Graph Searching Methods for Hide-and-Search Games on the Examples of Scotland Yard and Letters from Whitechapel

Obszyńska, Alicja D.

Award date:
2023

Graph Searching Methods for Hide-and-Search Games on the Examples of *Scotland Yard* and *Letters from Whitechapel*

Author: Alicja Obszyńska
Supervisor: Bart Smeulders

July 3, 2023

Abstract

Inspired by the hide-and-search board games *Scotland Yard* and *Letters from Whitechapel*, we are looking for an optimal strategy for the detectives. We focus on a simplified game on starlike trees, where the goal of the detectives is to find the criminal's hideout in as few moves as possible. We start with finding optimal strategies for special cases of such trees, namely, path graphs and stars, in order to build up to the general case of starlike trees. Finally, we suggest possible further steps for more complex graphs or different rules for the games.

Eindhoven University of Technology
Department of Applied Mathematics and Computer Science

Contents

1 Introduction **1**
- 1.1 *Scotland Yard* and *Letters from Whitechapel* 1
- 1.2 The Simplified Game 3

2 Preliminaries **3**
- 2.1 Notation 3
- 2.2 Variables 4

3 Problem Statement **4**

4 Path Graphs **5**
- 4.1 S_C in an Endpoint 6
- 4.2 S_C in the Centre 9
- 4.3 Other S_C 11
- 4.4 General S_C 14

5 Stars **15**

6 Trees **18**

7 Conclusion and Recommendations **23**
- 7.1 Further Research 23

8 Acknowledgements **24**

1 Introduction

Mathematics is employed in every board game in the world and some of them make use of graph theory. We will focus on one type of them - cops and robbers games, as they are arguably the most famous graph-searching games.

There are many variants of cops and robbers games, the classical one being a two-player game of perfect information, where one player is the robber and the other one is the cops and they can see all the moves. The goal of the cops is to capture the robber, and the goal of the robber is to run away. The board that they move on can be seen as a graph, where each vertex is a spot where they can stand and the edges correspond to the possible moves between vertices. There is also a possibility to pass, which means that the player stays in their current position. The players play in turns, moving along the edges between neighbouring vertices or deciding to pass [1].

We will focus on a different variant of the game, however. The games that we will consider are games of *imperfect information*, where the moves of the robber are not known to the cops. These are also known as *hide-and-search* games, where the cops would be the *seekers* and the robber - the *hider* [2].

1.1 *Scotland Yard* and *Letters from Whitechapel*

Two modern-day popular board games that we will focus on are *Scotland Yard* and *Letters from Whitechapel*. In both of them, we have *detectives* as the seekers and a *criminal* (Mr X in *Scotland Yard* and Jack the Ripper in *Letters from Whitechapel*) as the hider. We can have multiple players play as the detectives and one player as the criminal. Nevertheless, we consider these games as 1v1, as all the detectives have to cooperate or can be controlled by one person.

Both of them are games of imperfect information. However, the games differ in what information is revealed to the detectives. In *Scotland Yard*, the current location of the criminal is known every few turns and in *Letters from Whitechapel* the detectives check if the criminal was in the vertices around them.

Moreover, in *Scotland Yard*, there are three different means of transportation - depending which one the players use, they can get to different vertices. This can be seen in Figure 1, where different colours of edges correspond to different transportation methods - taxi (yellow), bus (green) and underground (red). The choice of transportation for each move of the criminal is also known to the detectives and it is marked as shown in the picture. The detectives check if the criminal is in the same vertex as they are in and if one of them is - they win, the criminal wins if he manages to escape the detectives for 23 turns.

In *Letters from Whitechapel*, the graph that we consider is two graphs put together - there are different vertices for the criminal and different for the detectives. This can be seen in Figure 2, where the black squares are the vertices for the detectives and the circles with numbers are the vertices for the criminal.

Figure 1: *Scotland Yard* board [3].

The detectives have two actions that they can choose from - either check if the criminal previously visited their neighbouring vertices or if he is in one of the neighbouring vertices right now - if that is the case, the detectives win. If the criminal manages to get to his hideout (a vertex that he chooses at the beginning of the game) in 15 turns, he wins. If he does not manage to do that, the detectives win.

Figure 2: *Letters from Whitechapel* board [4].

There are a few additional rules in both of the games, which allow the criminal to move more in one turn or hide his moves. We will not take them into consideration, as they introduce more unknowns and thus make it much easier

for the criminal to win. It is important, as we are looking for winning strategies for the detectives. This is not entirely possible in the given setting, as the *Scotland Yard* game is proven to be NP-complete [5]. However, there are neural networks made to play this game - an adversarial neural network designed to play only *Scotland Yard* [6] and the "Player of Games" designed to play multiple perfect information games and two imperfect information games [7].

1.2 The Simplified Game

Since *Scotland Yard* is NP-complete, we will focus on smaller graphs and different cases with simplified rules that would be a mix of the rules of the two board games. It is worthwhile to summarise these rules.

- there is one criminal and one detective,
- the criminal first makes all his moves, then the detective makes all their moves,
- the detective wants to get to the criminal's hideout as quickly as possible,
- the criminal wants to make the detective move as much as possible before they reach his hideout,
- the detective gets to know if the criminal was in a given vertex or if that vertex is the final destination only when they are in that vertex.

2 Preliminaries

Before diving into the problem, we need to introduce some definitions and notations first. Moreover, we will focus on connected graphs, as we need them to be representative of a possible board for a game where both players can play together.

2.1 Notation

We have two players in the game - the criminal, denoted as C, and the detective, denoted as D. The starting position of the criminal is fixed, not chosen by the criminal, and is denoted by S_C. The starting position of the detective (later referred to as "starting position" for short) is chosen by the detective and denoted by S_D. The final destination of the criminal, F, is where the detective strives to reach. If our graph is $G = (V, E)$, then S_C, S_D, $F \in V$.

Both players need to make *moves* to reach the final destination. A single move is visiting a vertex in V and can be done only along one of the existing edges or by staying in the same vertex. A sequence of visited vertices, say $(v_0, v_1, \ldots, v_k)$ is a $v_0 - v_k$ *walk*, as v_i and v_{i+1} $(\in V)$ are connected for all $i = 0, 1, \ldots, k-1$. Such a walk is denoted by $w(v_0, v_k)$ The *length* of such a walk is then k and is denoted by $l(w(v_0, v_k))$. The $v_0 - v_k$ walk that includes only distinct vertices is

called a *path*. *Distance* between two vertices v_0, v_k is defined as the length of the shortest $v_0 - v_k$ path and is denoted by $d(v_0, v_k)$.

2.2 Variables

There are a few variables that are used throughout the report, we introduce some of them here:

- the number of criminal's moves is $t \in \mathbb{N}$ (and is known to the detective),
- the number of moves of the detective is $d \in \mathbb{N}$ (and is to be found),
- the distance from S_C to the furthest vertex is m, defined as $m := \max_{v \in V}\{d(S_C, v)\}$.

Any other variables are defined further on.

3 Problem Statement

As mentioned earlier, the criminal starts in one of the vertices, labelled S_C, and needs to choose a final destination F and a walk leading to it.

The goal of the criminal is to make the detective move as much as possible before reaching F. We will call that situation *the worst-case scenario*. This corresponds to a strategy of the detective, which consists of a set of rules that they follow when making moves and their starting position S_D.

The detective's goal is to move as little as possible before reaching F and for that, they need to choose the *optimal strategy*. We need to define what *the optimal starting position and set of rules* mean so that it is clear what we shall prove later on. Note that they need not be unique, so there can exist multiple optimal strategies for a given graph. We focus on finding one of them for multiple types of graphs.

Definition 3.1. An optimal starting position of the detective is an S_D that results in the smallest d in all worst-case scenarios for a given t and some set of rules.

Definition 3.2. An optimal set of rules is an algorithm that the detective follows while moving, which results in the smallest d in all worst-case scenarios for given t and S_D.

Definition 3.3. An optimal strategy is a combination of a set of rules according to which the detective moves and a starting point that results in the smallest d for a given t.

Moreover, let us define W as the set of all walks that the criminal can make and R as the set of rules that the detective follows. Then, for a $w \in W$, we can define $n(S_D, w, R)$ to be the length of the walk of D given a walk w of the

criminal, a starting position S_D and a set of rules R. The number of moves of D is then defined as

$$d = \max_{w \in W} n(w, S_D, R).$$

Our objective is to find an R and an S_D that will result in the smallest d for a given graph G, i.e. an optimal strategy for a given graph G.

4 Path Graphs

We will first focus on the basic cases so that we can build up on them at the further stages. We start with the path graphs and multiple variants for S_C.

Definition 4.1 (Path graph)**.** A path graph is a graph whose vertices can be listed in the order v_1, v_2, …, v_n such that the edges are $\{v_i, v_{i+1}\}$ where $i = 1, 2, \ldots, n-1$.

The set of rules for the detective (R) that we choose for path graphs can be expressed with the following algorithm.

Algorithm 1 The set of rules for path graphs.

```
1:  if current vertex is F then
2:      D stops
3:  else
4:      if current vertex is an endpoint then
5:          D moves towards the other endpoint
6:      else if current vertex is S_C then
7:          if current vertex is S_D then
8:              D moves to the longer side of the graph
9:          else
10:             D moves in the same direction as earlier
11:         end if
12:     else if current vertex is not visited by C then
13:         D moves towards S_C
14:     else if D started moving towards S_C then
15:         D moves towards S_C
16:     else
17:         D moves away from S_C
18:     end if
19: end if
```

We will prove that this set of rules is optimal later on. For now, we will focus on finding the optimal starting position for the detective and criminal's moves. We assume that the criminal knows what the strategy of the detective is and their moves are a response to that strategy.

We will investigate a few different cases of path graphs which differ by the choice of S_C.

4.1 S_C in an Endpoint

The first case of path graphs that we consider is one where S_C is one of the outermost vertices - referred to as the endpoints of the graph. Without loss of generality, we assume that it is always the leftmost vertex, as if it were the rightmost one we can always rotate the graph so that it is on the left.

We also assume that $t \geq m$, as if $t < m$ we could always take the subgraph consisting of vertices that are reachable from S_C, i.e. those that are at a distance at most t from S_C.

Moreover, for ease of notation, we will label all vertices as $v_i \in V$ for $i = 0, 1, \ldots, m$ such that $d(S_C, v_i) = i$. An example of such a path graph can be seen in Figure 3.

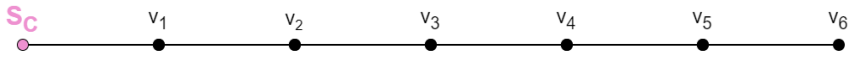

Figure 3: A path graph with S_C in an endpoint and $m = 6$.

Lemma 4.1. Let G be a path graph with $m + 1$ vertices. Let $t \geq m$ and let S_C be the leftmost vertex of G.
An optimal starting position is $S_D = v_j$, with the number of moves of the detective, d, such that:

- for $2 \leq m \leq t < 2m - 2$: $j = \lceil \frac{t}{2} \rceil + 1$ and $d = \lceil \frac{t}{2} \rceil + 1$,
- for $t \geq 2m - 2$: $j = m$ and $d = m$.

In order to prove that v_j is indeed an optimal starting position, we need to show:

(a) for this S_D there is no choice of a walk for C, w, that would make D move more, i.e. that
$$d = \max_{w \in W} n(w, S_D, R),$$

(b) for any other starting position $\tilde{S}_D$ there is a walk and F that make D move at least equally much, i.e. $\tilde{d} \geq d$, where $\tilde{d}$ corresponds to the number of moves required for D to reach F when starting in $\tilde{S}_D$ and d for S_D.

We will prove the two cases separately.

Proof. $2 \leq m \leq t < 2m - 2$
Let $j = \lceil \frac{t}{2} \rceil + 1$.

In order to use up all his moves, C can move to v_k with $k = \lceil \frac{t}{2} \rceil$ and go back to v_0 if t even or to v_1 if t odd.

We have that $S_D = v_j = v_{k+1}$ which is not visited, so D needs to go only towards F. Therefore, in case t is odd, C could go to v_0 via any walk instead of going to v_k and then to v_1, as this would increase the number of moves needed for D. Thus

$$d = j = \left\lceil \frac{t}{2} \right\rceil + 1.$$

(a) Let $\tilde{F} = v_f$ for some $f = 0, 1, \ldots, m$.
Let us define $W_{\tilde{F}}$ as the set of all walks of the criminal with $\tilde{F}$ as the final destination, then we need to show that

$$\max_{w_{\tilde{F}} \in W_{\tilde{F}}} n(w_{\tilde{F}}, S_D, R) = d.$$

Case distinction:

- $f \geq j$
C has t moves and uses j of them to get to v_j, so has at most $m-j$ moves to get to v_f from v_j, i.e. $f - j \leq m - j$.
Since we know that D starts in a visited vertex, they move only to the right, i.e. towards v_f and thus the number of moves is:

$$\tilde{d} = d(v_j, v_f) = f - j \leq m - j \leq t - j = t - \left(\left\lceil \frac{t}{2} \right\rceil + 1 \right) = \left\lfloor \frac{t}{2} \right\rfloor - 1 \leq d.$$

- $f < j$
We have that $d(v_f, v_j) = j - f$ and C can reach v_k with

$$k = \min\{f + \left\lfloor \frac{t-f}{2} \right\rfloor, m - 1\},$$

where they can turn around and go to v_f.
v_k is visited by C and v_{k+1} is not, so D needs to go to v_{k+1} and then turn back, so the number of moves is:

$$\begin{aligned}
\tilde{d} &= 2(d(v_k, v_j) + 1) + d(v_f, v_j) = 2\,(k + 1 - j) + (j - f) \\
&\leq 2\left(f + \left\lfloor \frac{t-f}{2} \right\rfloor + 1 - j \right) + j - f = 2\left\lfloor \frac{t-f}{2} \right\rfloor + f + 2 - j \\
&\leq t - f + f + 2 - \left(\left\lceil \frac{t}{2} \right\rceil + 1 \right) = t - \left\lceil \frac{t}{2} \right\rceil + 1 \\
&\leq \left\lfloor \frac{t}{2} \right\rfloor + 1 \leq d.
\end{aligned}$$

Since in both cases, we get that $\tilde{d} \leq d$, it holds that

$$\max_{w_{\tilde{F}} \in W_{\tilde{F}}} n(w_{\tilde{F}}, S_D, R) = d.$$

(b) Let $\tilde{S}_D = v_i$ for some $i = 0, 1, \ldots, t$. Case distinction:

- $t = 2$
  - $i = 0$ or $i = 2$
    Then C can choose $F = v_f$ with $f = t - i$ and thus
    $$\tilde{d} = d(v_i, v_f) = 2 = d.$$
  - $i = 1$
    C can go to v_i and back ($F = S_C$), so
    $$\tilde{d} = 2 + t - i = 2 + 1 = 3 > d.$$
- $t > 2$
  - $i \geq \left\lceil \frac{t}{2} \right\rceil + 1$ or $i \leq \left\lceil \frac{t}{2} \right\rceil - 2$
    Then $\exists\, v \in V : d(v_i, v) \geq \left\lceil \frac{t}{2} \right\rceil + 1 = d$, so C can choose $F = v$ and the number of moves of D is then $\tilde{d} = d(v_i, v) \geq d$.
  - $i = \left\lceil \frac{t}{2} \right\rceil - 1$ or $i = \left\lceil \frac{t}{2} \right\rceil$
    Then C can go to v_i and turn around, so
    $$\tilde{d} = 2 + t - i = \begin{cases} 2 + t - (\left\lceil \frac{t}{2} \right\rceil - 1) = \left\lfloor \frac{t}{2} \right\rfloor + 3 \geq \left\lceil \frac{t}{2} \right\rceil + 1 = d \\ 2 + t - \left\lceil \frac{t}{2} \right\rceil = \left\lfloor \frac{t}{2} \right\rfloor + 2 \geq \left\lceil \frac{t}{2} \right\rceil + 1 = d \end{cases}$$

  Therefore for all $\tilde{S}_D : \tilde{d} \geq d$.

□

Proof. $t \geq 2m - 2$
Let $j = m$ and $F = S_C$.
D moves only in one direction, so $d = d(S_D, S_C) = m$.

(a) All the other choices of F, say v_f, would be closer to S_D, because S_C and S_D are the endpoints of the graph. D would still move only in one direction, so $\tilde{d} = d(S_D, v_f) \leq d(S_D, F) = m = d$.

Hence, indeed, $\max_{w \in W} n(w, S_D, R) = d$.

(b) Let $\tilde{S}_D = v_i$ for some $i = 0, 1, \ldots, t$.

C can always go to v_{m-1} and back to S_C, as he has $t \geq 2m - 2$ moves. Therefore any choice of the starting position would be a visited vertex and D would have to first move to the right and then go to $F = S_C$. This gives $\tilde{d}$ moves with
$$\tilde{d} = 2 \cdot d(v_i, v_m) + i = d(v_i, v_m) + m > m = d.$$
Hence any other choice of the starting position results in a greater or equal number of moves for D.

□

4.2 S_C in the Centre

Another case of path graphs is the one with the criminal's starting position S_C in the very middle of the graph, as seen in Figure 4. This means that there are now two ways they can go. We will call the vertices and edges on two different sides of S_C *tails* and label the vertices belonging to them as v_i for the right tail and $\hat{v}_i$ for the left tail for $i = 0, 1, \ldots, m$ such that $d(S_C, v_i) = d(S_C, \hat{v}_i) = i$.

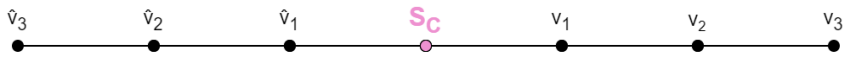

Figure 4: A path graph with S_C in the centre and $m = 3$.

In this case we assume that $t \geq m$, as if $t < m$ we could have taken a subgraph reaching up to $v_{\tilde{m}}$ such that $\tilde{m} = t$, analogously to the case with S_C in an endpoint.

Without loss of generality, we will focus on the starting positions on the right side of S_C, as the graphs are symmetric.

Lemma 4.2. Let G be a path graph with two tails of length $m \geq 1$ such that S_C is a vertex of G which is the centre. Let $t \geq m$.
Let $t \geq m$ and let S_C be the middle vertex of a path graph G. An optimal starting position is $S_D = v_j$, with the number of moves of the detective, d, such that:

- for $m \leq t < 3m - 2$: $j = \left\lceil \frac{t-m}{2} \right\rceil + 1$ and $d = \left\lceil \frac{t+m}{2} \right\rceil + 1$,
- for $t \geq 3m - 2$: $j = m$ and $d = 2m$.

Similarly to the previous case, in order to prove that v_j is an optimal starting position, we need to show (a) and (b) from 4.1. Again, we will prove the two cases separately.

Proof. $m \leq t < 3m - 2$
Let $S_D = v_j$ with $j = \left\lceil \frac{t-m}{2} \right\rceil + 1$.

In order to use up all his moves, C can move to v_k with $k = \left\lceil \frac{t-m}{2} \right\rceil$, turn around and go to $\hat{v}_m$ if $t - m$ even or to $\hat{v}_{m-1}$ if $t - m$ odd.

We choose $j = k + 1$, because if D starts in v_{k+1} which is not visited by C, they need to go only towards F. Therefore, in case $t - m$ is odd, to increase the number of moves needed for D, C can go to $\hat{v}_m$ via any walk instead of going to v_k and then to $\hat{v}_{m-1}$. Thus C chooses $F = \hat{v}_m$ and an arbitrary walk, and then

$$d = j + m = \left\lceil \frac{t-m}{2} \right\rceil + 1 + m = \left\lceil \frac{t+m}{2} \right\rceil + 1.$$

(a) We need to show that

$$\max_{w \in W} n(w, S_D, R) = d.$$

Any choice of the final destination, $\tilde{F} \neq \hat{v}_m$, and a walk for C that does not include v_j will result in a smaller number of moves for D, say $\tilde{d}$, as then C would end on the left side of v_j and therefore

$$\tilde{d} = d(v_j, \tilde{F}) < j + m = d.$$

Any choice of a walk for C leading to some $\tilde{F} = v_f$ with $f \geq j$ will also result in a smaller number of moves $\tilde{d}$, as then

$$\tilde{d} = d(v_j, \tilde{F}) = f - j \leq m - j \leq j + m = d.$$

Now, for any choice of a walk for C that includes v_j at least twice, i.e. $(v_0, v_1, \ldots, v_k, v_j, \ldots, v_{l-1}, v_l, v_{l-1}, \ldots, v_j, \tilde{F})$ and is of length t.
Every vertex that C visits on the right of v_k (i.e. $v_{k+1}, v_{k+2}, \ldots, v_m$) requires two moves, as C needs to go there and back. This means that $\tilde{F}$ would have to be two vertices to the right of F. Then, for every vertex visited by C, D needs to do two more moves to check the next vertex (get to know it is not visited) and go back, but also do two moves less in the end, as $\tilde{F}$ is forced to be placed two vertices to the right.
This means that for any choice of a walk of C which includes v_j, the number of moves for D is the same as d (or $d - 2$, if C visits v_m, as it is impossible to go further).

Hence, indeed, $\max_{w \in W} n(w, S_D, R) = d$.

(b) Let $\tilde{S}_D = v_i$ or $\hat{v}_i$ for some $i = 0, 1, \ldots, m$. Case distinction:

- $\tilde{S}_D = v_i$ or $\hat{v}_i$ with $i > j$
  C can then choose $\tilde{F}$ to be the further endpoint and the number of moves of D is $\tilde{d} = i + m > j + m = d$.
- $\tilde{S}_D = v_i$ or $\hat{v}_i$ with $i \leq j$
  In the case $\tilde{S}_D = v_i$, C can keep the same walk and $\tilde{F} = F$ as in the case for $S_D = v_j$ (or a reflection of it for $\hat{v}_i$), then D starts in a visited node and has to walk to $v_{k+1} = v_j$ (or $\hat{v}_{k+1} = \hat{v}_j$) and back. This results in the number of moves $\tilde{d} = d(v_i, v_j) + j + m = d(v_i, v_j) + d \geq d$.

  Hence any other choice of the starting position results in a greater or equal number of moves for D.

□

Proof. $t \geq 3m - 2$
Let $j = m$ and $F = \hat{v}_m$, then $d = 2m$, as there is only one way D can move.

(a) All the other choices for the final destination $\tilde{F}$ would be closer to v_m than $\hat{v}_m$ (by the definition of an endpoint).

Since the set of rules that D follows makes them move only in one direction when starting in an endpoint (as we then disregard whether a given vertex was visited or not), the number of moves will always be

$$\tilde{d} = d(v_m, \tilde{F}) = \begin{cases} m + m - f < 2m = d & \text{if } \tilde{F} = \hat{v}_f, \\ m - f < 2m = d & \text{if } \tilde{F} = v_f. \end{cases}$$

Hence, indeed, $\max_{w \in W} n(w, S_D, R) = d$.

(b) Let $\tilde{S}_D = v_i$ for some $i = 0, 1, \ldots, m-1$.
Choose $F = \hat{v}_m$ and the walk $(v_0, v_1, \ldots, v_{m-2}, v_{m-1}, v_{m-2}, \ldots, v_0, \hat{v}_1, \ldots, \hat{v}_m)$.
Then $\tilde{S}_D$ is always visited and therefore D has to make $\tilde{d}$ moves with

$$\tilde{d} = d(v_i, v_m) + m + m > 2m = d.$$

If we consider $\tilde{S}_D = \hat{v}_i$ for some $i = 0, 1, \ldots, m-1$, we can argue analogously and get the same result for $\tilde{d}$.

Hence any other choice of the starting position results in a greater or equal number of moves for D.

□

4.3 Other S_C

The last case of a path graph that we consider is with S_C anywhere else than the centre or the endpoints of the graph. This means that the graph is not symmetric and the two tails are not of the same length. We label the vertices similarly to the previous case: the ones on the right tail are labelled as v_i for $i = 0, 1, \ldots, m$ such that $d(S_C, v_i) = i$, and the ones on the left - as $\hat{v}_i$ for $i = 0, 1, \ldots, \hat{m}$ such that $d(S_C, \hat{v}_i) = i$.

Analogously to the previous cases, we assume that, without loss of generality, the left tail is shorter than the right one, i.e. $\hat{m} < m$, and $t \geq m$. An example of such a graph with $m = 4$ and $\hat{m} = 2$ can be seen in Figure 5. Since we are only looking into asymmetric graphs with two tails, we assume that $m \geq 2$ and $\hat{m} \geq 1$.

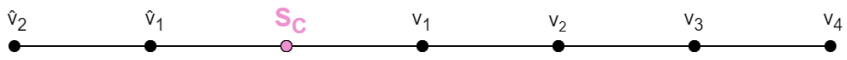

Figure 5: A path graph with $m = 4$ and $\hat{m} = 2$.

Lemma 4.3. Let G be a path graph with two tails of length $\hat{m}$ and m such that $m > \hat{m} \geq 1$ and S_C is the vertex of G between the tails. Let $t \geq m$.
An optimal starting position is $S_D = v_j$, with the number of moves of the detective, d, such that:

- for $m \leq t < 2m - 2 + \hat{m}$: $j = \left\lceil \frac{t-\hat{m}}{2} \right\rceil + 1$ and $d = \left\lceil \frac{t+\hat{m}}{2} \right\rceil + 1$,
- for $t \geq 2m - 2 + \hat{m}$: $j = m$ and $d = m + \hat{m}$.

We will prove the lemma analogously to the previous cases.

Proof. $m \leq t < 2m - 2 + \hat{m}$
Let $S_D = v_j$ with $j = \left\lceil \frac{t-\hat{m}}{2} \right\rceil + 1$.

Analogously to the case with S_C in the centre, in order to use up all his moves, C can move to v_k with $k = \left\lceil \frac{t-\hat{m}}{2} \right\rceil$, turn around and go to $\hat{v}_{\hat{m}}$ if $t - m$ even or to $\hat{v}_{\hat{m}-1}$ if $t - \hat{m}$ odd.

We choose $j = k + 1$, because if D starts in v_{k+1} which is not visited by C, they need to go only towards F. Therefore, in case $t - \hat{m}$ is odd, to increase the number of moves needed for D, C can go to $\hat{v}_{\hat{m}}$ via any walk instead of going to v_k and then to $\hat{v}_{\hat{m}-1}$. Thus C chooses $F = \hat{v}_m$ and an arbitrary walk, and then

$$d = j + \hat{m} = \left\lceil \frac{t - \hat{m}}{2} \right\rceil + 1 + \hat{m} = \left\lceil \frac{t + \hat{m}}{2} \right\rceil + 1.$$

(a) We need to show that

$$\max_{w \in W} n(w, S_D, R) = d.$$

Any choice of the final destination, $\tilde{F} \neq \hat{v}_{\hat{m}}$, and a walk for C that does not include v_j will result in a smaller number of moves for D, say $\tilde{d}$, as then C would end on the left side of v_j and therefore

$$\tilde{d} = d(v_j, \tilde{F}) < j + \hat{m} = d.$$

Any choice of a walk for C leading to some $\tilde{F} = v_f$ with $f \geq j$ will also result in a smaller or equal number of moves $\tilde{d}$, as then
$\tilde{d} = d(v_j, \tilde{F}) = f - j \leq m - j \leq m - \left\lceil \frac{m-\hat{m}}{2} \right\rceil - 1 = \left\lceil \frac{m+\hat{m}}{2} \right\rceil - 1 \leq \left\lceil \frac{t+\hat{m}}{2} \right\rceil + 1 = j + \hat{m} = d.$

Now, for any choice of a walk for C that includes v_j at least twice, i.e. $(v_0, v_1, \ldots, v_k, v_j, \ldots, v_{l-1}, v_l, v_{l-1}, \ldots, v_j, \tilde{F})$ and is of length t.
Every vertex that C visits on the right of v_k (i.e. $v_{k+1}, v_{k+2}, \ldots, v_m$) requires two moves, as C needs to go there and back. This means that $\tilde{F}$ would have to be two vertices to the right of F. Then, for every vertex visited by C, D needs to do two more moves to check the next vertex (get to know it is not visited) and go back, but also do two moves less in the end, as $\tilde{F}$ is forced to be placed two vertices to the right.
This means that for any choice of a walk of C which includes v_j, the number of moves for D is the same as d (or $d - 2$, if C visits v_m, as it is impossible to go further).

Hence, indeed, $\max_{w \in W} n(w, S_D, R) = d$.

(b) Let $\tilde{S}_D = v_i$ for some $i = 0, 1, \ldots, m$ or $\tilde{S}_D = \hat{v}_i$ for some $i = 0, 1, \ldots, \hat{m}$. Case distinction:

- $\tilde{S}_D = v_i$ or $\hat{v}_i$ with $i > j$
  C can then choose $\tilde{F}$ to be the further endpoint and the number of moves of D is $\tilde{d} = i + m > j + m = d$.
- $\tilde{S}_D = v_i$ or $\hat{v}_i$ with $i \leq j$
  In the case $\tilde{S}_D = v_i$, C can keep the same walk and $\tilde{F} = F$ as in the case for $S_D = v_j$ (or a reflection of it for $\hat{v}_i$), then D starts in a visited node and has to walk to $v_{k+1} = v_j$ (or $\hat{v}_{k+1} = \hat{v}_j$) and back. This results in the number of moves

$$\tilde{d} = \begin{cases} d(v_i, v_j) + j + \hat{m} = d(v_i, v_j) + d \geq d, \\ d(\hat{v}_i, \hat{v}_j) + j + m \geq d(\hat{v}_i, \hat{v}_j) + j + \hat{m} = d(\hat{v}_i, \hat{v}_j) + d \geq d. \end{cases}$$

  Hence any other choice of the starting position results in a greater or equal number of moves for D.

□

Proof. $t \geq 2m - 2 + \hat{m}$
Let $j = m$ and $F = \hat{v}_{\hat{m}}$, then $d = m + \hat{m}$, as there is only one way D can move.

(a) We need to show that

$$\max_{w \in V} n(w, S_D, R) = d.$$

All the other choices of the final destination $\tilde{F}$ would be closer to v_m than $\hat{v}_{\hat{m}}$ (by the definition of an endpoint).

Since the set of rules that D follows makes them move only in one direction when starting in an endpoint (as we then disregard whether a given vertex was visited or not), the walk that C can be arbitrary and the number of moves will always be

$$\tilde{d} = d(v_m, \tilde{F}) = \begin{cases} m + \hat{m} - f < m + \hat{m} = d & \text{if } \tilde{F} = \hat{v}_f, \\ m - f < m + \hat{m} = d & \text{if } \tilde{F} = v_f. \end{cases}$$

Hence, indeed, $\max_{w \in W} n(w, S_D, R) = d$.

(b) Let $\tilde{S}_D = v_i$ for some $i = 0, 1, \ldots, m - 1$.
Choose $\tilde{F} = \hat{v}_{\hat{m}}$ and the walk $(v_0, v_1, \ldots, v_{m-2}, v_{m-1}, v_{m-2}, \ldots, v_0, \hat{v}_1, \ldots, \hat{v}_{\hat{m}})$.
Then $\tilde{S}_D$ is always visited by C and therefore D has to make $\tilde{d}$ moves with

$$\tilde{d} = d(v_i, v_m) + m + \hat{m} > m + \hat{m} = d.$$

If we consider $\tilde{S}_D = \hat{v}_i$ for some $i = 0, 1, \ldots, \hat{m} - 1$, we can argue analogously and get

$$\tilde{d} = d(\hat{v}_i, \hat{v}_{\hat{m}}) + \hat{m} + m > m + \hat{m} = d.$$

Hence any other choice of the starting position results in a greater or equal number of moves for D.

□

4.4 General S_C

We can combine the results of this section to derive a formula for all cases of path graphs. We will not prove it, as it follows directly from the proofs above.

Lemma 4.4. Let G be a path graph with two tails of length $\hat{m} \geq 0$ and $m \geq 1$ such that $m \geq \hat{m}$ and S_C is a vertex of G between the tails. Let $t \geq m$.
An optimal starting position is $S_D = v_j$, with the number of moves of the detective, d, such that:

- for $2 \leq m \leq t < 2m - 2 + \hat{m}$: $j = \left\lceil \frac{t-\hat{m}}{2} \right\rceil + 1$ and $d = \left\lceil \frac{t+\hat{m}}{2} \right\rceil + 1$,
- for $t \geq 2m - 2 + \hat{m}$: $j = m$ and $d = m + \hat{m}$.

Now we can prove that the chosen set of rules, R, is indeed optimal. We will show that any other set of rules for a given graph G and a starting position of the detective $\tilde{S}_D$ is equally good or worse by showing that any deviation from the set of rules R would result in at least equally long walk for D.

Proof. Let G be a path graph with two tails of length m and $\hat{m}$ such that $m \geq \hat{m}$ and $m \geq 1$. Let $t \geq m$. Let d be defined as in Lemma 4.4.
Note that in the case $2 \leq m \leq t < 2m - 2 + \hat{m}$ we have that $d = \left\lceil \frac{t+\hat{m}}{2} \right\rceil + 1 < \left\lceil \frac{2m-2+\hat{m}+\hat{m}}{2} \right\rceil + 1 = m + \hat{m}$ and in the case $t \geq 2m - 2 + \hat{m}$: $d = m + \hat{m}$, so $\forall t \geq m : d \leq m + \hat{m}$.
Let us check all the rules in R.

- The rule in lines 1 and 2 is necessary for all the algorithms, as it is our objective to stop in F.
- The rule in lines 4 and 5 is also necessary, as if the endpoint is not F, then, in order to find F, D needs to move, and the only possible way to move is towards the other endpoint.
- The alternative to the rule in lines 6, 7 and 8, is for D to move to the shorter side of the graph. In principle, this move can be arbitrary, as the length of the walk is determined more by other rules and choosing S_C to be S_D does not give any information, so it would not be an optimal starting position (besides the case when $\hat{m} = 0$ and $m = 1$, then both possibilities for S_D give the same number of moves).
  However, there needs to be a rule for a situation when $S_D = S_C$, so we

choose this one, as it is more suitable in the case when S_C is an endpoint of G.

- The alternative to the rule in lines 6, 9 and 10 is for D to move in the opposite direction after arriving at S_C, which means going back to the previously checked vertex. This would make a loop, which makes the walk longer than it would have been otherwise (if keeping the rest of the rules the same as in R). The choice of other rules would also result in this loop making the walk longer.
- The alternative to the rule in lines 12 and 13 is for D to move away from S_C if the current vertex is not visited by C.
  Since the current vertex is not visited, any vertex further from S_C is not visited either, as G is a path graph. Thus, if D moves away from S_C, they are not going to check any visited vertex, so also not F. In order to find F they would have to turn around at some point, which makes a loop in their walk. This results in a walk of length at least $d+2$, as if D followed the rule in R, they would have turned around at least one vertex earlier.
- For the rule in lines 14 and 15, the alternative is for D to move away from S_C if D started moving towards S_C. C can then choose F to be on the other side of S_C. In that case, this alternative rule causes D to make a loop of length at least 2, if there is another rule that would cancel this one at some point, or to never arrive at F, as D would make infinitely many loops (caused by the rule in lines 4 and 5 or other possible additional rules).
- The alternative to the last rule is analogous to the previous one - if D is moving away from S_C, then D moves towards S_C. In this case, C can choose F to be on the same side of S_C as the current vertex. Then, analogously to the previous rule, D either has to make an additional loop or can never reach F.

Since any deviation from the rules in R results in an equally long or longer walk for D, the chosen set of rules is optimal. □

5 Stars

Another type of graphs that we consider is stars.

Definition 5.1 (Star)**.** A star $\mathcal{S}_k$ is a tree on $k+1$ vertices with one vertex having vertex degree k (called the internal node if $k \geq 2$) and the other k vertices having vertex degree 1 (called leaves).

We label the vertices in a star differently than in a path graph. Say the number of leaves is k, then the internal node is labelled as v_0 and all the leaves are v_i for $i = 1, 2, \ldots, k$, where we start on top of the graph and count clockwise,

as shown in Figure 6. Similarly to path graphs, we can assume that labelling without loss of generality, as stars are symmetric.

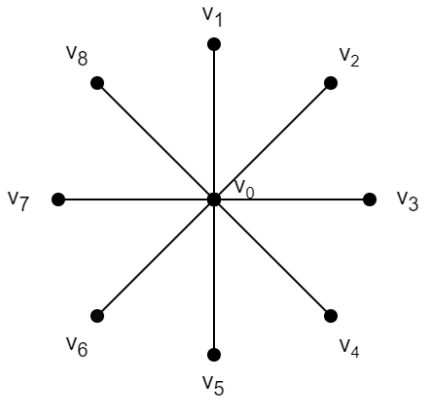

Figure 6: A star with 8 leaves.

If $S_C = v_0$, we can have $t \geq 1$, as it takes exactly one move for C to reach any leaf. We also assume that if $S_C = v_i$ for any $i \neq 0$, then $t \geq 2$, as if $t = 1$, then F could only be in S_C or in v_0, which are trivial cases.

The set of rules that we consider for stars is as follows.

Algorithm 2 The set of rules for stars.

```
if current vertex is F then
    D stops
else
    if current vertex is v_0 then
        D moves to the next leaf (counting clockwise)
    else
        D moves to v_0
    end if
end if
```

Checking a vertex does not give any information, unless it is F, in which situation D is done, because C could go anywhere from v_0 in one move. Therefore any other set of rules would result in the same or worse outcome, because D and C can move only along the existing edges or pass, i.e. they need to go to v_0 before moving onto any other vertex. Thus we can say that this set of rules is optimal.

Lemma 5.1. Let G be a star graph with k leaves. Let $t \geq 2$ and S_C be a vertex of G. An optimal starting position is $S_D = v_1$, with the number of moves of the detective, $d = 2k - 2$.

In order to prove this lemma, we need to show (a) and (b) from 4.1, as in the previous cases.

Proof. $t \geq 2$
Let $S_D = v_1$.

(a) All the leaves have only one neighbour - the internal node - so C and D always need to go there before going to another leaf. Therefore we can easily see that t does not influence the worst-case scenarios for any choice of S_D, as checking any vertex does not give any information (unless it is the final destination). Therefore we can assume that C takes the path (v_0, F) of length 1.

If $F = v_k$, then D needs to check all the vertices, starting in v_1 and ending in v_k, so the number of moves is

$$d = 1 + 2(k-2) + 1 = 2k - 2.$$

If $F = v_f$ with $0 < f < k$, then D needs to check all the vertices between v_1 and v_f (i.e. v_0 and v_i for all $i = 2, 3, \ldots, f-1$) and end in v_f, so the number of moves is

$$\tilde{d} = 1 + 2(f-1-1) + 1 = 2f - 2 < 2k - 2 = d.$$

If $F = v_0$, then D needs to make only one move and $1 < d$.

Hence, indeed, $\max_{w \in W} n(w, S_D, R) = d$.

(b) Let us consider two cases.

- $\tilde{S}_D = v_i$ for some $i = 2, 3, \ldots, k$
  As mentioned earlier, star graphs are symmetric, so the labelling does not matter and thus in a worst-case scenario for any v_i with $i \neq 0$, we get the number of moves $\tilde{d} = 2k - 2 = d$.

- $\tilde{S}_D = v_0$
  Choose $\tilde{F} = v_m$. Then D needs to check all the vertices ($k-1$ of them) before getting to F, so the number of moves is $\tilde{d} = 2(k-1)+1 = 2k-1 > 2k-2 = d$.

  Hence any other choice of the starting position results in a greater or equal number of moves for D.

□

6 Trees

As we have found optimal strategies for path graphs and stars, the next step is to investigate *starlike trees*. Let us first define starlike trees.

Definition 6.1 (Starlike tree)**.** A starlike tree is a tree which has exactly one vertex of vertex degree greater than 2 (called the root).

Without loss of generality and for ease of notation we will present the trees with the shortest branch on top of the graph, while the other branches increase in length in a clockwise direction. Note that some branches can be of equal length.

Similarly to stars, we will label the branches starting on top and going clockwise and, similarly to path graphs, we will label vertices with respect to their distance from the centre (labelled as v_0) which is always going to be the starting position of the criminal, S_C. Thus, if we have k branches, their lengths are $1 \leq m_1 \leq m_2 \leq \cdots \leq m_{k-1} \leq m_k$ and the vertices are labelled v_{i_k}, where i_k is the distance from $v_0 = S_C$ on the k^{th} branch. An example of such a graph can be seen in Figure 7.

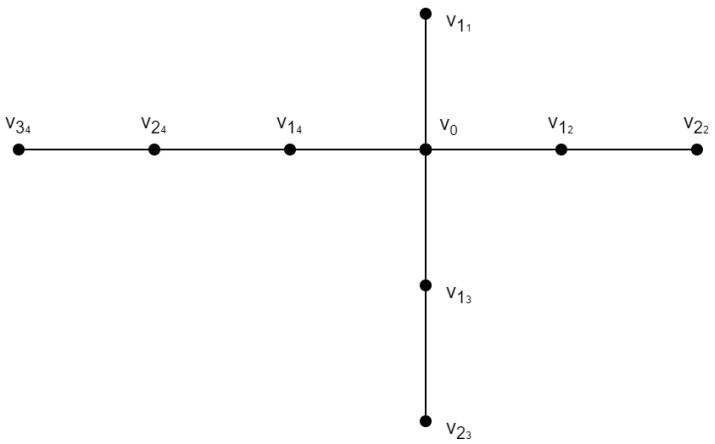

Figure 7: A starlike tree with 4 branches.

Analogously to path graphs, we assume that the number of moves of the criminal is $t \geq m_k$, as if it was smaller than m_k we could have taken a subgraph ending in the vertex which is at distance t from S_C (the centre). We will consider two cases of starlike trees, depending on the value of t:

$$m_k \leq t < 2 \sum_{l=1, l \neq k-1}^{k} (m_l - 1) + m_{k-1}$$

and

$$t \geq 2 \sum_{l=1, l \neq k-1}^{k} (m_l - 1) + m_{k-1}.$$

This case distinction is made analogously to the ones in path graphs, as in the latter case C is able to go to one-to-last vertex on each branch and go back, and end in the furthest vertex on the shortest branch. Note that if $m_l = 1$ for some $l \geq 2$, then C does not visit that branch at all, but D does still have to check it, analogously to stars.

We will propose strategies, so a combination of a starting position S_D and a set of rules R, for both cases and prove that a given strategy is optimal. We do that rather than considering S_D and R separately, like in previous cases, as now we can make use of those results. We will start with a strategy for the greater t, as we will use it for smaller t as well.

Lemma 6.1. Let G be a starlike tree with k branches of lengths m_l for $l = 1, 2, \ldots, k$, such that $1 \leq m_1 \leq m_2 \leq \cdots \leq m_{k-1} \leq m_k$ and S_C is the root of G. Let $t \geq 2 \sum_{l=1, l \neq k-1}^{k} (m_l - 1) + m_{k-1}$.
An optimal strategy consists of the set of rules R and the starting position S_D such that:
$S_D = v_{m_k}$, with the number of moves of the detective

$$d = m_k + m_{k-1} + 2 \sum_{l=1}^{k-2} m_l$$

and R is described with the following algorithm.

Proof. The rules in lines 1 to 8 and 12 to 17 are analogous to the ones in the algorithm for path graphs and those were proven to be necessary in an optimal set of rules.

The rule in lines 6, 9 and 10 is connected to the choice of S_D.
Choose $S_D = v_{m_k}$.
Since t is big enough for C to visit all vertices but the last one on all the branches and reach the furthest vertex on the $(k+1)^{th}$ branch, he can do that and end in $v_{m_{k-1}}$.
Let us choose w - the walk of C visiting all those vertices and $F = v_{m_{k-1}}$.
Then if D starts in v_{m_k} and checks all the branches clockwise, they end in $v_{m_{k-1}}$. That way they go to the endpoint and back on the shortest branches and only to the endpoints of the longest ones.
Essentially the order of checking the branches between the last one and the first one does not matter, as long as the second longest one is the last one to be checked (for the chosen S_D). That saves going back and forth on the longest branches, which would make longer loops in the detective's walk. An equally good rule would be to start in the endpoint of the second longest branch and go counterclockwise, as then D takes the same walk but in reverse.

Algorithm 3 The set of rules for starlike trees.

```
1: if current vertex is F then
2:     D stops
3: else
4:     if current vertex is an endpoint then
5:         D moves towards S_C
6:     else if current vertex is S_C then
7:         if current vertex is S_D then
8:             D moves to the longest branch
9:         else
10:            D moves to the next branch (counting clockwise)
11:        end if
12:    else if current vertex is not visited by C then
13:        D moves towards S_C
14:    else if D started moving towards S_C then
15:        D moves towards S_C
16:    else
17:        D moves away from S_C
18:    end if
19: end if
```

If they skipped a branch, however, C could choose the endpoint of that branch as the final destination, so D needs to check all the branches.

As mentioned earlier, since D starts in the longest one and ends in the second longest one, they traverse those only once. As they need to check all the other branches, they need to traverse them twice (go to the endpoint and back). This gives

$$d = m_k + m_{k-1} + 2\sum_{l=1}^{k-2} m_l.$$

Now we need to show (a) and (b) from 4.1, as in the previous cases.

(a) To show: $\max_{w \in W} n(w, S_D, R) = d$. Any choice of the final destination $\tilde{F} \neq v_{m_{k-1}}$ and any walk leading to it would make D move less, as D checks the branches clockwise and in that sense $v_{m_{k-1}}$ is the furthest vertex from $v_{m_k} = S_D$.

Hence, indeed, $\max_{w \in W} n(w, S_D, R) = d$.

(b) Let us consider two cases:

- $\tilde{S}_D = v_{i_k}$, where $0 \leq i_k < m_k$, so $0 \leq d(v_{i_k}, v_0) < d(S_D, v_0)$
  C can choose the same walk, w, and $\tilde{F} = F$, then the number of moves for D is:
  $$\tilde{d} = d(v_{i_k}, S_D) + d > d.$$

- $\tilde{S}_D = v_{i_h}$ for some $i_h = 1, 2, \ldots, h$ for some $h = 1, 2, \ldots, k-1$
  C can choose a walk that visits all branches in the same way as earlier* and ends in $\tilde{F} = v_{m_{h-1}}$ for $h \geq 2$ or in $\tilde{F} = v_{m_k}$ for $h = 1$.
  * Note that if $t = 2\sum_{l=1,l\neq k-1}^{k}(m_l - 1) + m_{k-1}$, C can visit all the vertices up to the one-to-last one on all the branches but the $(k-1)^{th}$ branch, so on that one, he reaches the vertex $v_{a_{k-1}}$ with $a_{k-1} = \lfloor \frac{m_{k-1}}{2} \rfloor$ and then turns around. When t is greater, he can move further on the $(k-1)^{th}$ branch, making D move more as well.
  Therefore the number of moves of D is:

$$\begin{aligned}
\tilde{d} &\geq \begin{cases} d(v_{i_{k-1}}, v_{m_{k-1}}) + m_{k-1} + m_{k-2} + 2\sum_{l\neq k-1,h-2} m_l & \text{if } h = k-1, \\ d(v_{i_1}, v_{m_1}) + m_1 + m_k + 2(\lfloor \frac{m_{k-1}}{2} \rfloor + \sum_{l=2}^{k-2} m_l) & \text{if } h = 1, \\ d(v_{i_h}, v_{m_h}) + m_h + m_{h-1} + 2(\lfloor \frac{m_{k-1}}{2} \rfloor + \sum_{l\neq h,h-1,k-1} m_l) & \text{else,} \end{cases} \\
&\geq d(v_{i_1}, v_{m_1}) + m_1 + m_k + m_{k-1} - 1 + 2 \cdot \sum_{l=2}^{k-2} m_l \\
&\geq m_k + m_{k-1} + 2\sum_{l=1}^{k-2} m_l \\
&= d.
\end{aligned}$$

  Hence $\tilde{d} \geq d$ for all other possibilities for a starting position.

Thus the chosen S_D and R form an optimal strategy. □

Lemma 6.2. Let G be a starlike tree with k branches of lengths m_l for $l = 1, 2, \ldots, k$, such that $1 \leq m_1 \leq m_2 \leq \cdots \leq m_{k-1} \leq m_k$ and S_C is the root of G. Let $m_k \leq t < 2\sum_{l=1,l\neq k-1}^{k}(m_l - 1) + m_{k-1}$.
An optimal strategy consists of the set of rules R, as introduced above, and the starting position S_D such that:
$S_D = v_{j_k}$ is a vertex on the longest branch with

- for $m_k \leq t < m_{k-1} + 2\sum_{l=1}^{k-2}(m_l - 1)$: $j_k = 1$ and the number of moves of the detective:
$$d = 1 + t + 2(k-2),$$
- for $m_{k-1} + 2\sum_{l=1}^{k-2}(m_l - 1) \leq t < 2\sum_{l=1,l\neq k-1}^{k}(m_l - 1) + m_{k-1}$:
$$j_k = \left\lceil \frac{t - (2\sum_{l=1}^{k-2}(m_l - 1) + m_{k-1})}{2} \right\rceil + 1$$
  and the number of moves of the detective:
$$d = j_k + m_{k-1} + 2\sum_{l=1}^{k-2} m_l.$$

Proof. Analogously The rules in R follow from the previous algorithms, and the proof for the rule in lines 6, 9 and 10 is analogous to the case above.
If we choose $S_D = v_{j_k}$ as defined above, C could visit the other branches and end in the endpoint of the second longest one. This means that D starts in a not visited vertex (analogously to path graphs), so they need to walk directly to the internal vertex, traverse the shorter branches up to the first not visited vertex twice and go to the endpoint of the second longest branch.

Moving as described above gives us the number of moves of the detective:

- for $m_k \leq t < m_{k-1} + 2\sum_{l=1}^{k-2}(m_l - 1)$:

$$d = 1 + t + 2(k-2),$$

- for $m_{k-1} + 2\sum_{l=1}^{k-2}(m_l - 1) \leq t < 2\sum_{l=1, l\neq k-1}^{k}(m_l - 1) + m_{k-1}$:

$$d = j_k + m_{k-1} + 2\sum_{l=1}^{k-2}(m_l - 1) + 2(k-2) = j_k + m_{k-1} + 2\sum_{l=1}^{k-2} m_l.$$

That is because D needs to reach the internal node, do all the moves of C and check one more vertex on each branch between the first and the last one, which gives $2(k-2)$ additional moves. Now we need to show (a) and (b) from 4.1, as in the previous cases.

(a) To show: $\max_{w\in W} n(w, S_D, R) = d$.
Analogously to path graphs, C can visit any further vertex, but uses two moves for that, as he would have to go there and back, so he has to end two vertices closer. This gives the same number of moves for the detective, d. If those further vertices are endpoints, say e of them, then the number of moves of D is $d - 2e$, because D would have checked the endpoints anyways and C uses up two moves for each, so removes two vertices in the end of his walk. Note that this formula is true only if $F \neq v_0$, because then D would finish in v_0 the first time they visit it, and then the number of moves of D would be even smaller.
C could also visit different branches (without reaching the endpoints), but the number of his moves, t, is still the same, so the number of moves for D is the same too.

Hence, indeed, $\max_{w\in W} n(w, S_D, R) = d$.

(b) Let us consider two cases:

- $\tilde{S}_D = v_{ik}$ for $i \neq j$
In this case, C can take the same walk as earlier and the number of moves of D is then

$$\tilde{d} = \begin{cases} d + i_h - j_k > d & \text{if } i_h > j_k, \\ d + 2(j_k - i_h) > d & \text{if } i_h < j_k. \end{cases}$$

- $\tilde{S}_D = v_{i_h}$ for some $i_h = 1, 2, \ldots, h$ for some $h = 1, 2, \ldots, k-1$
  Analogously to the case above, C can choose a walk that visits other branches in the same way as earlier and ends in $\tilde{F} = v_{m_{h-1}}$ for $h \geq 2$ or in $\tilde{F} = v_{m_k}$ for $h = 1$. The number of moves is then still the same (for smaller t) or bigger (for greater t), as C can go further on the longest branches (for the greater t) and checking any vertex on any branch requires one (if it is v_{gh} for some $g < h$ or any vertex on the last branch) or two moves (for any other vertex). The number of moves of D is then

$$\tilde{d} \geq \begin{cases} d + i_h - j_k > d & \text{if } i_h > j_k, \\ d & \text{if } i_h = j_k, \\ d + 2(j_k - i_h) > d & \text{if } i_h < j_k. \end{cases}$$

  Hence $\tilde{d} \geq d$ for all other possibilities for a starting position.

Thus the chosen S_D and R form an optimal strategy. □

7 Conclusion and Recommendations

We have found optimal strategies, that consist of a set of rules and a starting position of the detective, for path graphs, stars and starlike trees. We should note that path graphs and stars are special cases of starlike trees, but this division was needed to build up to the more general case.

As mentioned earlier, the game *Scotland Yard* was proven to be NP-complete, and therefore it is very difficult to find an optimal strategy for it. However, there are multiple possibilities to get closer to solving it.

7.1 Further Research

Further steps could include finding optimal strategies for a general case of trees and later for cyclic planar graphs as well. Then the specific cases of the boards for both games could be solved, as they are both cyclic graphs. This can be done while keeping the simplified rules, but another possibility is to consider more or different rules.

To make it more similar to the games, one might look for strategies for a game with turns, where the moves of the criminal and the detective are alternating. The objective would then be different, as we would not look for the hideout of the criminal, but rather for the criminal himself. One could also include more detectives, as it is in the game in order to cover the given graph to a greater extent and increase the chances of the detectives winning.

As the two games are originally built differently, another further step could be to look for strategies for graphs made like the one for *Scotland Yard*, where we have multiple means of transportation or like the one for *Letters from Whitechapel*, where we have different sets of vertices for the detectives and the criminal.

Another extension could be considering a probabilistic approach for graph searching methods for cops and robbers games as described in the book *Graph Searching Games and Probabilistic Methods* [1]. This, however, focuses on games with perfect information, so the first step would be to use determinization to handle the imperfect information, as in [8], where Monte Carlo Tree Search is applied to the *Scotland Yard* game.

8 Acknowledgements

I would like to thank Bart Smeulders for his supervision and many encouraging words. I would also like to thank my friends who immediately bought me both games when they heard about this project.

References

[1] Anthony Bonato and Pralat Pawel. *Graph Searching Games and Probabilistic Methods*. Chapman and Hall/CRC, Nov. 2017, pp. 29–30. ISBN: 9781315212135. DOI: 10.1201/9781315212135.

[2] Eduardo J Subelman. "A Hide-Search Game". In: *Journal of Applied Probability* 18.3 (1981), pp. 628–640. ISSN: 00219002. DOI: 10.2307/3213317. URL: http://www.jstor.org/stable/3213317.

[3] Jonathan Liu. *Getting Away With Murder: Letters From Whitechapel*. May 2011. URL: https://www.wired.com/2011/05/getting-away-with-murder-letters-from-whitechapel/.

[4] *Board Game Geek - Letters from Whitechapel board*. Apr. 2011. URL: https://boardgamegeek.com/image/971115/letters-whitechapel.

[5] Merlijn Sevenster. "The Complexity of Scotland Yard". In: *Interactive Logic (J. van Benthem, B. Löwe, and D. Gabbay, eds.)* (2006), pp. 209–246.

[6] Tirtharaj Dash et al. "Adversarial neural networks for playing hide-and-search board game Scotland Yard". In: *Neural Computing and Applications* 32.8 (Apr. 2020), pp. 3149–3164. ISSN: 14333058. DOI: 10.1007/s00521-018-3701-0.

[7] Martin Schmid et al. "Player of Games". In: (Dec. 2021). URL: http://arxiv.org/abs/2112.03178.

[8] P Nijssen and M H M Winands. "Monte Carlo Tree Search for the Hide-and-Seek Game Scotland Yard". In: *IEEE Transactions on Computational Intelligence and AI in Games* 4.4 (2012), pp. 282–294. ISSN: 1943-0698. DOI: 10.1109/TCIAIG.2012.2210424.