8: This is the graph obtained from attaching the degree 1 vertex of a pendant edge in the $(3, 2)$ -tree to some degree 3 vertex other than its origin vertex and the central vertex.

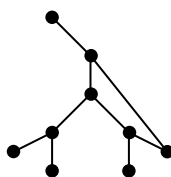


Figure 3.5.9: This is the graph obtained from attaching the degree 1 vertex of a pendant edge in the $(3, 2)$ -tree to the degree 1 vertex of a pendant edge with which it does not share an endpoint.

to see that this graph has the net graph as a minor, and we do not consider any other pendant edge links on this graph.

This proves that the graphs in Figure 3.5.10 form the set of forbidden multigraph minors for two robots on a graph.

□

Proof of Theorem 3.5.2. By Lemma 3.5.3, Lemma 3.5.4 and Lemma 3.5.5, we have that the graphs in Figure 3.5.3, Figure 3.5.6, and Figure 3.5.10 are forbidden multigraph minors for two robots.

Minimality is proved by showing a spine in each of the maximal minors of the graphs in Figure 3.5.3, Figure 3.5.6, and Figure 3.5.10. Minimality is shown in Figures 3.5.12, 3.5.13, 3.5.14, 3.5.15, 3.5.16, 3.5.17, 3.5.18 and in 3.5.19.

We can conclude that the forbidden multigraph minor set for two robots is the union of the sets of graphs in Figure 3.5.3, Figure 3.5.6, and Figure 3.5.10.

□

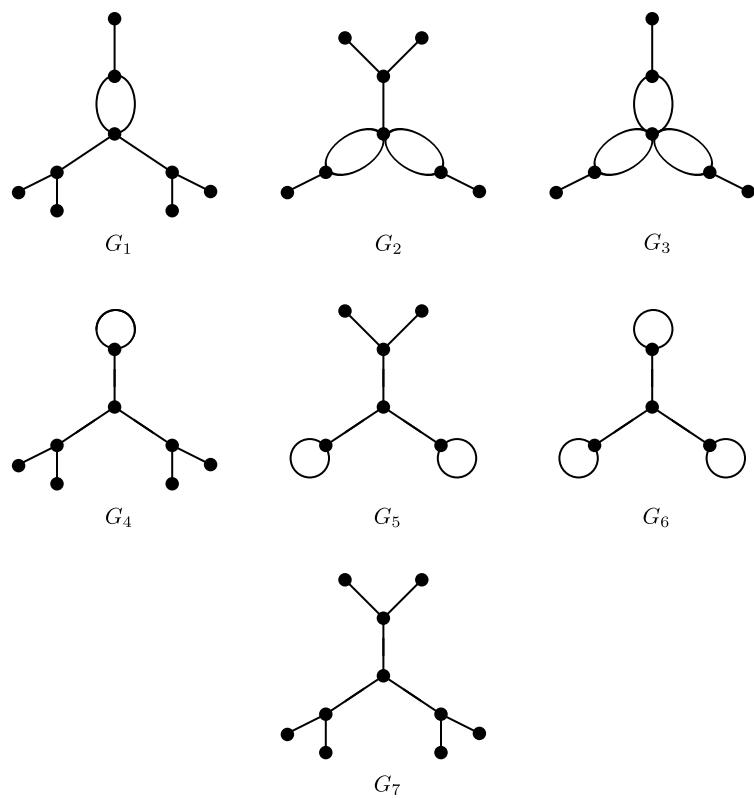


Figure 3.5.10: These are all the forbidden multigraph minors for $r(G) = 2$ that are obtained from linking pendant edges of the (3, 2)-tree.

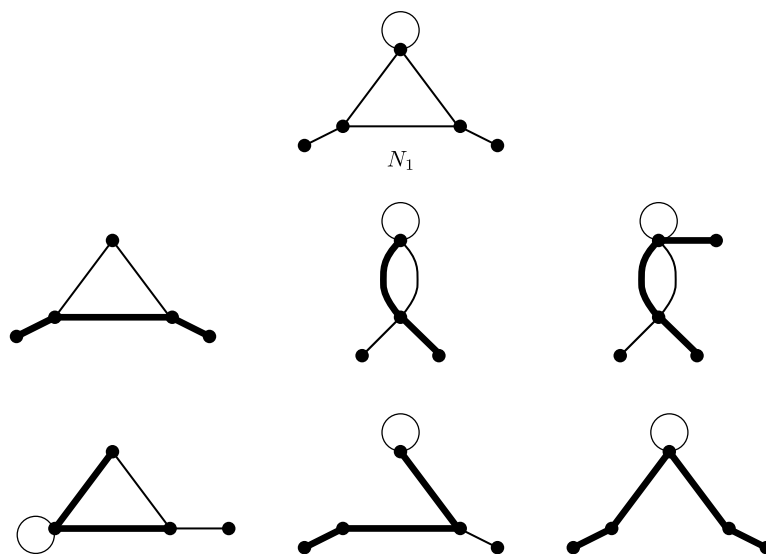


Figure 3.5.11: We show a spine in every possible maximal minor of N_1 .

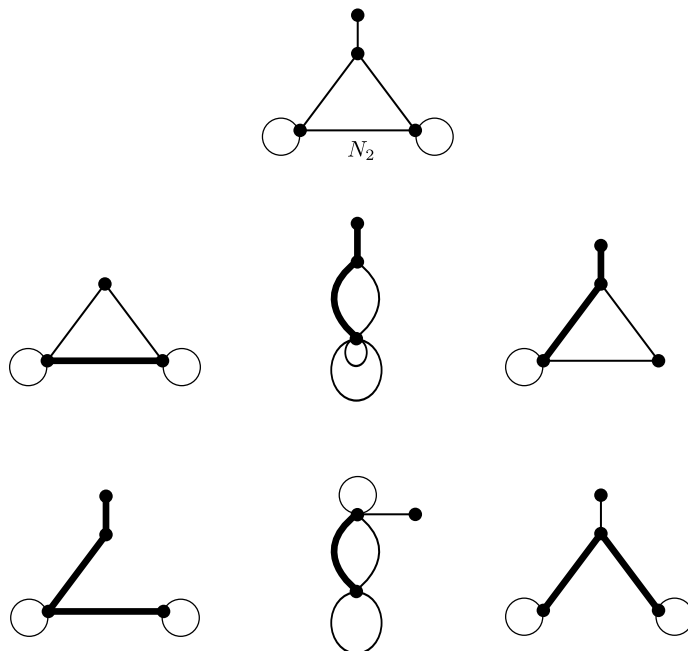


Figure 3.5.12: We show a spine in every possible maximal minor of N_2 .

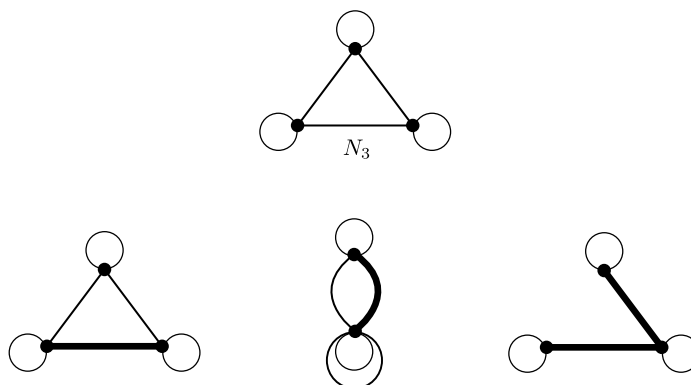


Figure 3.5.13: We show a spine in every possible maximal minor of N_3 .

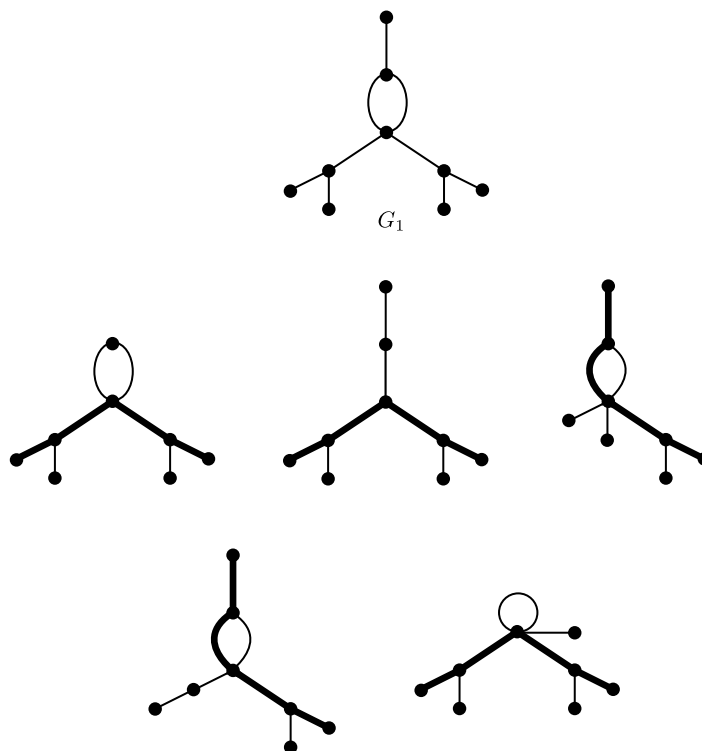


Figure 3.5.14: We show a spine in every possible maximal minor of G_1 .

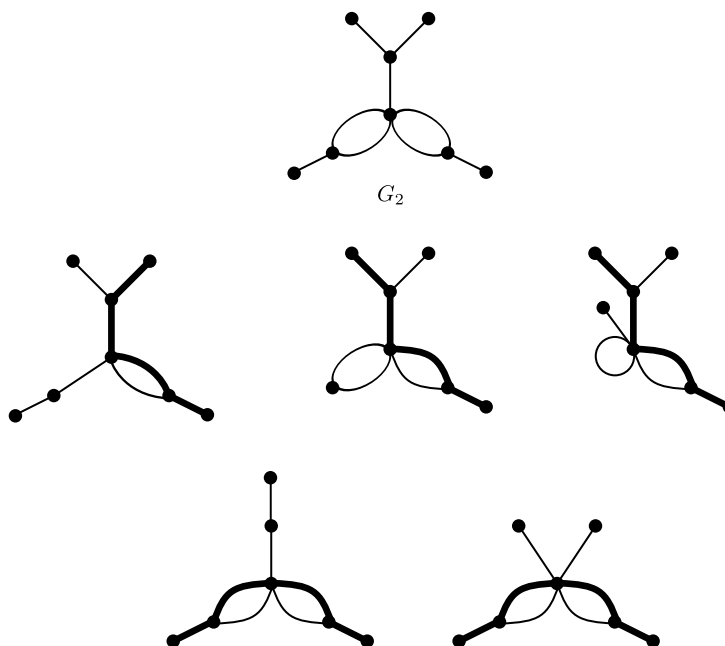


Figure 3.5.15: We show a spine in every possible maximal minor of G_2 .

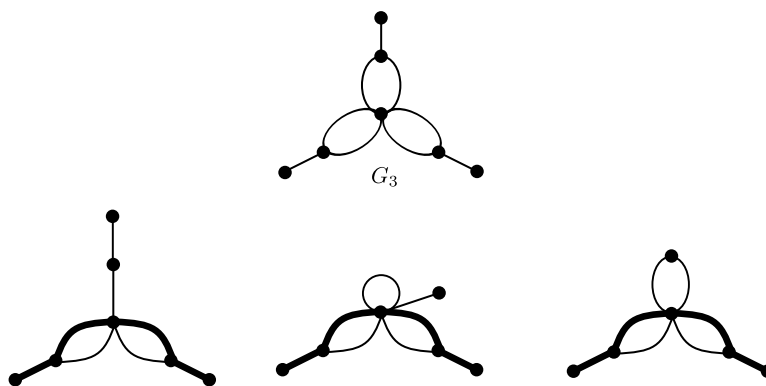


Figure 3.5.16: We show a spine in every possible maximal minor of G_3 .

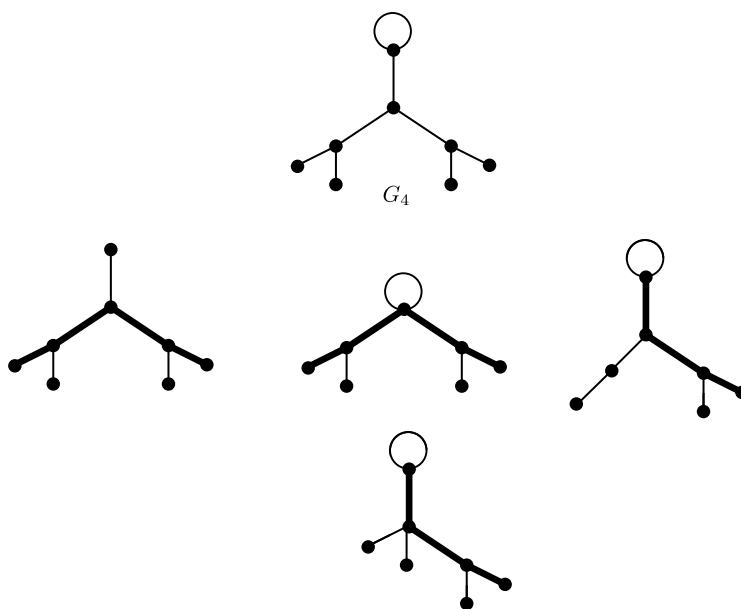


Figure 3.5.17: We show a spine in every possible maximal minor of G_4 .

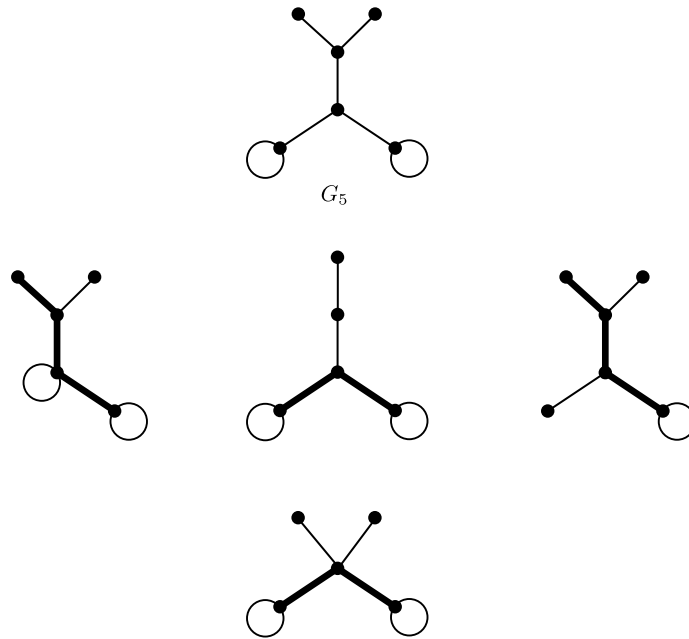


Figure 3.5.18: We show a spine in every possible maximal minor of G_5 .

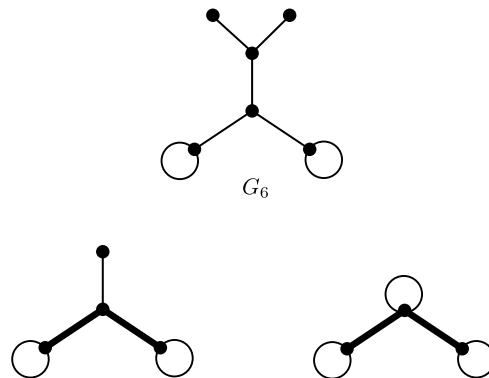


Figure 3.5.19: We show a spine in every possible maximal minor of G_6 .

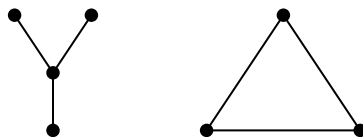


Figure 3.5.20: This is the complete set of forbidden simple graph minors for one robot.

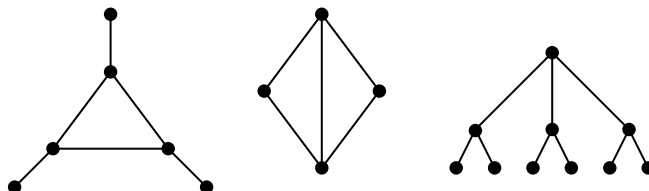


Figure 3.5.21: This is the complete set of forbidden simple graph minors for two robots.

3.5.3 Forbidden Simple Graph Minors

In Chapter 1, we had discussed how to obtain forbidden simple graph minors from a given set of forbidden multigraph minors. In particular, Theorem 1.2.17 gave us an algorithm with which we could obtain such a set. In this subsection, we give the forbidden simple graph minors for robot number 1 and robot number 2, which we obtained by checking minimality for the graphs obtained by a sequence of unsimplifying edge contractions on the graphs in Figure 3.5.3, Figure 3.5.6, and Figure 3.5.10 from the previous subsection.

Because the argument for showing how to obtain forbidden simple graph minors from forbidden multigraph minors is fairly involved, we suffice it to simply show the forbidden simple graph minors for $r(G) = 1$ and $r(G) = 2$.

The complete forbidden simple graph minor set for $r(G) = 1$ is given in Figure 3.5.20 and the complete forbidden simple graph minor set for $r(G) = 2$ is given in Figure 3.5.21.

4

Robot Number and Pathwidth

In this chapter, we will explore the concepts of path decomposition and pathwidth, and their relationship to robot number. Explorations into pathwidth were motivated by similar studies conducted by Tina Zhang in her senior project [10]. Zhang defined the notion of cop number which is analogous to robot number. We will refer to this notion as **Zhang cop number** in this chapter. Zhang gave a relationship between treewidth and the Zhang cop number which bounded Zhang cop number within a certain treewidth range. Since treewidth at most k is a minor closed property [1], this became a useful tool for Zhang to study forbidden minor theory as it pertained to the pursuit game she was studying. This was the primary motivation for us to explore pathwidth, and we will give the main result of our findings in Section 4.2.

Path decomposition was first defined by Bienstock, Robertson, Seymour and Thomas in 1989 [4], and went on to become one of the fundamental concepts used in the proof of the Forbidden Minor Theorem [7]. Pathwidth has also been studied in the context of pursuit games, and my research on pathwidth was guided by the research done by Barát, who studied the relationship of pathwidth to a pursuit game which, like the robots-target

game, involves one or more pursuers and an invisible target on a graph [2]. We will discuss Barát's game in greater detail in Section 4.2.

It should be noted that pathwidth is related only to simple graphs, and in finding a relationship between $r(G)$ and pathwidth, we will only consider the forbidden simple graph minors that we found in Section 3.5.3 of Chapter 3.

4.1 Path Decomposition and Pathwidth

In this section, we introduce the concepts of path decomposition and pathwidth.

Definition 4.1.1. Consider a graph G on n vertices. A **path decomposition** in G is some sequence of subsets V_i of vertices in G satisfying the following conditions:

1. Every edge in G has both endpoints in some V_i .
2. If $i \leq j \leq k$, then $V_i \cap V_k \subseteq V_j$.

△

Example 4.1.2. We illustrate the path decomposition of a graph G in Figure 4.1.1. The path decomposition shown in G is represented by the sequence of subsets V_i where $1 \leq i \leq 10$. Note that this path decomposition satisfies the conditions of Definition 4.1.1. For the first condition, clearly every edge in G is contained in some V_i . For the second condition, consider V_1 , V_2 and V_3 . We have here that $V_1 \cap V_3 \subseteq V_2$. Now consider V_1 , V_5 and V_{10} . We have that $V_1 \cap V_{10} = \emptyset$. Clearly, $\emptyset \subseteq V_5$. As such, for all $i \leq j \leq k$ in this path decomposition, $V_i \cap V_k \subseteq V_j$. ◇

We will now define the width of a path decomposition.

Definition 4.1.3. The **width** of a path decomposition is defined as the $\max_i |V_i| - 1$. △

Example 4.1.4. Consider graph G in Figure 4.1.2. In the same figure, we create graph W in which the nodes represent the sequence of subsets V_i of vertices which form a path

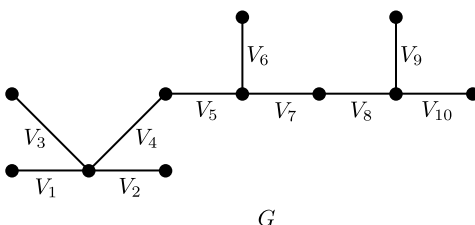


Figure 4.1.1: A path decomposition of G .

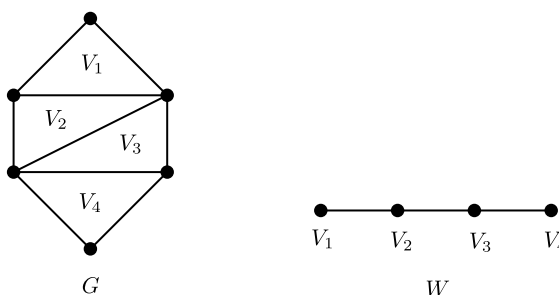


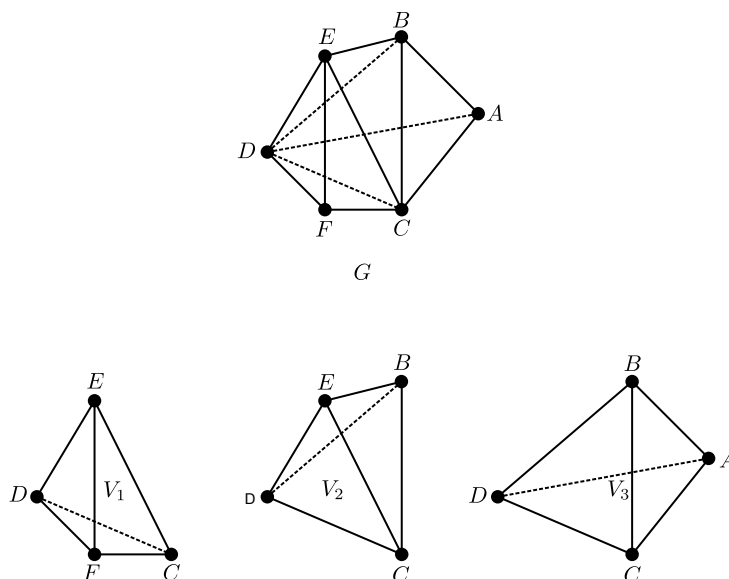
Figure 4.1.2: W is a representation of a path decomposition in G .

decomposition in G , where $1 \leq i \leq 4$. Once again, we can see that each V_i satisfies the conditions of Definition 4.1.1. Every edge in graph G is contained in some V_i . Consider the subsets of vertices in this path decomposition V_1 , V_2 and V_3 . We have $V_1 \cap V_3 = V_2$ and clearly $V_2 \subseteq V_2$. Then consider V_1 , V_3 and V_4 . We have that $V_1 \cap V_4 = \emptyset$. Clearly, $\emptyset \subseteq V_3$. Now, note that each V_i contains at most three vertices. By Definition 4.1.2, we can conclude that the width of G is 2. ◇

We give the definition of pathwidth below.

Definition 4.1.5. The **pathwidth** of a graph G is the minimum width of any path decomposition in G . △

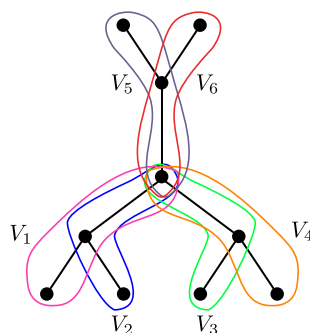
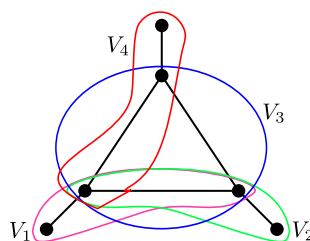
Example 4.1.6. Consider once again graph G in Figure 4.1.1. We showed a path decomposition of G where each V_i contained at most two vertices. The width of G is therefore 1, and the pathwidth of G is also 1. ◇

Figure 4.1.3: A path decomposition of G .

Example 4.1.7. Consider graph G in Figure 4.1.2. In Example 4.1.4 we showed that the width of G is 2. We observe that the pathwidth of G is also 2. \diamond

Example 4.1.8. Consider graph G in Figure 4.1.3. A path decomposition in G is shown, where V_i represents the sequence of subsets in the path decomposition and $1 \leq i \leq 3$. Observe that G is in fact three tetrahedra, $ABDC$, $EDCB$ and $EDFC$, each attached to the previous at one face exactly. Each edge of G is contained in some V_i in the path decomposition. Now consider V_1 , V_2 and V_3 . We have $V_1 \cap V_3 \subseteq V_2$, and that the two conditions of a path decomposition are satisfied. Now, note that each V_i has 4 vertices. This gives us width 3 and pathwidth 3. \diamond

Example 4.1.9. We will give the pathwidth of the $(3, 2)$ -tree which we had introduced in Section 3.2 of the previous chapter. A path decomposition of the $(3, 2)$ -tree is shown below in Figure 4.1.4. We will first show that this path decomposition satisfies the conditions of Definition 4.1.1. Each edge of the $(3, 2)$ -tree is contained in some V_i in the path decomposition. Now consider V_1 , V_2 and V_5 . We have $V_1 \cap V_5 \subseteq V_3$. In general, for all

Figure 4.1.4: A path decomposition of the $(3, 2)$ -tree.Figure 4.1.5: A path decomposition P of the net graph.

$i \leq j \leq k$ in this path decomposition, we have $V_i \cap V_k \subseteq V_j$. Now, note that the maximum size of each V_i in this path decomposition is 3, which gives us width 2 and pathwidth 2. \diamond

Example 4.1.10. Last, consider the net graph which we had also introduced in Section 3.2. We show a path decomposition of the net graph in Figure 4.1.5. We show another way of representing the same path decomposition in Figure 4.1.6, where we represent each V_i by drawing in imaginary lines. In the net graph, this gives us a path decomposition where each V_i is a set of vertices which form a triangle. Clearly, each edge of G is in some V_i where $1 \leq i \leq 4$. Now, consider V_1 , V_3 and V_4 . We have that $V_1 \cap V_4 \subseteq V_3$. In general, for all $i \leq j \leq k$ in this path decomposition, we have $V_i \cap V_k \subseteq V_j$. The maximum size of any V_i in this path decomposition is 3, which gives us width 2 and pathwidth 2. \diamond

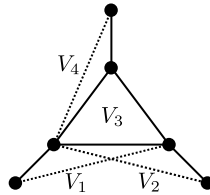


Figure 4.1.6: A different representation of path decomposition P of the net graph.

4.2 The Relationship Between Pathwidth and the Robots–Target Game

At the beginning of this chapter, we had noted that Zhang had given bounds for Zhang cop number using the notion of treewidth at most k , which is minor closed. Specifically, Zhang concluded that the two parameters are off by exactly 1 [10], which served as motivation for us to hypothesize that there is some analogous relationship between robot number and pathwidth.

Barát studies a game which is identical to the robots-target game other than a difference between the motions of the players on a graph. The pursuers in the Barát’s game are called the cops and the pursued is called the robber. Barát defines the notion of cop number for his game, which is analogous to robot number, and he concludes that it is within some range of pathwidth. This is the main result that we will use in our research in this section.

We will conclude this section with a conjecture that, if proved, could guide further research on forbidden minors for robot number.

Below we give a definition of the Barát Cops and Robber game, quoted from Barát’s own definition [2].

Definition 4.2.1 (Definition of the Barát Cops and Robber Game). Let a directed graph D be given. The robber stands on a vertex, and he can run at infinite speed to any other vertex along the directed edges in the indicated

direction. He is not permitted to run through a cop, however. The cops stand also on the vertices, and move by helicopters from vertex to vertex. Assume in this version, that the robber is invisible. The cops capture the robber, if a cop lands on a vertex where the robber stands and cannot move anywhere i.e. all out-neighbors are also occupied by cops. The goal is to decide how many cops are necessary to capture the robber. Denote this minimum by $cn(D)$. Here cn stands for cop number, overline for the invisible case. \triangle

Barát's game is similar to the robots-target game, but note that neither the cops nor the robber is allowed to hover along edges. It is clear that cop number and robot number are analogous concepts, which allows us to relate them to one another as we will go on to do in Theorem 4.2.3. Before that, we give the main result of Barát's research below [2].

Theorem 4.2.2. *Let G be a graph and let n be the number of cops on G and let $pw(G)$ be the pathwidth of G . Then for $n \geq 1$, we have $pw(G) \leq cn(G) \leq pw(G) + 1$.*

We use Theorem 4.2.2 to prove a relationship between robot number and pathwidth.

Theorem 4.2.3. *Let G be a graph and let $n \geq 1$. Then $pw(G) - 1 \leq r(G) \leq pw(G) + 2$.*

Proof. We claim that $cn(G) - 1 \leq r(G) \leq cn(G) + 1$.

First, let $cn(G) = n$. Then n cops can catch the robber on G . Since the robber is not allowed to hover along edges, it suffices for the cops to land on a vertex that the robber is on in order for the cops to win on G . However, since the target is allowed to hover along edges, the robot strategy for catching the target also involves checking the edges surrounding any vertex that a robot moves to. This is done by having $n + 1$ robots on G , where the $(n + 1)$ th robot acts as a helper robot. Every time a robot moves to some new vertex v in G , the helper robot clears the edges surrounding v . This shows a winning

$r(G) \backslash pw(G)$	1	2	3	4
1	✓	✗	✗	✗
2	✓	✓	?	?
3	✗	✓	✓	?

Figure 4.2.1: Relating robot number and pathwidth.

strategy for $n+1$ robots on G , and we can see that $r(G) \leq cn(G)+1$. From Theorem 4.2.2, we can infer that $r(G) \leq pw(G) + 2$.

Now suppose that $r(G) = n$. Then n robots can win on G . We know that when there is some robot strategy for winning on a graph, this strategy may be implemented by having one robot move at a time. We have $n+1$ cops. Consider n cops to be the main cops and suppose that the other cop is a helper cop. This cop guards any vertex from which another cop is to be lifted, ensuring that the robber is unable to travel through this vertex when the cop is lifted from it. We can deduce that two cops on a single vertex are as effective as one robot on a single vertex. Therefore $n+1$ cops can win on G . We have that $cn(G) \leq r(G) + 1$, which gives us that $cn(G) - 1 \leq r(G)$. From Theorem 4.2.2, we can deduce that $pw(G) - 1 \leq r(G)$.

We conclude that $pw(G) - 1 \leq r(G) \leq pw(G) + 2$. □

We conclude this section with a conjecture that the bounds for the range of pathwidth that robot number may be in is in fact even narrower than what we show in Theorem 4.2.3.

Consider the table in Figure 4.2.1. The figure shows whether for some given robot number, we can find a graph with that robot number which corresponds to either pathwidth 1, pathwidth 2, pathwidth 3 or pathwidth 4. We give a brief explanation of the results in the table below.

We know that the only graph corresponding to $r(G) = 1$ is a path. It is straightforward to see that the pathwidth of a path is at most 1. As a result, we can conclude that a graph with $r(G) = 1$ will not have pathwidth 2 or 3.

Next, consider $r(G) = 2$. Two robots can win on the claw graph, which is a path with a pendant edge and has pathwidth 1. This means that a graph having $r(G) = 2$ may have pathwidth 1.

In Lemma 3.2.1 from Section 3.2, we had described how a topological graph on which two robots can win is obtained. The simple graph version of such a graph is a path to which we add some pendant edges and triangles. Any graph that has a triangle as a minor must have pathwidth at least 2, which means that a graph with $r(G) = 2$ may also have pathwidth 2.

Though we do not have a proof for this, it seems likely that it is not possible for a graph with $r(G) = 2$ to have pathwidth 3 or 4. We give the following conjecture.

Conjecture 4.2.4. *Let G be a graph and let $r(G) = 2$. Then $pw(G) \leq 2$.*

Now consider $r(G) = 3$. By Theorem 4.2.3, we have that a graph with $r(G) = 3$ must not have pathwidth 1.

We know that the $(3, 2)$ -tree is a graph having $r(G) = 3$. In Example 4.1.9 of the previous section, we had found a path decomposition of the $(3, 2)$ -tree which had pathwidth 2, which means that a graph having $r(G) = 3$ may have pathwidth 2.

In Figure 4.2.2, we show a different path decomposition of the $(3, 2)$ -tree than the one shown in Example 4.1.9. It is not hard to see that the width and pathwidth of this decomposition is 3, which gives us that a graph with $r(G) = 3$ may have pathwidth 3.

Though we do not have a proof for this, we believe that a graph with $r(G) = 3$ cannot have pathwidth 4. We give the following conjecture.

Conjecture 4.2.5. *Let G be a graph and let $r(G) = 3$. Then $pw(G) \leq 3$.*

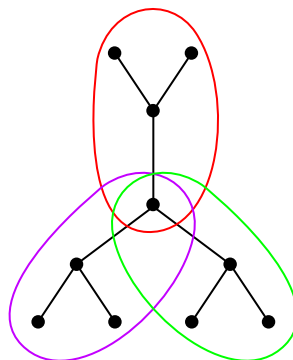


Figure 4.2.2: This is a path decomposition of the $(3, 2)$ -tree which has pathwidth 3.

In general, it seems possible that robot number is only off from pathwidth by 1. We do not have a rigorous argument to prove this result, but give the following conjecture.

Conjecture 4.2.6. *Let G be a graph. Then $pw(G) - 1 \leq r(G) \leq pw(G)$.*

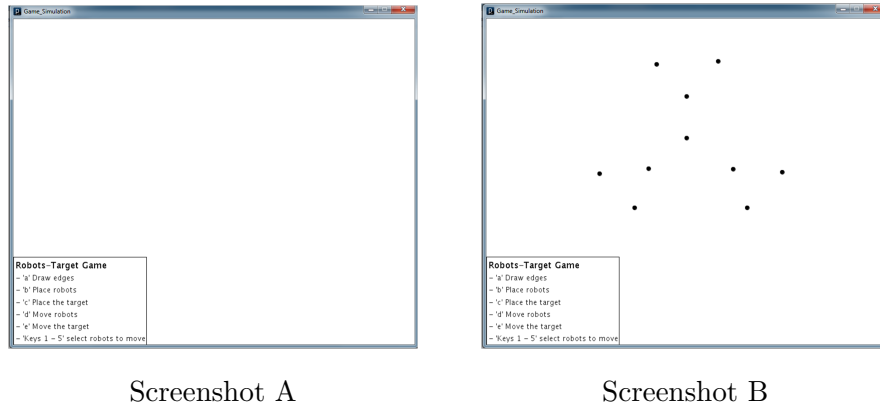
5

Simulating the Robots–Target Game on a Computer

As part of the research for this project, we wrote a program on *Processing*, an open source programming language, which allows us to play the robots-target game interactively on a computer. The code for this program was written using objects and classes and applying a variety of programming fundamentals such as variables, states, arrays, for loops, conditional statements and functions. This program considers the game to be a turn-taking game, even though the game is not defined to be one as such. Note that this aspect of the program does not affect the accuracy or efficiency of the program in any way.

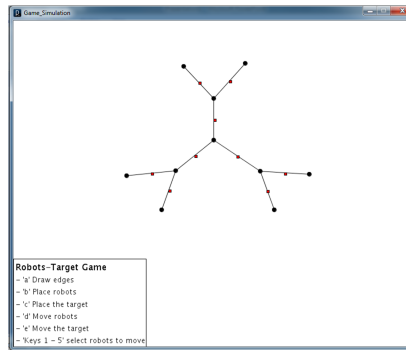
Broadly, there are four stages in the use of this program. In the first, the user draws a graph on the screen. Second, the user places the robots and the target on the screen. Then, the user moves the robots on the graph. In the final stage, the user moves the target. After the first two stages are completed, the user will go back and forth between Stage 3 and Stage 4.

We give some details on the function and use of each stage below. This discussion is accompanied by screenshots of the sketch.



Screenshot A

Screenshot B

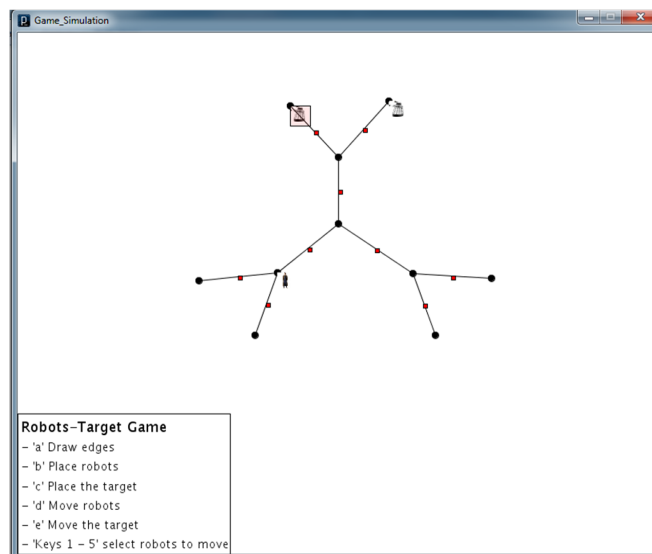


Screenshot C

Figure 5.0.1: Stage 1: drawing a graph in the sketch.

In **Stage 1**, shown in Figure 5.0.1, the user first runs the program and draws a graph to play the game on. As shown in Screenshot A, a blank sketch with a set of instructions on how to use the program appears on the screen. Next, in Screenshot B, the user places the vertices of the graph by clicking on the screen wherever a vertex is required. Finally, the user draws edges between the vertices in Screenshot C by first clicking on the origin vertex and then dragging the mouse to the destination vertex. This is shown in Screenshot C. Observe the small red square at the midpoint of each edge. This was implemented into the program as a way to let the user have the robots and target hover along edges.

Next, in **Stage 2**, the user places the robots and the target on the graph as shown in Screenshot D of Figure 5.0.2. The user places robots and the target on the graph simply

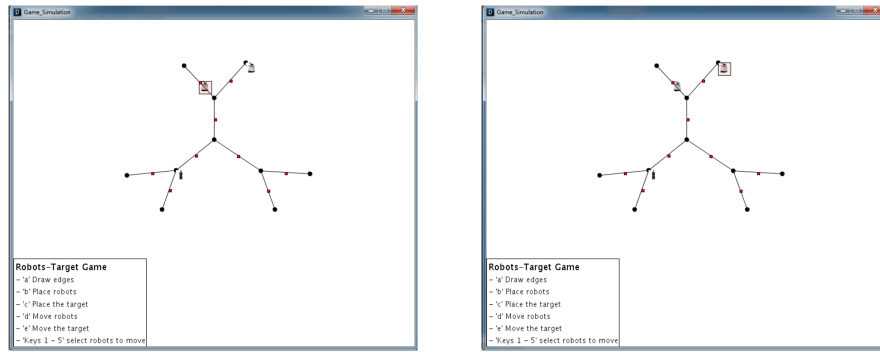


Screenshot D

Figure 5.0.2: Stage 2: placing the robots and the target on the graph.

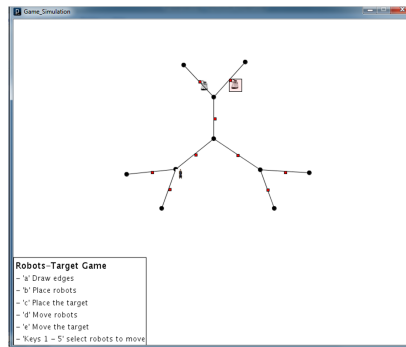
by clicking on any of either the vertices or the midpoints between edges. Observe that one of the robots in Screenshot D has a translucent red square around it. This red square indicates which robot is selected at a given time. The purpose of this square will become clearer in the following paragraph.

In **Stage 3**, shown in Figure 5.0.3, the user moves the robots on the graph. A robot is moved on this graph simply by clicking on a desired destination midpoint or vertex. In Screenshot E, the user moves the robot with the red square around it to the midpoint of the top left edge simply by clicking on it. Recall from Stage 2 that the red square indicates the robot that is selected at a given time. In the code, each robot that is placed on the graph is assigned a number from 1 to 5, and when the robots are placed on the graph, the first robot placed is assigned the number 1 and automatically becomes the selected robot. After that, each robot that is placed is consecutively assigned numbers 2 to 5. Note that we can have up to five robots in a sketch. In order to change the selected robot to move, the user will press any of keys 1 to 5. In this case, the user will press “2” to select



Screenshot E

Screenshot F



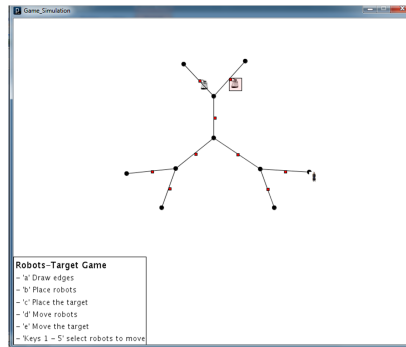
Screenshot G

Figure 5.0.3: Stage 3: selecting and moving the robots.

the other robot on the graph, and we can see in Screenshot F that this selects that robot. Now that once this robot is selected, it can be moved to a vertex or midpoint in the graph just as the other robot was moved, by clicking on the required vertex or midpoint. This is shown in Screenshot G.

Finally, in **Stage 4** shown in Screenshot H of Figure 5.0.4, the user moves the target. The target is moved in the same way as the robots are moved, by clicking on the required destination vertex or midpoint.

As stated in the instructions on the screen, the user can go from one stage to another by pressing keys from “a” to “e”. The robots-target game is a pursuit game requiring both players to move continuously on a graph, and this program was an efficient alternative to



Screenshot H

Figure 5.0.4: Stage 4: moving the target.

representing this motion using a paper and pencil. We used this program extensively to determine forbidden minors and forbidden subquotients for different robot numbers.

We give the code for the program below.

```
Vert [] vert = new Vert[100];
Edge [] edge = new Edge[10000];
Dalek [] dalek = new Dalek[5];
Dalek doctor;

void setup() {
  size(800, 650);
  background(255);
  smooth();
}

int state = 0;
int numvertices = 0;
int numdaleks = 0;
int numedges = 0;
int i = 0;
int currentDalek = 0;
int n = 0;

void mousePressed() {

  if (state==0) {
    if (numvertices < vert.length) {
      vert[numvertices] = new Vert();
      vert[numvertices].place();
      numvertices = numvertices + 1;
    }
  }
}
```

```

else if (state==1) {
  for (i = 0; i < numvertices; i = i + 1) {
    if (dist(mouseX, mouseY, vert[i].x, vert[i].y) < 20) {
      edge[numedges] = new Edge();
      edge[numedges].startpoint(i);
    }
  }
}

else if (state ==2) {
  if (numdaleks < dalek.length) {
    for (n = 0; n < numvertices; n = n + 1) {
      if (dist(mouseX, mouseY, vert[n].x, vert[n].y) < 20) {
        dalek[numdaleks] = new Dalek();
        dalek[numdaleks].placevertex(n);
        numdaleks = numdaleks + 1;
      }
    }

    for ( n = 0; n < numedges; n = n + 1) {
      if (dist(mouseX, mouseY, edge[n].midx, edge[n].midy) <20) {
        dalek[numdaleks] = new Dalek();
        dalek[numdaleks].placeedge(n);
        numdaleks = numdaleks + 1;
      }
    }
  }
}

else if (state == 3) {
  for (n = 0; n < numvertices; n = n + 1) {
    if (dist(mouseX, mouseY, vert[n].x, vert[n].y) <20) {
      doctor = new Dalek();
      doctor.changetoDoctor();
      doctor.placevertex(n);
    }
  }

  for (n = 0; n < numedges; n = n + 1) {
    if (dist(mouseX, mouseY, edge[n].midx, edge[n].midy) < 20) {
      doctor = new Dalek();
      doctor.changetoDoctor();
      doctor.placeedge(n);
    }
  }
}

else if (state == 4) {
  for (n = 0; n < numvertices; n = n + 1) {
    if (dist(mouseX, mouseY, vert[n].x, vert[n].y) < 20) {

```

```

        dalek[currentDalek].placevertex(n);
    }
}

for ( n = 0; n < numedges; n = n + 1) {
    if (dist(mouseX, mouseY, edge[n].midx, edge[n].midy) <20) {
        dalek[currentDalek].placeedge(n);
    }
}
}

else if (state == 5) {
    for (n = 0; n < numvertices; n = n + 1) {
        if (dist(mouseX, mouseY, vert[n].x, vert[n].y) < 20) {
            doctor.placevertex(n);
        }
    }

    for ( n = 0; n < numedges; n = n + 1) {
        if (dist(mouseX, mouseY, edge[n].midx, edge[n].midy) <20) {
            doctor.placeedge(n);
        }
    }
}
}

void mouseReleased() {

    if (state==1) {
        boolean found = false;
        for (i = 0; i < numvertices; i = i + 1) {
            if (dist(mouseX, mouseY, vert[i].x, vert[i].y) < 20) {
                edge[numedges].endpoint(i);
                found = true;
            }
        }
        if (found == true) {
            edge[numedges].drawedge();
            edge[numedges].midpoint();
            numedges= numedges+1;
        }
    }
}

void keyPressed() {

    if (key == '1') {
        currentDalek = 0;
    }

    else if (key == '2') {

```

```

    currentDalek = 1;
}

else if (key == '3') {
    currentDalek = 2;
}

else if (key == '4') {
    currentDalek = 3;
}

else if (key == '5') {
    currentDalek = 4;
}
}

void draw() {
    background(255);
    fill(255);
    rect(0, 475, 265, 400);
    fill(0);
    textSize(18);
    text("Robots-Target Game", 5, 498);
    textSize(15);
    text("\n- 'a' Draw edges\n- 'b' Place robots\n- 'c' Place the target\n- 'd' Move robot
for (n = 0; n < numvertices; n = n + 1) {
    vert[n].place();
}
for (n = 0; n < numdaleks; n = n + 1) {
    dalek[n].show();
}
for ( n = 0; n < numedges; n = n + 1) {
    edge[n].drawedge();
    edge[n].midpoint();
}

if (numdaleks > 0) {
    fill(255, 0, 0, 25);
    rect(dalek[currentDalek].x, dalek[currentDalek].y, 25, 25);
}
if (doctor != null) {
    doctor.show();
}

if (keyPressed && key=='a') {
    state = 1;
}

else if (keyPressed && key== 'b') {
    state = 2;
}
}

```

```
    else if (keyPressed && key == 'c') {
        state = 3;
    }

    else if (keyPressed && key == 'd') {
        state = 4;
    }

    else if (keyPressed && key == 'e') {
        state = 5;
    }
}

class Dalek {

    int n;
    int x, y;
    boolean onavertex = true;
    PImage d1;

    Dalek() {
        x = mouseX;
        y = mouseY;
        d1= loadImage("Dalek Icon.jpg");
    }

    void changetoDoctor() {
        d1 = loadImage("DoctorWho.jpg");
    }

    void show() {

        if (onavertex == true) {
            x = vert[n].x;
            y = vert[n].y;
        }

        else {
            x = edge[n].midx;
            y = edge[n].midy;
        }
        image(d1, x, y, 20, 20);
    }

    void placevertex(int q) {
        onavertex = true;
        n = q;
    }

    void placeedge(int m) {
```

```
    onavertex = false;
    n = m;
}
}

class Vert {
    int x, y;

    Vert() {
        x = mouseX;
        y = mouseY;
    }

    void place() {
        stroke(0);
        fill(0);
        ellipse(x, y, 8, 8);
    }
}

class Edge {
    int midx, midy;
    int x, y;
    int start, end;

    Edge() {
    }

    void startpoint(int j) {
        start = j;
    }

    void endpoint(int k) {
        end = k;
    }

    void midpoint() {
        midx = vert[start].x/2 +vert[end].x/2;
        midy = vert[start].y/2 + vert[end].y/2;
        fill(255, 0, 0);
        rect(midx, midy, 5, 5);
    }

    void drawedge() {
        stroke(0);
        strokeWeight(1);
        line(vert[start].x, vert[start].y, vert[end].x, vert[end].y);
    }
}
```

Bibliography

- [1] Stefan Arnborg, Andrzej Proskurowski, and Derek Corneil, *Forbidden minors characterization of partial 3-trees*, Discrete Math **80** (1990), 1–19.
- [2] János Barát, *Directed path-width and monotonicity in digraph searching*, Graphs and Combinatorics **22** (2006), 161–172.
- [3] Jim Belk, Personal communication (2013).
- [4] Dan Bienstock, Neil Robertson, Paul Seymour, and Robin Thomas, *Quickly Excluding a Forest*, Journal of Combinatorial Theory **52** (1991), 274–283.
- [5] Reinhard Diestel, *Graph Theory*, Springer-Verlag, New York, 2000.
- [6] F.V. Fomin, *Helicopter search problems, bandwidth and pathwidth*, Discrete Applied Mathematics **85** (1998), 59–70.
- [7] Neil Robertson and P.D. Seymour, *Graph Minors. XX. Wagners conjecture*, Journal of Combinatorial Theory **92** (2004), 325–357.
- [8] Robin Wilson, *Introduction to Graph Theory*, Prentice Hall, New York, 1996.
- [9] Boting Yang, *Strong-mixed Searching and Pathwidth*, Journal of Combinatorial Optimization **13** (2007), 47–59.
- [10] Tina Zhang, *Cops and Robbers*, Senior Projects Spring 2009 (2009).