RAT-CATCHING: A PURSUIT-EVASION GAME ON UNDIRECTED GRAPHS

AIDEN BAILEY

Abstract. This paper introduces a pursuit-evasion game and describes the undirected graphs on which the game is winnable. The game consists of a rat walking in a graph and an exterminator attempting to catch it. A winning strategy for the exterminator is a sequence of vertices which intersects every walk. Such a strategy exists if and only if every component of the graph is a lobster tree. The proof is constructive, and provides a linear-time pursuer trajectory on such a graph.

Contents

1. Introduction

This paper is motivated by the following puzzle.

Imagine you are an exterminator hired to exterminate a rat hiding in a building consisting of three adjacent rooms in a line. The extermination process is a turn-based game. Every turn, you open a door to a room of your choice. If the rat is in this room, you kill it and win. Otherwise, you close the door, and the rat moves to a room adjacent to its actual hiding place. The rat must move, and it may move to a room you have previously searched. The challenge is to find a strategy that is guaranteed to locate the rat in a finite number of searches.

A winning strategy to this game would be to open door 2 twice. If the rat is not caught by turn 1, it must have been hiding in either room 1 or room 3. In either case, it must move to an adjacent room, and both of these rooms have only one neighbor: room 2. Then our second search of room 2 is guaranteed to find the rat.

What if the building consisted of four adjacent rooms? What about five, or larger n? What if the adjacency structure of the building is more complicated than a sequence of rooms in a line?

For instance, Figure 1.1 depicts a possible strategy to exterminate a 5-room building in only 6 searches. Room searches are filled in, and all possible rat paths are snuffed out, one by one.

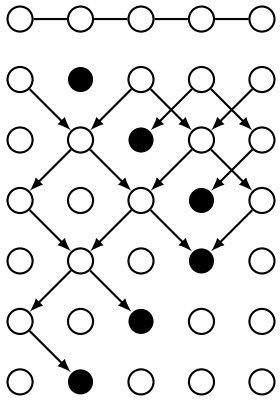

FIGURE 1.1. Solution to the 5-room variation

This generalized problem can be defined in the language of graph theory. It asks:

Problem 1.2. *Given a graph G, is there a k-vertex sequence $(s_n)_{n=1}^k$ in G, for some finite k, which intersects every k-vertex walk $(t_n)_{n=1}^k$ through G ($s_m = t_m$ for some $m \leq k$)?*

The main result of this paper is that for undirected graphs, this property is equivalent to being a "lobster forest", a graph whose components are lobsters.

Definition 1.3. A tree is a **lobster** if there is a path within distance 2 of all vertices. See Figure 1.4 for an example.

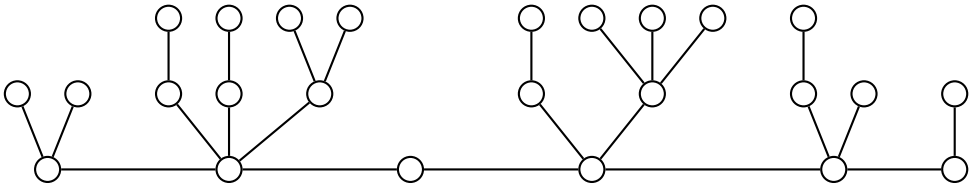

FIGURE 1.4. A lobster tree

Some background in pursuit-evasion games is provided in Section 2. A useful characterization of graphs, called "spinewidth," is defined and investigated in Section 3. The main result is proved in Section 4. Finally, some unanswered questions and possible applications are addressed in Section 5.

2. PURSUIT-EVASION

Problem 1.2 is one of a larger class of problems in game theory, called "pursuit-evasion games." This game consists of one group (the pursuers) attempting to track down members of another group (the evaders) in an environment. This environment can be continuous or discrete.

Pursuit-evasion games have a rich history. As early as the 18th century [9], mathematicians were calculating "pursuit trajectories", the paths traced by pursuers. Not long after, the evader was introduced. Now, optimal strategies were required

for both agents simultaneously. These were analytical problems; they involved continuous space and continuous time. It wasn't until 1976 that a pursuit-evasion problem was posed in a discrete, graph-theoretic environment [12]. However, in this problem, agents occupied points on edges, and still moved continuously. Finally, in 1978, the *Cops and Robbers* game [4, 10], utilized discrete time. Instead of occupying points on the edge of a graph, agents occupied vertices, and moved to neighbors one step at a time.

Pursuit-evasion problems are abundant, and are applicable in a number of fields, including "mobile robotics, search-and-rescue, surveillance, tracking, harvesting and many others" [7]. Much literature is written on this subject, especially in recent years [2, 11], in part due to the vast number of ways these games may vary. A fundamental difference is between continuous and discrete formulations, for both environment and time. There may be one pursuer or several. The evader may be adversarial (optimal), drunk (random), or simple (moving on a predetermined path). The evader may be visible to the pursuer or invisible, and vice versa. Movement may be optional or forced. Velocity may be bounded, unbounded continuous, or discontinuous.

In our case, the exterminator is the sole pursuer, and the rat is the sole evader. However, since the problem is deterministic and not probabilistic (we want a guarantee of success, not an expectation), it is equivalent to A) exterminating many rats, starting everywhere and making every possible decision, and also to B) an intelligent rat, fully aware of our entire plan and able to choose its starting location.

Our query can be described as a pursuit-evasion problem with the following attributes:

- The environment is discrete (a graph).
- At every step, the pursuer and the evader each occupy a vertex of the environment graph. (In some other pursuit-evasion games, the players may occupy a point on an edge [12].)
- The pursuer succeeds when he occupies the same vertex as the evader.
- The pursuer and evader move simultaneously. These time steps, or "turns," are discrete.
- The pursuer has unlimited locomotion. This is often described as the pursuer having a "helicopter."
- The evader must move to a neighbor each turn. Broken down further:
  - The evader may move at most distance 1 each turn. This is often described as the evader having "velocity 1."
  - The evader must move each turn. In other words, we require the evader follow a walk on the graph. (In some other pursuit-evasion games, the evader is allowed to follow a "lazy" walk, where he may either move or remain still.)
- The evader is invisible. The pursuer has no information available to him, aside from his own location and the shape of his environment.
- The evader is adversarial. The evader has perfect information on the pursuer's location and plan, and the evader chooses his own starting vertex.

We ask on what graphs does the exterminator have a winning strategy.

3. Spinewidth

Before answering the main problem, it will be helpful to define a certain value on graphs, and to develop an understanding of this value's relationship to other graph-theoretic structures. The value is a measure of how "fat" the graph is, i.e. how far its vertices are from a central path.

Definition 3.1. The **spinewidth** of a connected graph G, written $\mathrm{sw}(G)$, is the smallest number n such that G contains a path within distance n of all vertices in G. A **spine** is such a path within distance n of every vertex.

To frame this concept in well-established terms, here are some immediate observations:

- A traceable graph is a graph with spinewidth 0.
- A path graph is a tree with spinewidth 0.
- A **caterpillar** is a tree with spinewidth ≤ 1.
- A **lobster** is a tree with spinewidth ≤ 2.

Remark 3.2. Observe that spinewidth is analagous to the notion of radius, with a single point center replaced by a path center. Recall radius measures how far a graph's vertices are from a central point, chosen to minimize this distance. Indeed, for this reason, spinewidth is bounded by radius. To see why, consider the 0-long path given by (o), where o is the central vertex achieving radius $r(G)$.

The majority of our investigation into this property will deal with spines of trees. This will be sufficient because, in general, spinewidth is determined by a graph's spanning trees, in the following sense:

Proposition 3.3. *The spinewidth of a graph G is the minimum spinewidth over all spanning trees of G.*

Proof. Suppose H is a spanning tree of G. Let $(q_i)_{i=0}^{z}$ be a spine for H achieving spinewidth n. Then q is also a path through G, and every vertex in G connects to q by a path of length at most n. In other words, the spinewidth of G is at most n. This proves the spinewidth of G is a lower bound on the spinewidth of all spanning trees of G.

Now suppose the graph G has spinewidth w, and let $(p_i)_{i=0}^{l}$ be a spine of G. We will construct a spanning tree H of G with spinewidth w.

Begin with only the edges of the spine p. For each non-p vertex v, take the first edge of a shortest path from v to p. The path traced by these edges, from any v to a neighbor 1 closer to p, will be a shortest path to p, and hence of length $\leq w$. So, the subgraph H resulting from this process has spinewidth at most w.

Because H includes every vertex, it is spanning. Because it begins with l edges and adds $v-(l+1)$ more (one for each non-p vertex, of which there are $l+1$) and because every vertex is connected to the spine p which is itself connected, H is a connected subgraph with v vertices and $v-1$ edges, hence it is a spanning tree.

We already knew w was a lower bound on spanning tree spinewidth, and now we have a tree whose spinewidth is at most w, so $\mathrm{sw}(H)=w$, proving the claim. □

The remainder of this section deals with three principal areas of interest. First, we analyze the relationship between spines and longest paths. Second, we describe a family of "fundamental trees" for the value of spinewidth. Third, we analyze how

spinewidth interacts with the interior operation on graphs (the removal of all leaf nodes).

But before all that, we begin with a few technical lemmas. Recall that the length of a path is the number of edges. This number is one less than the number of vertices. The distance between two connected vertices is the length of the shortest path connecting them (or equivalently the shortest walk).

Lemma 3.4. *For any connected subgraph H of a tree G, each vertex v in G is closest to a unique vertex x in H, and any walk from v to $y \in H$ runs through x.*

Proof. Recall that on a tree there is a unique simple path between any two vertices.

Let x be one of the vertices on H closest to v. The unique simple path between x and any other y on H is the one which lies entirely on (connected) H.

These two paths $v \to x$ and $x \to y$ must be disjoint, since the latter consists entirely of vertices in H, and if any H vertex is visited before x then x was not closest to v. Thus their concatenation $v \to x \to y$ is simple.

In other words, for all $y \in H$, the unique simple path from v to y begins with the unique simple path from v to x. In general, all walks $v \to y$ (which can be loop-erased to the simple path $v \to y$) must contain x.

For each $y \in H \setminus \{x\}$, since the unique path $v \to y$ is an extension of the path $v \to x$, it is strictly longer. Thus, x is the unique vertex of H closest to v. □

Lemma 3.5. *Suppose H is a connected subgraph of a tree G, and vertex v is distance $n+1$ from H, where $n \geq 0$. Then among v's neighbors:*

- *Exactly one is distance n from H*
- *The rest are distance $n+2$ from H*
- *All share the same nearest H-point with v.*

Proof. Let $(t_i)_{i=0}^{n+1}$ be the unique simple path from v to the nearest vertex h in H, as guaranteed by Lemma 3.4. Observe $t_0 = v$ and $t_{n+1} = h \in H$. Moreover, t_1 exists since $n \geq 0$. Because t is a path, $(t_0, t_1) \in E$ and t_1 is a neighbor of $t_0 = v$.

The truncation $t'_i = t_{i+1}$ is an n-long path from $t'_0 = t_1$ to $t'_n = h$, so t_1 is distance at most n from H. Any shorter path $t_1 \to H$ would concatenate with (t_0, t_1) to produce a path shorter than $n+1$ from v to H, a contradiction. Hence t_1 is distance n from H.

Suppose y is a neighbor of v other than t_1. Observe $y \neq h$, or else $(v, y) = (v, h)$ is the unique simple path $v \to h$, whose first edge is (v, t_1), so $y = t_1$. Moreover, $y \notin H$, or else h was not the unique vertex of H nearest to v.

If we have $t_j = y$ for some $j = 2, ..., n$, then the truncation $t_0 \to t_j$ of simple path t would be a different simple path $v \to y$ than the 1-long simple path (v, y) (impossible). So, y does not occur on t and it may be adjoined to the head while remaining a simple path:

$$s_i = \begin{cases} y & i = 0 \\ t_{i-1} & 1 \leq i \leq n+2 \end{cases}$$

Thus s is the unique simple path $y \to h$. Observe it has length $n+2$.

By Lemma 3.4, any walk from y to H must run through the nearest vertex of H. By the construction of t, only $t_{n+1} = h$ lies in H, so $s_{n+2} = h$ must be this nearest vertex. Therefore, s is the shortest path from y to the nearest vertex of H, hence y is distance $n+2$ from H.

In both cases, the constructed shortest path to H ended at h, the nearest H-point to v, so all neighbors of v share the same nearest H-point with v. □

Lemma 3.6. *On a connected graph G with spinewidth n, for each endpoint y of a spine of minimum length, there is a vertex x closer to y than to any other vertex on the spine, and x is distance n away from y.*

To see an instance of the behavior this lemma guarantees, refer to Figure 3.7. Throughout this paper, many figures with relevant spines will be drawn so the spine lies at the bottom of the graph.

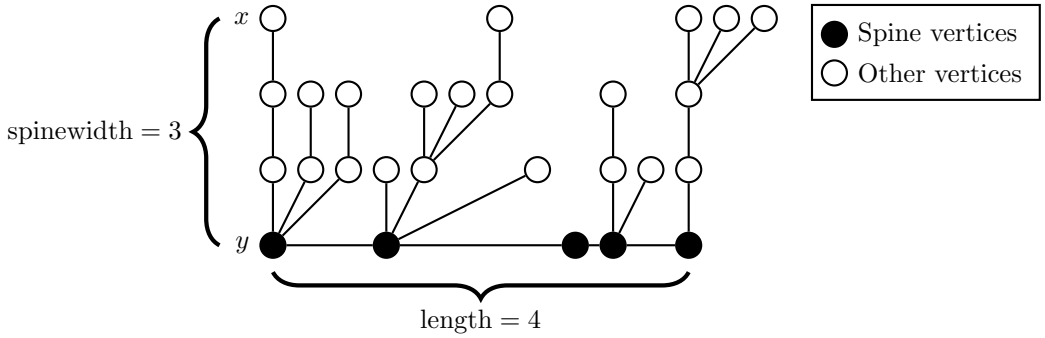

FIGURE 3.7. $d(x, y) = \mathrm{sw}(G) = 3$

Proof. Let $(p_i)_{i=0}^{l}$ be a spine of minimum length. It suffices to prove the claim for $y = p_0$. The same logic applies to both endpoints.

Assume all vertices are either A) just as close or closer to p_i than to p_0, for some $i \neq 0$, or B) within $n-1$ distance of p_1.

Consider the shorter path $p'_i = p_{i-1}$ which skips p_0. For any type A vertex, a shortest path from v to spine vertex p_i is still a shortest path to a vertex of p'_{i-1}, since by assumption $p_i \neq p_0$. For any type B vertex, the path to p_0 of length at most $n-1$, followed by the edge (p_0, p_1) is at most an n-long path to $p_1 = p'_0$. Therefore, every vertex is at most n away from p', so p' is a shorter spine than p, a contradiction. Hence the claim is shown. □

Proposition 3.8. *For a tree,*

$$\textit{minimum spine length} + 2 \times \textit{spinewidth} = \textit{maximum path length}$$

This result should not be surprising. It is intuitive that the two simple paths guaranteed by Lemma 3.6, each of length $\mathrm{sw}(G)$, could concatenate with the spine of minimum length to form a path, and that this path is of maximal length. In Figure 3.7, G has spinewidth 3 and minimum spine length 4, and indeed the longest paths have length 10.

Proof. Let $(p_i)_{i=0}^{l}$ be a spine of minimum length for tree G, where l is the minimum spine length, and w the spinewidth.

First, suppose $l = 0$. Consider all the furthest vertices from p_0, all having distance w from p_0. Assume the simple paths to these far vertices all begin with the same edge (p_0, c). Then every vertex not within $w-1$ of p_0 is within $w-1$ of c,

so (p_0, c) is a path achieving distance $w-1$ from all vertices. But this is impossible, since w was the spinewidth. So, there are at least two edges which begin simple paths to far vertices. Then there are at least two disjoint simple paths of length w beginning at p_0, say to far vertices x_0 and x_l. Their concatenation $x_0 \to x_l$ is $2w$-long.

Now, suppose $l > 0$. From Lemma 3.6 each endpoint p_0, p_l is the closest spine vertex to some vertex x_0, x_l which is w far from it.

The concatenation of simple paths $x_0 \to p_0, p_0 \to p_l, p_l \to x_l$ is itself a simple path, since no p_i precedes p_0 nor succeeds p_l (by definition of x_i), and since any intersection between the first and last paths among the non-p_i vertices would constitute a path from p_0 to the intersection to p_l. This would be a simple path $p_0 \to p_l$ other than the one lying solely on spine p (impossible).

The path $x_0 \to x_l$ then has length $2w + l$.

$$d(x_0, x_l) = d(x_0, p_0) + d(p_0, p_l) + d(p_l, x_l) = 2w + l$$

We have shown that whether $l = 0$ or $l > 0$, there is a path of length $2w + l$. We want to determine which paths in G have length equal to or greater than this one. Let $(t_i)_{i=0}^{k}$ be a k-long simple path through G for some $k \geq 0$. Let p_a and p_b be the unique spine vertices closest to t_0 and t_k respectively, invoking Lemma 3.4. By definition of spinewidth, the paths $p_a \to t_0$ and $p_b \to t_k$ are both bounded by length w. Also the simple path $p_a \to p_b$ along p is bounded by l (the length of p). The simple path t between t_0 and t_k is the shortest walk between t_0 and t_k (since the shortest walk is a simple path, and trees have unique simple paths). Therefore, the length k of the arbitrary simple path t is bounded by the length of the walk $t_0 \to p_a \to p_b \to t_k$, which is at most $2w + l$. In other words, the path we already found is of maximum length $2w + l$. □

Moreover, we can observe that the only time this bound is an equality and not a strict $<$ is when p_a and p_b are exactly l apart, which is to say they are opposite endpoints of the spine p.

By the same logic as above, the path $t_0 \to p_a \to p_b \to t_k$ is simple. By simple path uniqueness, this must be t itself, so t must visit each vertex in p.

In other words, every vertex in G is within w of some vertex in t, so t is itself a spine, whenever t is of maximum length. We state this final result as its own corollary:

Corollary 3.9. *Any longest path in a tree contains a minimum length spine. So, a longest path in a tree is a spine.*

This result is not true of graphs in general. For instance, Figure 3.10 depicts a graph without a spine achieving maximum path length.

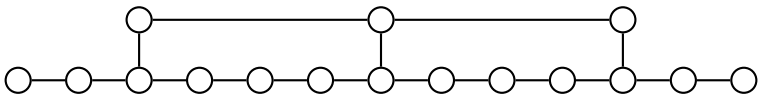

FIGURE 3.10. Maximum path length 13, maximum spine length 12

Definition 3.11. Define T_n to be the tree which consists of 3 copies of the path graph P_n and a node adjacent to the head of each copy.

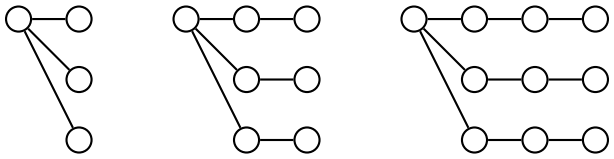

FIGURE 3.12. T_1, T_2, and T_3

Provided in Figure 3.12 are the first three examples of T_n

In a sense, these graphs are "fundamental" trees of spinewidth n. Their presence is the sole determining factor in the value of spinewidth (for trees). Equivalently, T_{n+1} is the only minimal forbidden subgraph for trees of spinewidth $\leq n$.

Proposition 3.13. *A tree G has spinewidth $\geq n$ iff it contains T_n as a subgraph.*

Proof. Suppose T_n is a subgraph of the tree G. The induced degree of every vertex on the spine (p_i) of G is at most 2 (being a path), so at least 1 of the 3 central edges is untouched by p. This exclusion isolates one of the three axes from the spine. The shortest path from the extreme point x of this axis to the spine must then include the unique simple path from x to the center, which is distance n. So, the spinewidth is at least n.

Now suppose tree G has spinewidth $\geq n$, and let $(p_i)_{i=0}^m$ be a path of G achieving maximum length m (and therefore a spine of G by Corollary 3.9).

There is at least one vertex x which is distance n away from p, or else p would be a spine achieving spinewidth $\leq n-1$. From Lemma 3.4, there is a unique nearest p_j to x, for some j. Assume p_j is less than n away from an endpoint e_1. Then the simple path from the other endpoint e_2 to x (running through p_j) has length

$$d(e_2, x) = d(e_2, p_j) + d(p_j, x) = m - d(e_1, p_j) + n \geq m - (n-1) + n = m + 1$$

But then p wasn't a longest path of G, a contradiction. Therefore, p_j is at least distance n away from both endpoints.

Thus three disjoint n-long simple paths $(p_j \to x, p_j \to e_1, p_j \to e_2)$ begin at p_j, exactly the structure which defines T_n. □

The same does not hold for arbitrary graph G. After all, every T_n is a subgraph of the complete graph K_{3n+1}, which are all traceable and have spinewidth 0. In general, trees are the connected graphs which attain high spinewidth. Adding an edge never increases spinewidth, it only possibly decreases it.

Definition 3.14. A **leaf** in a graph, or an **external node**, is a vertex with degree ≤ 1. The **interior** $I(G)$ of a graph G is the subgraph induced by the set of non-leaf nodes, or **internal nodes**.

This concept is visually represented in Figure 3.15. The white nodes are leaves and the black nodes are internal nodes. The subgraph of black nodes is the interior of the total graph.

The graph this figure provides can be described as a tree with spinewidth 3. Inspecting the black subgraph, we find another tree, this one with spinewidth 2. One might suspect that taking the interior of a tree always reduces the spinewidth by 1, and indeed this is true.

Proposition 3.16. *The interior of a connected graph is connected (or $\emptyset$).*

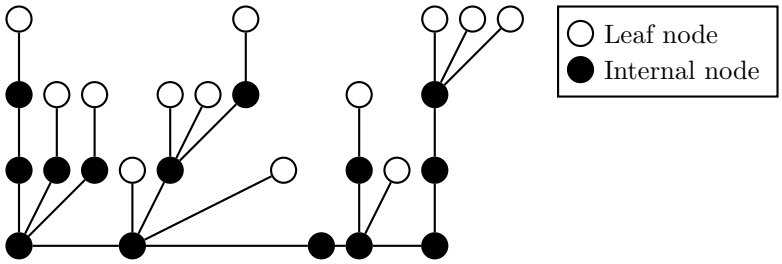

FIGURE 3.15. Graph interior

Proof. Consider the size of $I(G)$

Case I) $I(G) = \emptyset$.

Case II) $I(G)$ has 1 vertex, and thus is trivially connected.

Case III) $I(G)$ has at least 2 vertices. Let x and y be vertices in $I(G)$. They are inherited from G, which is connected, so let $(s_n)_{n=1}^{l}$ be a path in G with $s_1 = x, s_l = y$. All vertices s_i for $1 < i < l$ are of degree at least 2, since $s_i \sim s_{i-1}, s_i \sim s_{i+1}$. Hence these are all kept in $I(G)$. Then the path s lies entirely in $I(G)$, so x and y are connected in $I(G)$, and $I(G)$ is connected.

□

Corollary 3.17. *The interior of a tree is a tree (or $\emptyset$).*

Proof. $I(G)$ is a subgraph, so any cycle in $I(G)$ must also occur in G. □

Lemma 3.18. *In a connected graph G other than the isolated point K_1 or isolated edge K_2, every leaf has a non-leaf neighbor.*

Proof. Suppose v is a leaf in an arbitrary connected graph G without a non-leaf neighbor.

Case I) $d_v = 0$
Then v is connected to no other vertices, so it is all of G, and $G = K_1$.

Case II) $d_v = 1$
Then v has one neighbor u, which is also a leaf, and hence connects to no other vertices. Then $G = K_2$.

□

Proposition 3.19. *If the spinewidth of a tree G is $n+1$ then the spinewidth of $I(G)$ is n, for any $n \geq 0$. That is,*

$$\mathrm{sw}(G) = n + 1 \implies \mathrm{sw}(I(G)) = n$$

Furthermore, G and $I(G)$ share the same minimum length spines.

Proof. Let $(p_i)_{i=0}^{l}$ be a spine of minimum length for tree G with spinewidth $n+1$.

By Lemma 3.6, both endpoints of p are the heads of simple paths consisting of no other p vertices. Then the endpoints both border at least one p vertex (p_1 and p_{l-1} respectively) as well as one non-p vertex. Each has degree $d \geq 2$.

Non-endpoints of p also have degree $d \geq 2$, since $p_i \sim p_{i-1}$ and $p_i \sim p_{i+1}$. Then p is entirely non-leaf, and is retained in $I(G)$.

There are no vertices distance $n+2$ from p, so from Lemma 3.5, all vertices distance $n+1$ from p have exactly one neighbor, and are therefore leaves not retained in $I(G)$. In other words, there are no vertices distance $n+1$ from p in $I(G)$, hence

$$\mathrm{sw}(I(G)) \leq n$$

Let $(q_i)_{i=0}^{l'}$ be a spine for $I(G)$.

Graph G is neither K_1 nor K_2, since $\mathrm{sw}(G) \geq 1$, so we may apply Lemma 3.18 to it. Any vertex v in G is either a non-leaf or a leaf with a non-leaf neighbor. Then $d(v,q)$ is bounded by either $\mathrm{sw}(I(G))$ or $\mathrm{sw}(I(G))+1$ respectively.

q is a simple path in G within distance $(\mathrm{sw}(I(G))+1)$ of every vertex, so

$$n+1 \leq \mathrm{sw}(I(G))+1 \implies \mathrm{sw}(I(G)) \geq n$$

$$\mathrm{sw}(I(G)) = n$$

Furthermore, a minimum length spine of G is a path within distance n of all of $I(G)$, so it is a spine of $I(G)$. Likewise, a spine of $I(G)$ is a path within distance $n+1$ of all of G, so it is a spine of G. Those spines achieving minimum length in either graph achieve the same length in the other, so G and $I(G)$ share minimum length spines. □

Corollary 3.20. *The spinewidth of a tree G is the minimum n such that $I^n(G)$ is a path graph.*

Proof. The above result gives that each application of I lowers spinewidth until it hits zero, at which point $I^n(G)$ remains a tree, so $I^n(G)$ is a path graph.

Because spinewidth only lowers by one per each application, n is also the first time $I^n(G)$ can be a path graph. □

Incidentally, this gives a linear-time algorithm to find a longest path in a tree. The tree can be successively reduced to its interior until $I^n(G)$ has at most 2 leaves. At this point it is a path graph, and backtracking to find a path of n vertices from each endpoint gives a path of maximum length, by Proposition 3.8.

Proposition 3.21. *The minimum spine $(p_i)_{i=0}^{l}$ of a tree G is unique up to reversal, and any path in G is a spine iff it contains p as a subpath.*

Proof. By Proposition 3.19, every application of I reduces the spinewidth by 1 but retains the same minimum length spines. If spinewidth began at n, then after n applications, the only spines of $I^n(G)$ are the traces of the path graph, of which there are at most two (forwards and backwards). Then G also had only one minimum length spine $(p_i)_{i=0}^{l}$, say of length l, up to reversal.

The backward direction of the claim is trivial, since any path containing p as a subpath is trivially a spine itself.

Now, suppose $(q_i)_{i=0}^{k}$ is another spine of G with non-minimum length $k > l$. Truncate q as much on each side as possible while the result remains a spine. Call this truncation $(q'_i)_{i=0}^{z}$. To prove the claim, it suffices to show this spine is of minimum length. Then it would be the unique spine of minimum length p.

If both endpoints q'_0, q'_z have vertices x_0, x_z closest to them and distance n away, then the unique simple path $x_0 \to q'_0 \to q'_z \to x_z$ is of length $2n+z$. From Proposition 3.8, all simple path lengths are bounded by $2n+l$, so $z \leq l$ and q' is of minimum length.

Now suppose there are no vertices whose nearest q-point is an endpoint at distance n. Without loss of generality, call this endpoint q'_0. Then the truncation $(q''_i)_{i=0}^{z-1}$ given by

$$q''_i = q'_{i+1}$$

maintains the same distance to any vertex whose closest q' point was not q'_0. For all other vertices, the distance grows by only 1, since q'_0 and q''_0 are neighbors. These distances were at most $n-1$, so this new truncation q'' is also a spine. But this violates our assumption that q' could not be truncated further while remaining a spine. A contradiction.

Thus, the claim is shown. □

4. Exterminable Graphs

We now return to our original query Problem 1.2. Let us begin by giving a name to the property we are searching for.

Definition 4.1. An **extermination sequence** for graph G is a k-vertex sequence of vertices in G which intersects every k-vertex walk through G. An **exterminable** graph is one which admits an extermination sequence.

Although the final result of this paper only applies to undirected graphs, the problem itself is well-posed for directed graphs as well. For this reason, and since many of the intermediate steps apply to both types, we will consider directed graphs in addition to undirected graphs.

Listed below are some principles of intuition endowed by the puzzle, and also where these principles will appear as formalized proofs throughout this section.

- In general, larger buildings are more difficult to exterminate.
  - Any "sub-building" (subgraph) should be easier than the larger building it appears in. After all, we may always consider a sub-building as belonging to a larger complex, and perform the same room checks. The rat won't be more able to escape our search than if it had more options. This is proved for subgraphs in Proposition 4.3
- Multiple disconnected buildings can be checked one by one.
  - If we are hired to exterminate a rat hiding in one of several buildings, and we would be able to kill the rat if it were hiding in any one of the buildings, then we can win. Simply checking each building individually by its particular strategy should suffice. Exactly this strategy is used to prove this result in Corollary 4.5
- Sometimes there are two groups of rats, and exterminating both groups is only as hard as exterminating one.
  - Inspecting Figure 1.1, one might notice that halfway through the strategy, all rats beginning in an even-numbered room have been killed. Furthermore, even if we refrained from searching any more rooms, the dirty region would never grow back to the full building. This is because there are two groups of rats which can never meet, and which alternate habitats every turn. This distinction is illustrated by Figure 4.2, where the extermination strategy has been split into its two halves. One kills all "even" rats, the other all "odd" rats.
    If we can kill all of one group, that same strategy should be able to kill all of the other, by simply waiting until they appear identical to

the ones we already know how to kill. This property, which is only relevant for bipartite graphs, is proven in Proposition 4.8

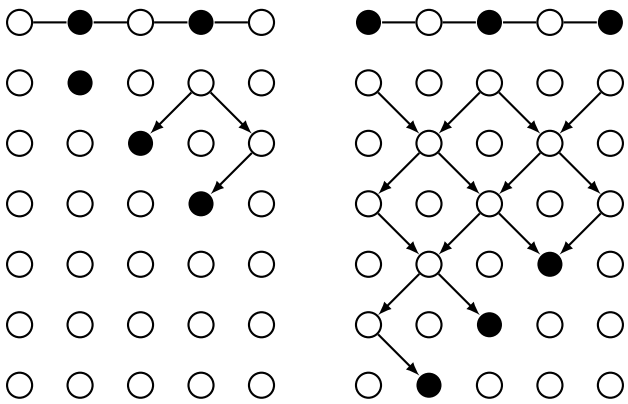

FIGURE 4.2. The two groups of rats in P_5

- The exterminator reduces the "dirty" region of the graph.
  - Another valid mathematical interpretation of the problem is: instead of finding a sequence which catches every possible rat, find a sequence which reduces the set of possibly rat-infested rooms from everywhere to nowhere. These two notions of exterminable, one microscopic, considering every walk individually, and one macroscopic, considering the total region of not-yet-intersected walks, ought to be equivalent. A sequence of rooms which meets every possible rat path should also be a sequence of rooms which reduces the "dirty" region of the graph, the set of rooms where a rat might be hiding, from everywhere to nowhere. This equivalence is formalized and proved in Definition 4.9 and Proposition 4.12.
- "Progress" must always be possible. Stagnation is defeat.
  - A "simple" strategy should never return to a prior state, only ever make progress until there is no progress to be made (no rats survive). For instance, if all 3 rooms in the original puzzle had been connected in a triangle, there could be no successful strategy, because no matter which room was checked, the rat could be anywhere next turn. The first, general statement appears as Corollary 4.14, and the second statement (that a circle of rooms cannot be exterminated) appears as Corollary 4.16.

Proposition 4.3. *If H is a subgraph of G, and G is exterminable, then H is exterminable.*

Proof. Let $(s_n)_{n=1}^k$ be an extermination sequence of G, and let v be a vertex in H. Define sequence $(s'_n)_{n=1}^k$ in H as

$$s'_n = \begin{cases} s_n & s_n \in H \\ v & s_n \notin H \end{cases}$$

Any walk $(t_n)_{n=1}^k$ through H is also a walk through G, so it intersects s at some $m \leq k$. Then s' also intersects t at m:

$$s_m = t_m \in H \Rightarrow s'_m = s_m = t_m$$

□

Proposition 4.4. *A directed graph is exterminable iff every strongly connected component is exterminable.*

Proof. Every strongly connected component of a directed graph G is a subgraph of G, so the forward direction follows from Proposition 4.3.

Let C_i, for $i = 1, ..., l$, be the strongly connected components of G, ordered so that for any $j > i$ there is no path $C_j \to C_i$. This order is possible since the "condensation" of a directed graph (the graph constructed by quotienting out by strong connectivity) is directed acyclic, and hence defines a partial order, which can be extended to a total order.

Suppose these C_i are exterminable, so each C_i has a k_i-vertex extermination sequence $(s_{i,n})_{n=1}^{k_i}$. Define d_i to be the partial sums of k_i:

$$d_j = \sum_{i=1}^{j} k_i$$

Define sequence $(s_n)_{n=1}^{d_l}$ through G as the concatenation of these sequences in order:

$$1 \leq n - d_{j-1} \leq k_j \implies s_n = s_{j,n-d_{j-1}}$$

Strong induction on i will prove the following claim: "For any d_l-vertex walk $(t_n)_{n=1}^{d_l}$ and any $m \geq d_i$, if $t_m \in C_i$ then t intersects s."

Suppose the claim holds for all $j < i$. Let $m \geq d_i$ and t be a walk with $t_m \in C_i$. There is a walk $t_{d_{i-1}+1} \to t_m$ given by truncating t. From the ordering of C_i, we must have that $t_{d_{i-1}+1} \in C_j$ for some $j \leq i$.

If $j < i$, then the desired result follows from the inductive hypothesis ($d_i \geq d_j, t_{d_i} \in C_j$). Else, $j = i$.

By the component ordering, t can never leave C_i after vertex $d_{i-1} + 1$. If it moved to an earlier C_j, it would be a direct violation of the ordering. If it moved to a later C_j, its eventual return by vertex m would be a violation. Then we may truncate t to a walk t' lying entirely in C_i.

$$t'_n = t_{n+d_{i-1}}, \text{ for } n = 1, ..., k_i$$

But $s_{i,n}$ is an extermination sequence for C_i, so there is some $x \in 1, ..., k_i$ at which $s_{i,x} = t'_x$. Then s intersects t at $d_{i-1} + x$:

$$s_{d_{i-1}+x} = s_{i,x} = t'_x = t_{d_{i-1}+x}$$

Thus the induction is shown. Therefore, no matter which strongly connected component C_i contains t_{d_l}, we have $d_l \geq d_i$, so t must intersect s. In other words, s is an extermination sequence. □

In an undirected graph, this reduces to the following statement:

Corollary 4.5. *An undirected graph is exterminable iff every component is exterminable.*

We are building towards proving the desired microscopic-macroscopic equivalence. An intermediate step is to describe "partial cleanings" of a graph G, i.e. reductions of partially dirty states, via a sequence which intersect all walks beginning in some specified subset of G.

Definition 4.6. A subset $S \subset V$ is **sub-exterminable** if some k-vertex sequence intersects every k-vertex walk beginning in S.

Before we link this new concept to the upcoming reformulation of exterminability, let us clarify how it coincides with our prior construction:

Proposition 4.7. *$G = (V, E)$ is exterminable iff every subset $S \subset V$ is sub-exterminable.*

Proof. Both directions are trivially true. Any walk in G beginning in S is a walk in G, so it intersects any extermination sequence of G. If every subset is sub-exterminable, then V is sub-exterminable, so some sequence intersects every walk beginning in V (all possible walks). □

This concept allows us to describe the exterminability of smaller portions of a graph. It is a necessary step to begin to describe the gradual descent of the dirty region. In the meantime, it gives a language to formalize our intuition about the simplification of bipartite graph strategies.

Proposition 4.8. *If undirected bipartite G has parts A and B, and A is sub-exterminable, then G is exterminable.*

Proof. Let $(s'_n)_{n=1}^k$ be the k-vertex sub-extermination sequence for A, for some k.

Define $2k$-vertex sequence $(s_n)_{n=1}^{2k}$ in G as

$$s_n = \begin{cases} s'_n & n \leq k \\ s'_{2k+1-n} & n > k \end{cases}$$

Let $(t_n)_{n=1}^{2k}$ be a $2k$-vertex walk in G.

First suppose $t_1 \in A$. Then the first half of t is a k-vertex walk beginning in A, and at some $m \leq k$, it intersect s'. At this m, it also intersects s:

$$s_m = s'_m = t_m$$

Now suppose $t_1 \in B$. The walk t must alternate between parts of G (consecutive t-points are adjacent). Then t_{2k}, occurring odd many $(2k-1)$ steps after t_1, must lie in A. Define the reversed walk $t'_n = t_{2k+1-n}$. Now we have $t'_1 \in A$, so t' intersects s' at some $m \leq k$ by the same logic at above. Then s_n intersects t_n at $2k + 1 - m$:

$$s_{2k+1-m} = s'_m = t'_m = t_{2k+1-m}$$

t was an arbitrary $2k$-vertex walk, so s is an extermination sequence. □

In fact, "undirected" is not a property required in the above proof. Instead of concatenating a reversal of the original sequence, we could have concatenated the original sequence in the same order, shifted to the right by either 0 or 1 so that this second run-through begins on a turn with opposite parity as the first. However, we only need the statement on undirected graphs, and the reversal method guarantees a faster extermination sequence whenever both apply.

Finally we may define our two notions of exterminability and show their equivalence. The first notion involves a sequence of vertices which intersects every walk. The second involves a sequence of searches which reduces the set of possible rat locations from everywhere to nowhere. As the exterminator progressively cleans the building, there will be some shrinking region in which a rat might still be hiding. We wish to formalize the step-by-step shrinking.

For a set of possible rat locations S, and for room choice u, we will define $\rho(S,u)$ to be the set of possible rat locations next turn, given that the rat was not in room u. In other words, $\rho(S,u)$ is a reduction of the dirty region S, generated by searching room u. Then, we can consider the space of all possible dirty regions, i.e. vertex sets, and ask whether we can reduce each S_1 to each S_2, step by step. This structure is a directed graph Ω_G, whose vertices are dirty regions and whose edges are one-step reductions.

Definition 4.9. For any directed graph $G=(V,E)$, consider the following function ρ defined on vertex sets in G. $\mathcal{P}(V)$ is the set of vertex sets.

$$\rho : \mathcal{P}(V) \times V \to \mathcal{P}(V)$$

$$\rho(S,u) = \{v \in V | (s,v) \in E, s \in S, s \neq u\}$$

Define directed graph Ω_G on the collection of vertex sets as

$$\Omega_G = (\mathcal{P}(V), E')$$

$$E' = \{(S,T) | \rho(S,u) = T, u \in V\}$$

For any dirty region S and vertex u, $\rho(S,u)$ is the next dirty region resulting from searching vertex u. So, Ω_G is the road map for cleaning. It shows which dirt arrangements can be reduced to each other. We would like to say that a cleaning route through this map is also an extermination sequence, and vice versa.

As an example, Figure 4.10 provides the road map for path graph P_3. Vertex sets are represented by the black region of the graph. The two-edge path $V \to \emptyset$ corresponds to the two-vertex extermination sequence for P_3. Since we don't care too much about the vertex labels, and vertex sets are fundamentally identical with automorphic copies, we may just as easily work with Ω_G up to automorphism, for readability. Such a condensed road map is provided in Figure 4.11.

Proposition 4.12. *S is sub-exterminable iff there is a (directed) walk from S to $\emptyset$ in Ω_G.*

Proof. Suppose there is no (directed) walk from S to $\emptyset$ in Ω_G. Let $(s_n)_{n=1}^{k}$ be a k-vertex sequence, and consider the sequence $(t_n)_{n=1}^{k+1}$ in Ω_G constructed by:

$$t_1 = S$$

$$t_{n+1} = f(t_n, s_n)$$

By definition of the edge set of Ω_G, each (t_n, t_n+1) is an edge in Ω_G, so t is a walk. Then by the assumption, $t_{k+1} \neq \emptyset$. Let $x_{k+1} \in t_{k+1}$.

Construct a k-vertex walk x_n through G by the following induction. For each $n \in k, ..., 1$, we maintain that $x_{n+1} \in t_{n+1}$. By choice of x_{k+1} this holds for the base case $n=k$. For smaller n, since we have

$$x_{n+1} \in t_{n+1} = f(t_n, s_n) = \{v \in V | (x,v) \in E, x \in t_n, x \neq s_n\}$$

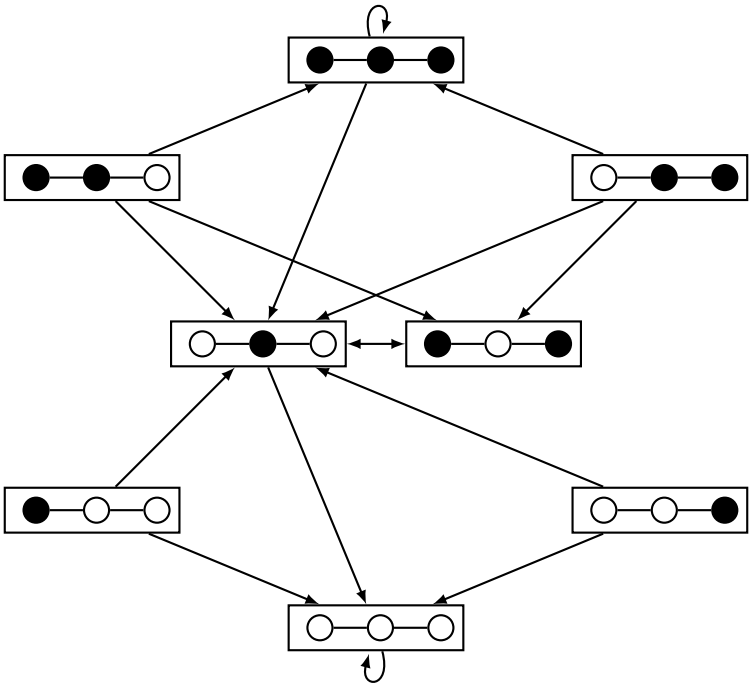

FIGURE 4.10. Ω_{P_3}

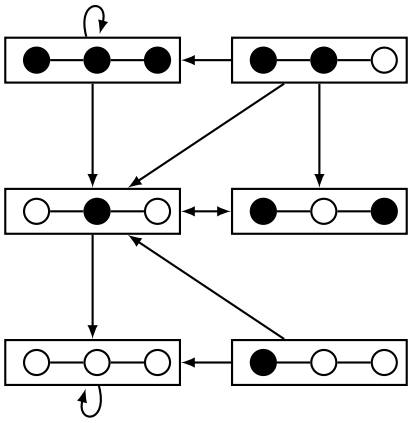

FIGURE 4.11. Ω_{P_3} up to automorphism

there must be some $v \in t_n, v \neq s_n$ such that $(v, x_{n+1}) \in E$. Define x_n to be this v. Then x is a walk in G and each $x_n \in t_n$ as promised.

Also, since each $x_n \neq s_n$, we have constructed a walk x_n beginning at $x_1 \in t_1 = S$ not intersecting the sequence s_n, so s_n cannot be a sub-extermination sequence. s_n was arbitrary, so S has no sub-extermination sequence. This concludes the forward direction.

Suppose Ω_G admits a (directed) path from S to $\emptyset$, and call this path $(t_n)_{n=1}^{k+1}$, of length k. Since each $(t_n, t_{n+1}) \in E$, we know that at each n, there is some s_n satisfying

$$t_{n+1} = f(t_n, s_n)$$

This defines a k-vertex sequence $(s_n)_{n=1}^{k}$ in G.

Let $(x_n)_{n=1}^{k}$ be an arbitrary k-vertex walk through G beginning in S. Since $x_1 \in S = t_1$ and $x_n \notin \emptyset = t_{k+1}$, there is some minimum m at which

$$x_m \in t_m, x_{m+1} \notin t_{m+1}$$

By the construction of t, $t_{m+1} = f(t_m, s_m)$. If s_m were not x_m, then the presence of the edge (x_m, x_{m+1}) would force the inclusion of x_{m+1} into t_{m+1}, so we observe $s_m = x_m$.

Since x was an arbitrary k-vertex walk beginning in S, s is a sub-extermination sequence for S, and S is sub-exterminable. □

The above bijection between extermination sequences and paths through Ω_G has the property that length is retained. For this reason, the shortest extermination sequence and the shortest path from V to $\emptyset$ have (approximately) the same size.

Corollary 4.13. *Any shortest extermination sequence in G has size either $d_{\Omega_G}(V, \emptyset)$ or $d_{\Omega_G}(V, \emptyset) + 1$.*

Proof. There are two reasons why a k-vertex sequence $(s_n)_{n=1}^{k}$ might result in a path from V to $\emptyset$ in Ω_G. Either A) every possible walk has been intersected by step k, or B) all the possible walks that have not yet been intersected by step k have nowhere to go next turn (they are at a vertex with out-degree $d^+ = 0$). Either way, the next dirty region (represented by t_{k+1} in the above proof) is the empty set.

In type A, s is an extermination sequence. In type B, we must append a vertex to the end of s. The choice of vertex does not matter. This is because s does not intersect every k-vertex walk through G, however there are no $(k+1)$-vertex walks through G which don't intersect s in the first k steps.

If there is a type A s which constructs a shortest path from V to $\emptyset$ in Ω_G, then this s is a shortest extermination sequence, and it has $d_{\Omega_G}(V, \emptyset)$ vertices. Otherwise, there are only type B sequences constructing shortest paths from V to $\emptyset$ in Ω_G, and the shortest extermination sequence has $d_{\Omega_G}(V, \emptyset) + 1$ vertices. □

The contrapositive of Proposition 4.12 is useful to prove that certain graphs are not exterminable. It is stated below for convenience.

Corollary 4.14. *G is not exterminable whenever some path-component in Ω_G does not contain $\emptyset$.*

We now begin to classify graphs with respect to exterminability. We find two classes which cannot be exterminable and one class which is. This will be enough to describe all undirected graphs.

Proposition 4.15. *If G has no vertex with in-degree ≤ 1 (if minimum in-degree $\delta^- \geq 2$), then G is not exterminable.*

Proof. Using Corollary 4.14, it suffices to find a family of vertex sets closed under arrows in Ω_G, not containing $\emptyset$.

Consider the (singleton) collection $\{V\}$. Pick arbitrary $x, y \in V$. Since y has in-degree ≥ 2, it has at least two in-neighbors, so at least one is not x. Call this vertex z.

$$(z, y) \in E, z \neq x \implies y \in f(V, x)$$

Since y was arbitrary, $V \subset f(V, x)$. Then for any x, the arrow in Ω_G generated by x points back to V, so our collection is closed, proving the claim. □

Corollary 4.16. *No undirected cycle is exterminable. Any exterminable undirected graph is acyclic.*

Proof. Corollary of above (cycles are 2-regular graphs). The second claim follows from incorporating Proposition 4.3. □

This concludes the intuitive proofs about exterminable graphs, i.e. the results promised in the beginning of this section.

The remainder of the section will rely heavily on results from Section 3 about spinewidth. It will be shown that all trees with spinewidth greater than 2 are not exterminable, whereas all trees with spinewidth at most 2 are. The latter is a constructive proof, and supplies an exact extermination sequence.

Proposition 4.17. *T_3 is not exterminable.*

Proof. Using Corollary 4.14, it suffices to find a family of vertex sets closed under directed edges in Ω_G (i.e. a path-component of some S) not containing $\emptyset$.

Consider the family depicted by Figure 4.18. Vertex sets are represented by the black regions in the graph. All arrows (up to automorphism) in Ω_{T_3} beginning in the family depicted are included in the figure.

Since there is no path to $\emptyset$, T_3 is not exterminable. □

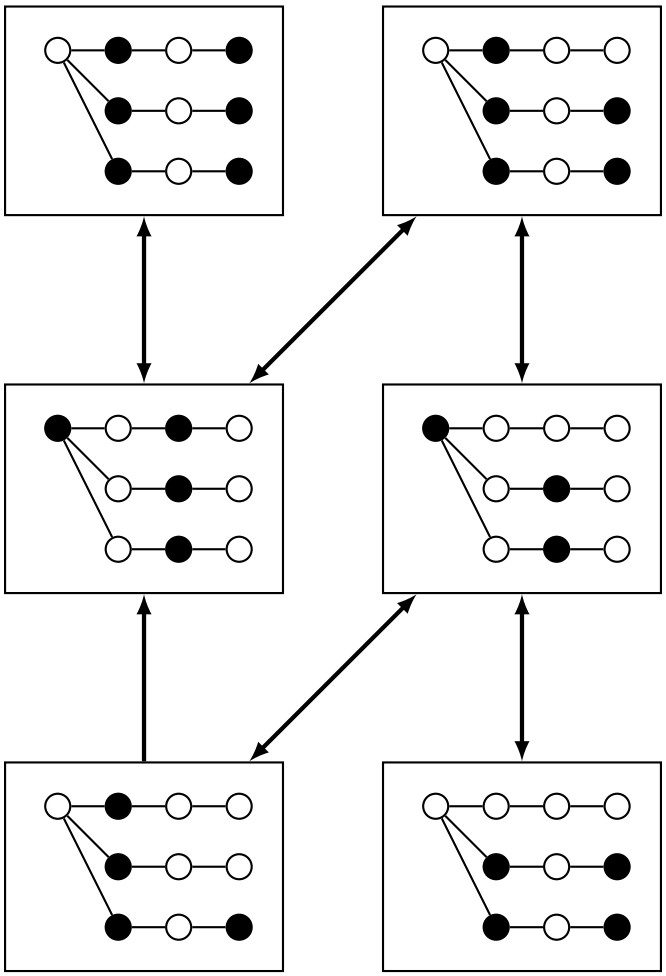

FIGURE 4.18. Path component in Ω_{T_3} not containing $\emptyset$

Proposition 4.19. *If tree G has spinewidth ≤ 2, then G is exterminable.*

Proof. First consider $G = K_1$. Let $V = \{v\}$. The 1-vertex sequence (v) intersects the only 1-vertex walk $t = (v)$.

Next consider $G = K_2$. Let $V = \{u, v\}$. The 2-vertex sequence (v, v) intersects the two 2-vertex walks $t = (u, v)$ and $t' = (v, u)$.

Finally consider when G is neither K_1 nor K_2, so by Lemma 3.18 there is some non-leaf vertex and the interior of G is nonempty.

$$I(G) \neq \emptyset$$

If G has spinewidth 0, it is a pathgraph P_n for $n \geq 3$, and $I(G)$ is also a path graph P_{n-2}. Otherwise, by Corollary 3.17 and Proposition 3.19, $I(G)$ is a tree of spinewidth ≤ 1. In either case, $I(G)$ is a tree of spinewidth ≤ 1.

Let $(p_i)_{i=1}^{l}$ be a longest path through $I(G)$, say with l vertices. By Proposition 3.21, p contains the minimum spine of $I(G)$, which by Proposition 3.19 is also a spine of G, so p is a spine of both $I(G)$ and G. G is bipartite (G is a tree), so

let A be the part containing p_1. For each i in $1, ..., l$, define k_i to be the number of non-p neighbors of p_i (in $I(G)$), and label these neighbors as $v_{i,j}$ for $j = 1, ..., k_i$.

Let q_i be the partial sums of $2k_i + 1$.

$$q_i = \sum_{j=1}^{i} (2k_j + 1)$$

Define sequence $(s_n)_{n=1}^{q_l}$ in G:

$$1 \leq n \leq 2k_i + 1 \implies s_{q_{i-1}+n} = \begin{cases} v_{i,n/2} & n \text{ even} \\ p_i & n \text{ odd} \end{cases}$$

Visually, this sequence is a walk, a contour tracing of $I(G)$, as illustrated by Figure 4.20. The example graph G used in Figure 4.20 and Figure 4.21 is the same lobster from Figure 1.4.

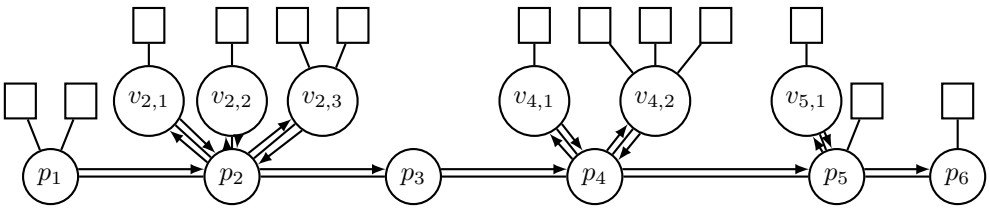

FIGURE 4.20. Contour tracing of $I(G)$

I claim this sequence is a sub-extermination sequence for part A. It would then follow by Proposition 4.8 that G is exterminable.

By Lemma 3.4, to each vertex v in G there is a unique nearest p_z, so we label each vertex by its nearest spine point's index, by defining $\varphi : G \to \mathbb{N}$ as $\varphi(v) = z$. A example of a φ-labelling is provided in Figure 4.21

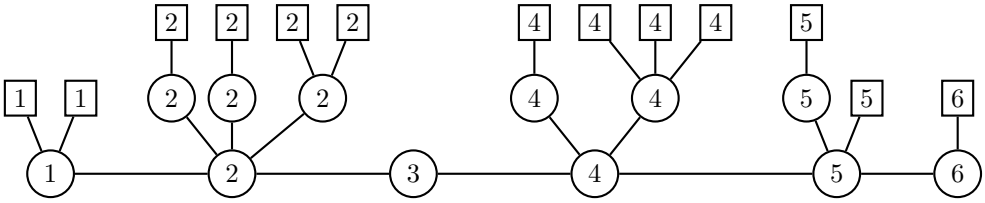

FIGURE 4.21. A φ-labelling of G

Suppose $(t_n)_{n=1}^{q_l}$ is a q_l-vertex walk beginning in A. Since walks alternate parts of a bipartite graph, and since s and t begin in the same part, they will always be in the same part. Consider arbitrary edge (t_n, t_{n+1}) on walk t.

Case I) Both t_n and t_{n+1} are p-points.
Then their edge lies on p, i.e. $t_n = p_i, t_{n+1} = p_{i\pm1}$, or else the 1-long (t_n, t_{n+1}) would be a simple path $t_n \to t_{n+1}$ other than the path lying entirely on p (impossible).

Case II) At least one is not a p-point.
Then this vertex has distance ≥ 1 from p, so we may apply Lemma 3.5 and see that t_n and t_{n+1} share the same nearest p-point. In other words $\varphi(t_n) = \varphi(t_{n+1})$.

Thus, on any walk t, the only time $\varphi(t_n)$ changes is when t walks along p. Now, assume t never intersects s.

$$\forall n : t_n \neq s_n \tag{4.22}$$

Then:

$$t_1 \neq s_1 = p_1$$
$$t_{q_l} \neq s_{q_l} = p_l$$

Since p is a longest path in tree $I(G)$, its endpoints can have no non-p neighbors in $I(G)$, or else p could be extended to a path with longer length. So, the endpoints of p have no non-leaf neighbors in G. That is, the only vertices v with $\varphi(v) = 1$ are p_1 and neighbors of p_1. Of these, the only vertex in the same part as p_1 is p_1. Likewise for p_l. This tells us:

$$\varphi(t_1) > 1 = \varphi(s_1)$$
$$\varphi(t_{q_l}) < l = \varphi(s_{q_l})$$

Then there is some minimum m at which $\varphi(t_m) \leq \varphi(s_m)$.

First, observe that $\varphi(t_m) = \varphi(s_m)$. Since φ of a walk changes by at most 1 per step, the only other possibility for m would be both s and t walk along p and pass through each other at step m, at edge (p_w, p_{w+1}) for some w:

$$s_{m-1} = p_w \implies \varphi(s_{m-1}) = w$$
$$t_{m-1} = p_{w+1} \implies \varphi(t_{m-1}) = w + 1$$
$$s_m = p_{w+1} \implies \varphi(s_m) = w + 1$$
$$t_m = p_w \implies \varphi(t_m) = w$$

But then s_m and t_m are neighbors, and must lie on different parts of bipartite G, a contradiction. So, we do have $\varphi(t_m) = \varphi(s_m)$. Call this shared value w.

Because m is the first occasion at which $\varphi(t_m) = \varphi(s_m)$, one of these values must have just changed, either a decrease from t or an increase from s (or both).

In the first case, we must have $t_m = p_w$, since $\varphi(t_m)$ can only change while walking along p. But the only possible s_m which satisfies both A) s_m and t_m lie in the same part, and B) $\varphi(s_m) = w$, is $s_m = p_w$, so t and s intersect, a contradiction.

Otherwise $\varphi(t_m) = \varphi(t_{m-1})$ and $s_m = p_w, s_{m-1} = p_{w-1}$. (In fact, the only m at which this s step occurs is $m = q_{w-1} + 1$.) Since A) t_m lies on the same part as s_m, and B) $\varphi(t_m) = w$, we must have that t_m is a leaf of some $v_{w,j}$.

For the next $2k_w$ steps, t cannot intersect s, so for any $n \leq k_w$, $t_{m+2n} \neq s_{m+2n} = p_w$. Then t is restricted to $v_{w,j}$ and its leaves. Furthermore, it must alternate between these two options. But this means

$$t_{m+2j-1} = v_{w,j} = s_{q_{w-1}+2j} = s_{m+2j-1}$$

so t intersects s, a contradiction.

All cases are exhausted, so our assumption (4.22) was incorrect, and every q_l-vertex walk t beginning in A must intersect s. In other words, s is a sub-extermination sequence for part A of bipartite G, and therefore G is exterminable. □

Theorem 4.23. *A connected graph is exterminable iff it is a lobster.*

Proof. Recall that a lobster is just a tree with spinewidth ≤ 2.

Suppose connected G is a lobster. Then by Proposition 4.19 it is exterminable

Now suppose connected G is exterminable. By Corollary 4.16 G is acyclic, so G is a tree. By Proposition 4.3 and Proposition 4.17 G does not contain T_3. By Proposition 3.13 G has spinewidth ≤ 2. In other words, G is a lobster. □

Corollary 4.24. *An undirected graph is exterminable iff every component is a lobster.*

Proof. Combines Theorem 4.23 with Corollary 4.5. □

5. Future Directions

This problem, and other variations, are still ripe with undiscovered phenomena. Notably, this paper leaves unsolved the problem on general directed graphs. Provided below are two more tools for analyzing such graphs and their exterminability.

First, a directed graph can be reversed and remain exterminable. Recall the graph transpose, or G^T, is the graph constructed by reversing all arrows.

Proposition 5.1. *G is exterminable iff G^T is exterminable.*

Proof. By symmetry, it suffices to prove only the forward direction. So, suppose G is exterminable. Let $(s_n)_{n=0}^k$ be an extermination sequence for G. Define sequence $(s'_n)_{n=0}^k$ as the reversal of s

$$s'_n = s_{k-n}$$

Let $(t'_n)_{n=0}^k$ be a $(k+1)$-vertex walk through G^T. Define sequence $(t_n)_{n=0}^k$ in G as the reversal of t'

$$t_n = t'_{k-n}$$

Observe that t_n is a walk through G:

$$(t_{n+1}, t_n) = (t'_{k-n-1}, t'_{k-n}) \in E(G^T) \implies (t_n, t_{n+1}) \in E(G)$$

Then at some step m, t intersects s. Thus, at $k-m$, s' intersects t':

$$s'_{k-m} = s_m = t_m = t'_{k-m}$$

Hence s' is an extermination sequence for G^T, and G^T is exterminable. □

Second, graphs can be transmuted via an action by their automorphism group, and remain exterminable.

We must define this action of the symmetric group $\mathfrak{S}_n$ on the set of graphs with n vertices. The typical "vertex-shuffling" action attributed to $\mathfrak{S}_n$ permutes the n vertices, which carry their arrows with them. This has the result that for any graph G and any permutation σ, G is isomorphic to $\sigma(G)$. No graph can be permuted to a "different" graph. However, the new action defined below, an "arrow-shuffling" action, does not have this property. It is able to transmute a graph into a "different" graph. We will differentiate between these two actions by writing $\nu(\sigma, G)$ to mean a vertex-shuffled G, and $\alpha(\sigma, G)$ to mean an arrow-shuffled G.

Definition 5.2. Define left action α of $\mathfrak{S}_n$ on n-vertex directed graphs. Let $\sigma \in \mathfrak{S}_n$ and let G be a graph with n vertices. The vertex set of $\alpha(\sigma, G)$ is defined to be the same vertex set as G. The edge set is defined as

$$(u, v) \in E(G) \iff (u, \sigma(v)) \in E(\alpha(\sigma, G))$$

To be clear, this is not the typical action of the symmetric group on G. Usually, we consider permutations of vertices. This new action is different: it keeps vertices fixed but moves the targets of arrows.

For instance, consider the following automorphism σ on the 8-node 1-regular graph, depicted by Figure 5.3. The transformation fixes the first edge, reflects the second, and swaps the third with the fourth.

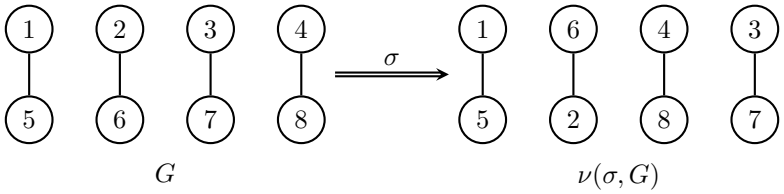

FIGURE 5.3. Specific automorphism on 8-node 1-regular graph

The transmuted graph $\alpha(\sigma, G)$, under the arrow-shuffling action, would then be the graph depicted in Figure 5.4. Notice this new graph is not isomorphic to the original.

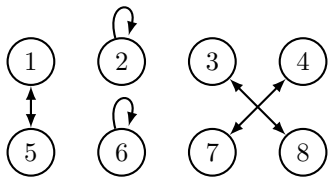

FIGURE 5.4. $\alpha(\sigma, G)$

This action is especially useful when we restrict to the automorphism group $\mathrm{Aut}(G)$ of a graph with n vertices, i.e. the subgroup of $\mathfrak{S}_n$ which is trivial under the vertex-shuffling action. This definition of automorphism group coincides with the standard definition for graphs.

Proposition 5.5. *G is exterminable iff $\alpha(\sigma, G)$ is exterminable for all $\sigma \in \mathrm{Aut}(G)$.*

Proof. Fix $\sigma \in \mathrm{Aut}(G)$. First, there is a bijection between sequences in G and sequences in $\alpha(\sigma, G)$ which preserves walks.

For sequence $(s_n)_{n=0}^k$ in G, for any k, define $\Psi(s)_n$ as

$$\Psi(s)_n = \sigma^n(s_n)$$

Clearly the original s_n can be recovered from $\Psi(s)_n$ by applying σ^{-n}. The existence of an inverse mapping proves bijectivity.

This mapping Ψ of sequences preserves intersections.

$$s_n = t_n \implies \Psi(s)_n = \sigma^n(s_n) = \sigma^n(t_n) = \Psi(t)_n$$

It also preserves walks.

$$(s_n, s_{n+1}) \in E(G) \implies (\sigma^n(s_n), \sigma^n(s_{n+1})) \in E(G)$$
$$\implies (\Psi(s)_n, \Psi(s)_{n+1}) = (\sigma^n(s_n), \sigma^{n+1}(s_{n+1})) \in E(\alpha(\sigma, G))$$

So, if sequence s intersects every walk through G, then $\Psi(s)$ intersects every walk through $\alpha(\sigma, G)$.

The backwards direction of the claim is trivial, since the identity of $\mathrm{Aut}(G)$ has trivial action on G. □

Neither property found use in the paper, although they are not without their merits. For instance, they can be used to show the graphs of Figure 5.6, Figure 5.7, and Figure 5.8 are equally exterminable.

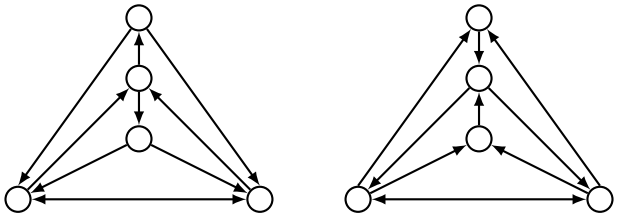

FIGURE 5.6. Equally exterminable graphs, Collection I

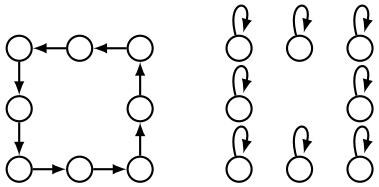

FIGURE 5.7. Equally exterminable graphs, Collection II

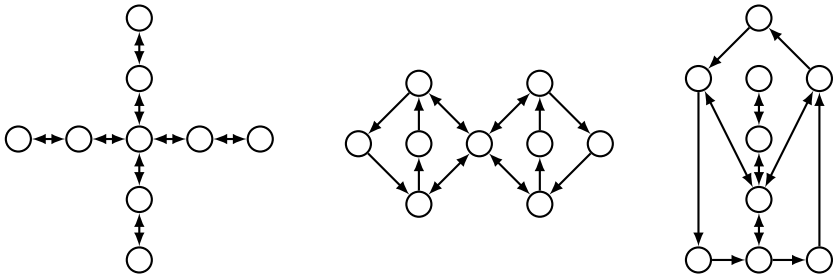

FIGURE 5.8. Equally exterminable graphs, Collection III

The common weakness in these methods is that they apply too rarely. The first only equates the exterminabilities of two graphs, and when the original graph is undirected it is entirely useless. The second relies on symmetries inherent in G, which are rather rare across general graphs.

The original rat problem can be further generalized by asking "What if a team of n exterminators was hired? How many exterminators does this building require?" A formalization for this property is given below.

Definition 5.9. An **n-extermination sequence** is an n-large collection of k-vertex sequences $(s_{j,i})_{i=1}^{k}$ through G, one for each $j = 1, ..., n$, such that every k-vertex walk $(t_i)_{i=1}^{k}$ in G intersects at least one sequence ($s_{j,i} = t_i$ for some j and i). Equivalently, it is a k-large sequence S_i of vertex sets, each of size at most n, such that every k-vertex walk $(t_i)_{i=1}^{k}$ in G lies in one of the sets at some step ($t_i \in S_i$ for some i). A graph G is **n-exterminable** if n is the smallest number such that G admits an n-extermination sequence.

Many of our results can be very slightly tweaked to say more in this new language. Some examples are listed below.

Proposition 5.10. *Suppose G is n-exterminable.*

(i) Every subgraph of G is $(\leq n)$-exterminable.
(ii) G has an n-exterminable strongly connected component.
(iii) G^T is n-exterminable.
(iv) $n \geq \delta^-$ (the minimum in-degree).
(v) $n \geq \delta^+$ (the minimum out-degree).
(vi) $n \leq 1$ iff G is exterminable.

We may further attain some upper and lower bounds on n, relating this number to some established graph-theoretic properties.

Proposition 5.11. *G is 0-exterminable iff G is a directed acyclic graph.*

Proof. The only graph for which every k-long walk intersects one of an empty set of sequences is a graph with no k-long walks for large enough k. This class of graphs is the well-established "directed acyclic graph." □

Proposition 5.12. *Suppose G is n-exterminable. Then $n \leq V - \alpha(G)$, where $\alpha(G)$ is the anticlique number of G.*

Proof. Let A be an anticlique of size $\alpha(G)$, and let B be its complement of size $V - \alpha(G)$. The $(V - \alpha(G))$-extermination sequence (B, B) intersects every 2-vertex walk, since no edge goes from A to A. □

Corollary 5.13. *Complete graph K_a is $(a-1)$-exterminable.*

Proof. K_a is $(a-1)$-regular, so $n \geq a - 1$. K_a has anticlique number $\alpha(K_a) = 1$ (every node is loop-less), so $n \leq a - 1$. □

Corollary 5.14. *Suppose G is n-exterminable. Then $n \geq \omega(G) - 1$, where $\omega(G)$ is the clique number of G.*

Proof. Subgraphs require less or equally many exterminators as the whole graph, by Proposition 5.10. The clique in G of size $\omega(G)$ requires $\omega(G) - 1$. Then the whole graph requires at least that many. □

Corollary 5.15. *If $a \leq b$, then complete bipartite graph $K_{a,b}$ is a-exterminable.*

Proof. $K_{a,b}$ has minimum degree $\delta(K_{a,b}) = a$, so $n \geq a$. $K_{a,b}$ has anticlique number $\alpha(K_{a,b}) = b$, so $n \leq (a+b) - b = a$. □

The following are two unproven claims on n-exterminable graphs.

Conjecture 5.16. *An $m \times n$ 2-dimensional grid, for $m \leq n$, is $(\lfloor \frac{m}{2} \rfloor + 1)$-exterminable. More generally, an $n_1 \times \cdots \times n_d$ d-dim grid, for $n_1 \leq \cdots \leq n_d$, is $\left(\left\lfloor \frac{V}{2n_d} \right\rfloor + 1\right)$-exterminable, where $V = \prod_{i=1}^{d} n_i$.*

This value is certainly an upper bound. However, it is difficult to prove it is a lower bound.

Conjecture 5.17. *For every $n \geq 0$ there is an n-exterminable tree.*

This should hold because a complete rooted binary tree of arbitrarily high depth should require arbitrarily many exterminators.

One application of this framework is on Markov chains. Suppose we have a Markov process, and we wish to discover which state the process is in. Assume that the number of states we are able to check per turn of the process does not depend on which states we check, and call this number n. This goal is accomplishable in finite time exactly when the corresponding directed graph (weights do not matter for now) is n-exterminable.

However, this is not a complete picture. Markov chains are not simply progressions of possible states, they have accompanying probabilities. What if we also cared about the expected number of turns before the conclusion of our search. It is conceivable that a search strategy which prioritizes worst-case duration will not prioritize expected-case duration, and vice versa.

Indeed, this is often the case. Provided below are a few simple examples of this behavior.

Definition 5.18. Define $X_{x \times y}$ to be the tree which consists of x copies of the path graph P_y, called axes, and a node adjacent to the head of every copy. Enumerate the path graph vertices as $v_{i,j}$, for $i = 1, ..., x$ and $j = 1, ..., y$. Name the center vertex o, adjacent to each $v_{i,1}$.

It is easy to see this is an extension of the symmetric T-shaped tree family constructed in Definition 3.11:

$$T_n = X_{3 \times n}$$

Example 5.20. Consider the following two strategies on $X_{x \times 2}$. One is greedy (always searching the most likely room), one is methodical (with a guarantee of eventual capture). To calculate expected probabilities, we use random walks on the graph. Let $(g_n)_{n=0}^{\infty}$ be the constant sequence given by $g_n = o$ for all n. Let (s_n) be the extermination sequence of $X_{x \times 2}$ as constructed by Proposition 4.19.

Then the respective durations, expected and maximum, are

	g	s
Maximum	∞	$4x - 2$
Expected	$\frac{9x+1}{2x+1} \approx 4.5$	$\frac{2x^3 + 9x^2 - 9x - 6 + (12x+6) \cdot 2^{-x}}{2x^2 + x} \approx x + 4$

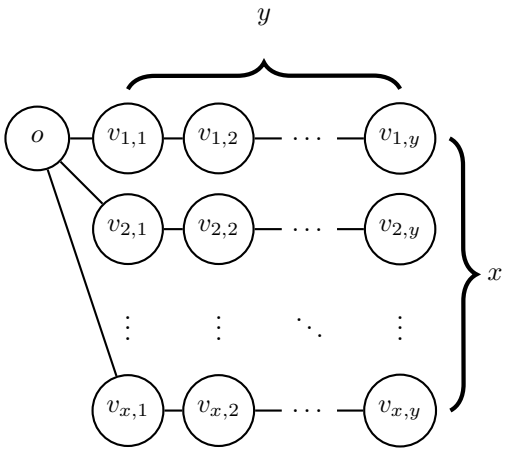

FIGURE 5.19. $X_{x\times y}$

In fact, the infinitely many mixes between these two strategies, i.e. those which check only the center vertex some finite number of times, then proceed with the methodical strategy, are all mutually incomparable (none is strictly preferable to another). They may even all be "maximal," in the sense that no strategy is strictly better than any of them, better in both the expected case and the worst case simultaneously.

Example 5.21. Let $\alpha \in [0,1]$ be arbitrary. Consider the maximal strategies on the weighted directed graph in Figure 5.22. We begin with initial distribution $P(x) = \alpha, P(y) = 1-\alpha$.

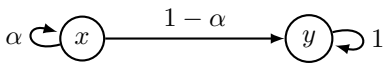

FIGURE 5.22. Weighted Directed Graph

The strategies which visit x twice are never maximal. This is because after the first visit to x, there is 0% chance of the rat being there ever again. Therefore, upon visiting x, a maximal strategy will visit y once, then terminate (at this point the rat will have been caught with 100% certainty).

This means that every maximal strategy is uniquely defined by how many times it visits y before visiting x. Call this number n. If it never visits x (the unique constant y strategy) let us say $n = \infty$.

	General n	$n = 0$	$n = \infty$
Maximum	$n+2$	2	∞
Expected	$\frac{1+\alpha^n-3\alpha^{n+1}+\alpha^{n+2}}{1-\alpha}$	$2-\alpha$	$\frac{1}{1-\alpha}$

We can condense this into three cases, depending on the relative size of α and $2-\phi \approx 0.382$, where ϕ is the golden ratio $\phi = \frac{1+\sqrt{5}}{2} \approx 1.618$.

Case I) $\alpha > 2 - \phi$
The expectation curve is monotonically increasing. The longer the strategy waits to search vertex x, the worse the expected search duration is. This agrees with the maximum duration trend, so the only maximal strategy is to immediately search vertex x.
Maximal values of n: $\{0\}$.

Case II) $\alpha = 2 - \phi$
The expectation curve is flat. It has constant value ϕ. No matter how long we wait to search vertex x, we expect to terminate after ϕ searches. Since all strategies have equal expected duration, the maximality condition is solely dependent on maximum duration. Then the only maximal strategy is to immediately search vertex x.
Maximal values of n: $\{0\}$.

Case III) $\alpha < 2 - \phi$
The expectation curve is monotonically decreasing. The longer the strategy waits to search vertex x, the better the expected search duration is, approaching a value of $\frac{1}{1-\alpha}$, which is attained by the constant-y strategy. This goes directly against the maximum duration trend, so every strategy is maximal. There are infinitely many maximal strategies.
Maximal values of n: $\mathbb{N} \cup \{\infty\}$.

Even without turning to probability and random walks, one might be curious about an extermination sequence of minimum length. Corollary 4.13 provides a tool to find this length on a chosen graph, although it requires finding a shortest path through a directed graph of non-polynomial size.

I believe that the algorithm constructed in Proposition 4.19 is close to optimal in its total duration, but I do not provide a proof of this claim. Recall this full sequence is the concatenation of one sequence constructed per each component, each of which is the interior contour tracing plus its reversal immediately after.

Conjecture 5.23. *For isolated node K_1, the sole 1-vertex sequence is a shortest extermination sequence.*

For other trivial (edgeless) G, any 2-vertex sequence is a shortest extermination sequence.

For nontrivial undirected graph G whose components are lobsters, the extermination sequence constructed by Proposition 4.19, modified so every trivial component is skipped, has minimum duration.

Acknowledgments

It is a pleasure to thank my mentor, Yao Luo, for guiding me through writing my first paper and steering me in more interesting and worthwhile directions. I thank Loren Troan for generalizing the rat problem to multiple exterminators, and for posing the grid problem (Conjecture 5.16). It has come to the attention of the author that the characterization of exterminable undirected graphs was previously obtained by Christian Lawson-Perfect in [5].

References

[1] A. Bonato. "A new prehistory of Cops and Robbers." The Intrepid Mathematician. April 4, 2018. https://anthonybonato.com/2018/04/04/a-new-prehistory-of-cops-and-robbers/

[2] A. Bonato, R. Nowakowski. *The Game of Cops and Robbers on Graphs*. The Student Mathematical Library, Volume 61, 2011.
http://dx.doi.org/10.1090/stml/061
[3] A. Kehagias, D. Mitsche, P. Pralat. "The Role of Visibility in Pursuit / Evasion Games." February 25, 2014. https://arxiv.org/abs/1402.6136.
[4] A. Quilliot. "Jeux et pointes fixes sur les graphes." Thèse de 3ème cycle, Université de Paris VI, 1978, 131–145.
[5] C. Perfect. "Solving the 'princess on a graph' puzzle." Check My Working. December 15, 2011. https://www.checkmyworking.com/2011/12/solving-the-princess-on-a-graph-puzzle/
[6] M. Adler, H. Räcke, N. Sivadasan, C. Sohler, B. Vöcking. "Randomized Pursuit-Evasion in Graphs." *Automata, Languages and Programming*. ICALP 2002. Lecture Notes in Computer Science, vol 2380. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45465-9_77
[7] M. Casini, A. Garulli. "A new class of pursuer strategies for the discrete-time lion and man problem." *Automatica*, Volume 100, 2019, 162-170.
https://doi.org/10.1016/j.automatica.2018.11.015
[8] N. Stiffler, J. O'Kane. "Complete and optimal visibility-based pursuit-evasion." The International Journal of Robotics Research, 2017, Vol. 36(8) 923–946.
https://doi.org/10.1177/0278364917711535
[9] P. Nahin. *Chases and Escapes: The Mathematics of Pursuit and Evasion*. Princeton University Press, 2012.
[10] R. Nowakowski, P. Winkler. "Vertex to vertex pursuit in a graph." Discrete Mathematics 43 (1983) 230–239.
[11] T. Chung, G. Hollinger, V. Isler. "Search and pursuit-evasion in mobile robotics." *Autonomous Robots* 31, 299–316. Springer Science + Business Media, 2011.
https://doi.org/10.1007/s10514-011-9241-4
[12] T. Parsons. "Pursuit-evasion in a graph." *Theory and Applications of Graphs*. Lecture Notes in Mathematics, vol 642, 1978. Springer, Berlin, Heidelberg.
https://doi.org/10.1007/BFb0070400
[13] V. Isler, N. Karnad. "The role of information in the cop-robber game." Theoretical Computer Science 399 (2008) 179–190.