Strategies and Modelling of Scotland Yard Game

Freddie Wallwork and Ed Churchill

The University of Warwick

January 10, 2022

1 Introduction

We concentrate on a popular board game called Scotland Yard involving 2-5 'Detective' players searching for a player 'MisterX'. The game is played on a graph with vertices representing locations in London and 3 edge types representing taxi, bus and underground travel. The players take it in turns to move to a vertex adjacent to their current location.

- The aim of the Detectives is to 'capture' MisterX by landing on the same vertex as him or not allowing him to move.
- The aim of MisterX is to avoid 'capture' by the Detectives for 24 turns.

The winner is the side that achieves their aim. The winning side exists and is unique.

The game begins with all players choosing a vertex, at random, from a set of possible starting vertices. The Detectives locations are at all times common knowledge however the location of MisterX is only revealed to the Detectives after his 3rd, 8th, 13th, 18th and 24th turns. Each Detective receives 4 underground tickets, 8 bus tickets and 11 taxi tickets.

On each turn MisterX moves to an unoccupied adjacent vertex, writes the vertex number down and covers it with the ticket used to get there (taxi, bus, underground). He therefore records his move but only revealing the mode of transport used to Detectives. If this is not possible due to the positions of the Detectives, the game is over and the Detectives win. The Detectives then move in the same way on either a taxi, bus or underground edge, spending a respective ticket in the process.

Furthermore, MisterX receives 5 blank tickets, which allow him to hide his mode of transport on a turn. He also receives 2 double-move tokens, which allow him to take 2 turns before the Detectives get a chance to move.

There are always at least 4 searchers in the game. When there are less than 4 Detectives, then 'Policemen' are added as searchers, who do not have the ticket restriction that Detectives have. These players are controlled jointly by the Detective players.

2 Model of the game

We model a simplified version of the game in order to improve the performance of the player algorithms. We will remove rules involving travel tickets for the Detectives, essentially making all Detectives policemen (however we will continue to refer to all searchers as Detectives throughout). We will further simplify the game by removing all double-move tokens and blank tokens for MisterX. To balance these disadvantages to the player MisterX, we will reduce the number of Detectives to 3.

We construct an undirected graph $G = (V, E = E_t \cup E_b \cup E_u)$, where:

- $V = \{1, 2, \ldots, 199\}$ is the set of vertices representing the locations on the board
- E_t is the set of edges representing taxi travel
- E_b is the set of edges representing bus travel
- E_u is the set of edges representing underground travel

Definition 2.1. For $k \in \{0, 1, 2, \ldots, 24\}$ and $j \in \{1, 2, 3\}$:

- LM_k is the location (vertex) of MisterX at the end of turn k.
- LD_k^j is the location (vertex) of Detective j at the end of turn k.

Definition 2.2. For $k \in \{1, 2, \ldots, 24\}$ and $j \in \{1, 2, 3\}$:

- M_k is the set of possible vertices that MisterX can move to on turn k.
- D_k^j is the set of possible vertices that Detective j can move to on turn k.
- $XTransMode_k$ is the transport mode used by MisterX on the turn k. (taking values t, b or u for taxi, bus or underground respectively)

The game runs as follows:

Algorithm 1: SCOTLAND YARD GAME

Input: LM_0, LD_0^1, LD_0^2, LD_0^3
Output: Winner

```
for k in {1,2,...,24} do
    LM_k = MisterXMove
    Record XTransMode_k
    (LD_k^1, LD_k^2, LD_k^3) = DetectiveMove
    if LM_k ∈ {LD_k^1, LD_k^2, LD_k^3} then
        STOP: Detectives win
STOP: MisterX Wins
```

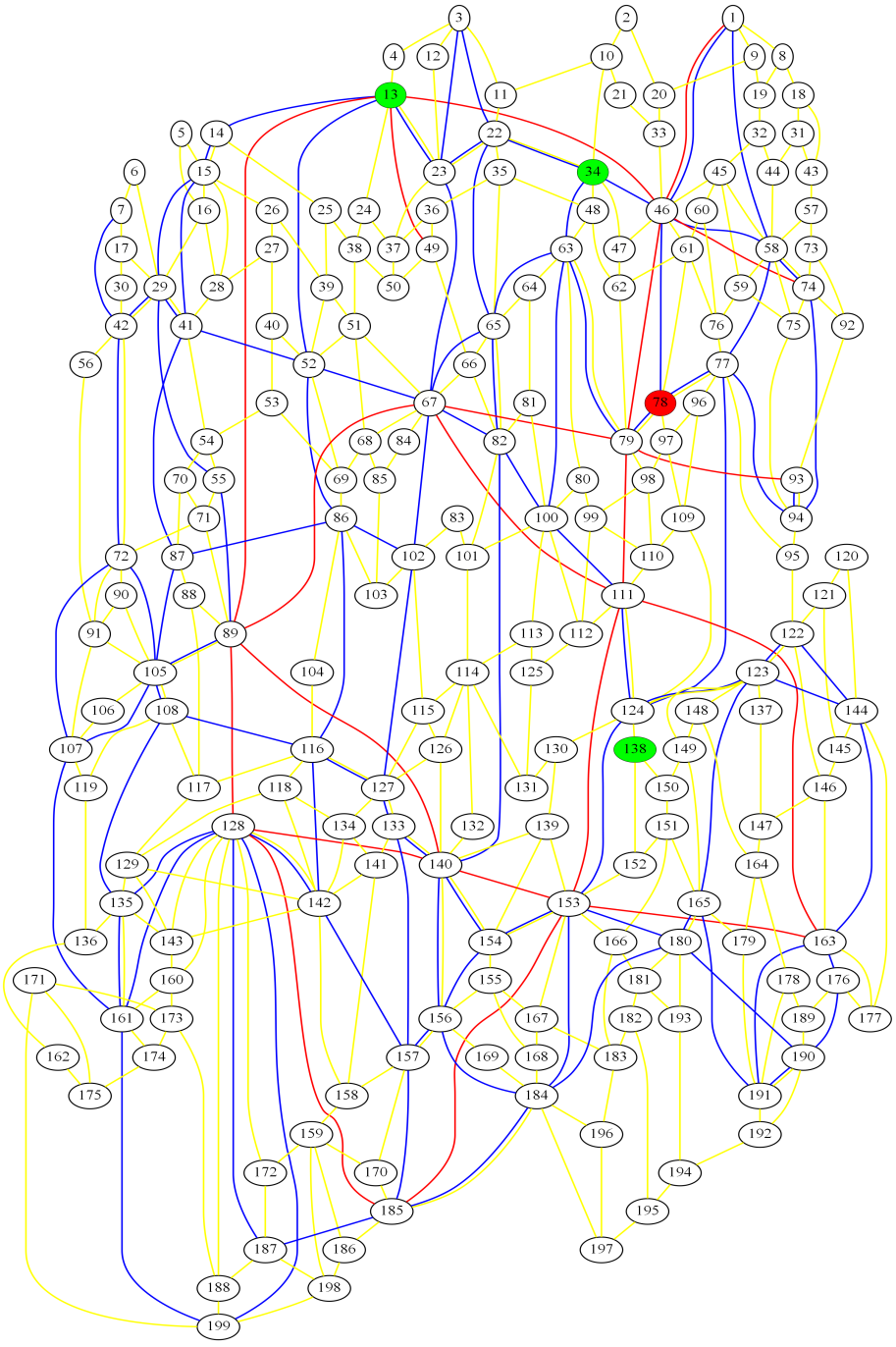


Figure 1: The graph $G(V, E_t \cup E_b \cup E_u)$ where edges of E_t, E_b and E_u are represented in yellow, blue and red respectively. MisterX's position is highlighted in red and those of the Detectives in green.

Modelling in Python

In order to model and visualise the game, we used Python together with a package called Graphviz. There are 9 files in total and what each one does is described below.

- *'taxi_edges.txt'*, *'bus_edges.txt'* and *'tube_edges.txt'* are simple text files. They contain the data for E_t, E_b and E_u respectively.
- *'generate_graphs.py'* is a Python file containing functions that ultimately output an image of the graph, including the locations of MisterX and the Detectives. The vertex where MisterX is located is highlighted **green**. The vertices where the Detectives are located are highlighted **red**. If a Detective is on the same vertex as MisterX, then that vertex is highlighted **orange**.
- *'players.py'* is a Python file containing two Python classes, one to model MisterX and one to model a Detective. Each class contains a 'constructor' function assigning the respective character an initial location. Each class also contains a function to calculate the possible vertices the character can move to, given their current location.
- *'grahical_distance_calculator.py'* is a Python file containing functions which calculate graphical distance, as defined in Section 4 of this paper.
- *'game_one.py'*, *'game_two.py'* and *'game_three.py'* are Python files containing functions to run the game with the strategies outlined in Sections 3, 4 and 5 respectively.

3 Initial Strategies

As an initial strategy, we will select the move of every player by choosing a vertex uniformly at random from their possible moves (i.e. from M_k for MisterX and D_k^j for Detective j). For a finite set S, we denote by $Random(S)$ an element chosen uniformly at random from S.

Algorithm 2: MisterXRandom STRATEGY

Input: k=TURN, LM_{k-1}, LD_{k-1}^1, LD_{k-1}^2, LD_{k-1}^3
Output: LM_k

```
Calculate M_k
if M_k = ∅ then
    Stop Detectives win
else
    LM_k = Random(M_k)
    Return LM_k
```

Algorithm 3: DetectiveRandom STRATEGY

Input: k=TURN, LD^1_{k-1}, LD^2_{k-1}, LD^3_{k-1}
Output: LD^1_k, LD^2_k, LD^3_k

for j *in* {*1,2,3*} **do**
    Calculate D^j_k
    if $D^j_k = \emptyset$ **then**
        $LD^j_k = LD^j_{k-1}$
    else
        $LD^j_k = Random(D^j_k)$
Return LD^1_k, LD^2_k, LD^3_k

Results

We ran the game 100 times with the MisterXRandom strategy for MisterX and DetectiveRandom strategy for the Detectives. The Detectives won 26 times and MisterX won 74. We expected MisterX to win the majority of the time, however were expecting a larger majority, since the board is so large and the Detectives don't consider MisterX's location when moving.

possibly add paragraph to explain why detectives occasionally win

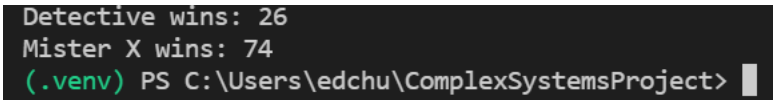


Figure 2: The results from running the game 100 times with RandomMove strategy for both MisterX and Detectives

4 Updated Strategy for Detectives

We now improve the strategy of the Detectives. To do this, we introduce a set of possible positions for MisterX from the perspective of the Detectives. The Detectives can calculate this set based on the positions of MisterX at the end of turns 3, 8, 13, 18 and the transport modes used on his turn. We will call the set '$PossLM_k$' with $PossLM_0 = V$. Then the following algorithm is executed by the Detectives prior to their turn.

Algorithm 4: PossLM SET UPDATE

Input: k=TURN, $PossLM_{k-1}$, LD^1_{k-1}, LD^2_{k-1}, LD^3_{k-1}, $XTransMode$
Output: $PossLM_k$

```
if k ∈ {3, 8, 13, 18} then
    PossLM = {LX_k}
else
    for vertex in PossLM_{k-1} do
        Add to PossLM_k all vertices adjacent to vertex in G(V, E_{XTransMode})
    PossLM_k = PossLM_k \ {LD^1_{k-1}, LD^2_{k-1}, LD^3_{k-1}}
Return PossLM_k
```

Following their turn the Detectives remove their own locations from $PossLM_k$ with the single operation

$$PossLM_k = PossLM_k \backslash \{LD^1_k, LD^2_k, LD^3_k\}$$

Using this algorithm, we will update the Detectives' strategy to one where all Detectives move towards the possible location of MisterX. To do this, we define a function for graphical distance.

Definition 4.1.

- For $u, v \in V$, we define $distance(u, v)$ to be the minimum length of a path from u to v on the graph $G(V, E_t \cup E_b \cup E_u)$.
- For $v \in V$ and $U \subset V$, we define:

$$distance(u, N) := \min\{distance(u, v) : v \in N\}$$

The Detectives execute the following algorithm to minimize the distance between their locations and $PossLM_k$. (Note: As MisterX's location is not revealed until turn 3, the Detectives use RandomMove on turns 1 and 2.)

Algorithm 5: DetectiveRush STRATEGY

Input: k=TURN, $PossLM_k$, LD^1_{k-1}, LD^2_{k-1}, LD^3_{k-1}
Output: $\{LD^1_k, LD^2_k, LD^3_k\}$

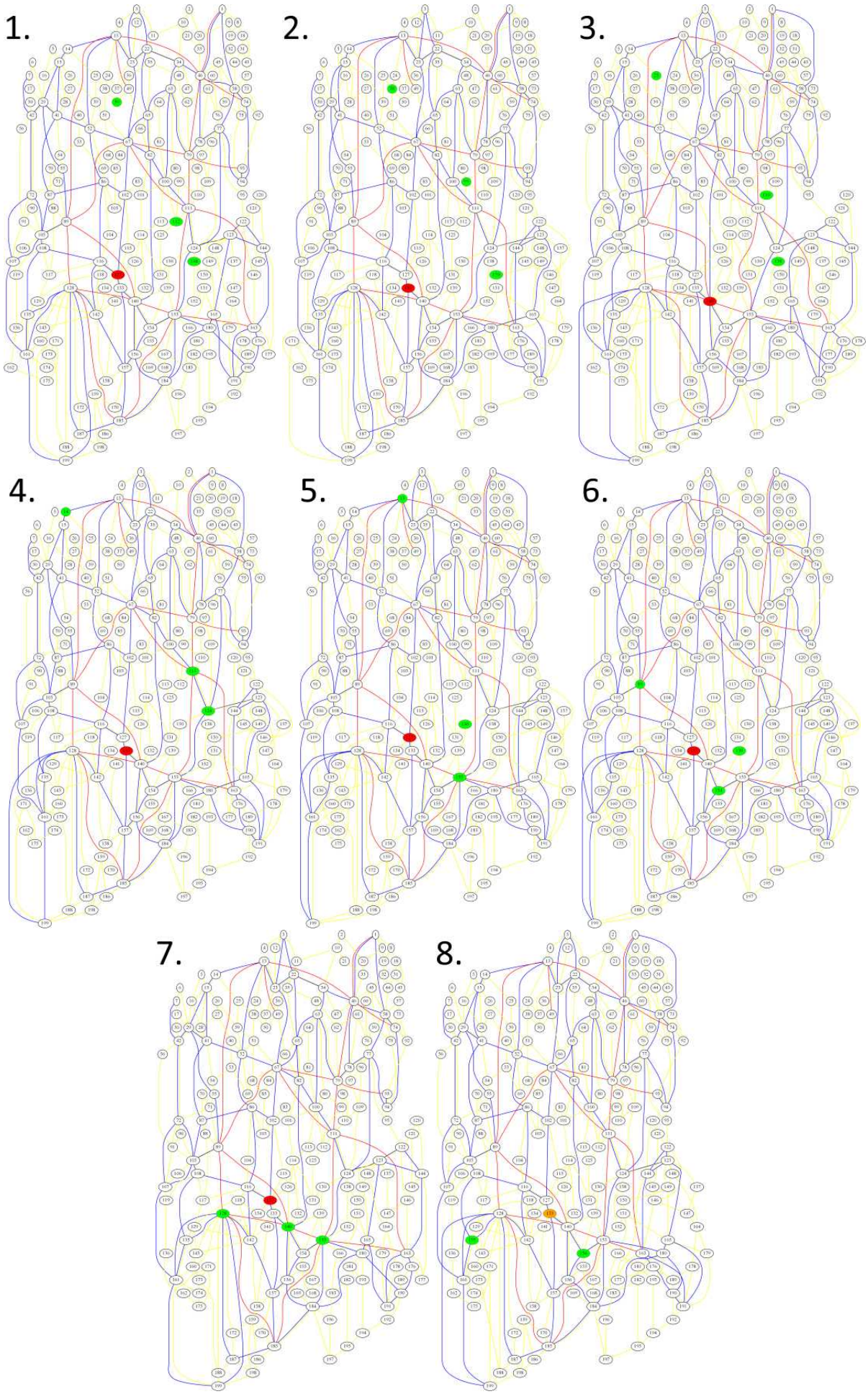
```
if k < 3 then
    LD^1_k, LD^2_k, LD^3_k = DetectiveRandom
else
    for j in {1,2,3} do
        Calculate D^j_k
        if D^j_k = ∅ then
            LD^j_k = LD^j_{k-1}
        else
            d = distance(LD^j_{k-1}, PossLM_k)
            for move in D^j_k do
                if distance(move,PossLM_k)< d then
                    LD^j_k = move
Return {LD^1_k, LD^2_k, LD^3_k}
```

Results

We ran the game 100 times with the MisterXRandom strategy for MisterX and DetectiveRush strategy for the Detectives. The Detectives won every time.

```
Detective wins: 100
Mister X wins: 0
(.venv) PS C:\Users\edchu\ComplexSystemsProject>
```

A Graphviz visualisation of a sample game shows the proficiency of the Detectives' strategy. The following figure shows the player locations at the end of turns 1-8 with the Detectives catching MisterX on the orange vertex on turn 8.


1.
2.
3.
4.
5.
6.
7.
8.

5 Updated Strategy for MisterX

We now improve MisterX's performance.

We first try a simple strategy to maximise the distance between MisterX and the nearest Detective.

Algorithm 6: MisterXRun STRATEGY 1

Input: k=TURN, LM_{k-1}, LD^1_{k-1}, LD^2_{k-1}, LD^3_{k-1}
Output: LM_k

Calculate M_k
if $M_k = \emptyset$ **then**
    Stop Detectives win
else
    $d = distance(LM_{k-1}, \{LD^1_{k-1}, LD^2_{k-1}, LD^3_{k-1}\})$
    $LM_k = LM_{k-1}$
    for *move in* M_k **do**
        if $distance(move, \{LD^1_{k-1}, LD^2_{k-1}, LD^3_{k-1}\}) > d$ **then**
            $LM_k = move$
    if $LM_k = LM_{k-1}$ **then**
        $LM_k = MisterXRandom$
    Return LM_k

The results for this were generally unsuccessful with MisterX winning only around 5% of games. We improve on this strategy by taking into account the distances to *all* Detectives. When considering a given move $v \in M_k$, we still want nearer Detectives to take priority. Therefore, for each Detective j, we calculate $3^{-distance(v, LD^j_k)}$. We then sum these values over j, obtaining

$$\sum_{j=1}^{3} 3^{-distance(v, LD^j_k)} \qquad (\dagger)$$

and choose $v \in M_K$ minimizing (†). Nearer detectives contribute more to the sum than farther ones. We implement this in the following algorithm.

Algorithm 7: MisterXRun STRATEGY 2

Input: k=TURN, LM_{k-1}, LD^1_{k-1}, LD^2_{k-1}, LD^3_{k-1}
Output: LM_k

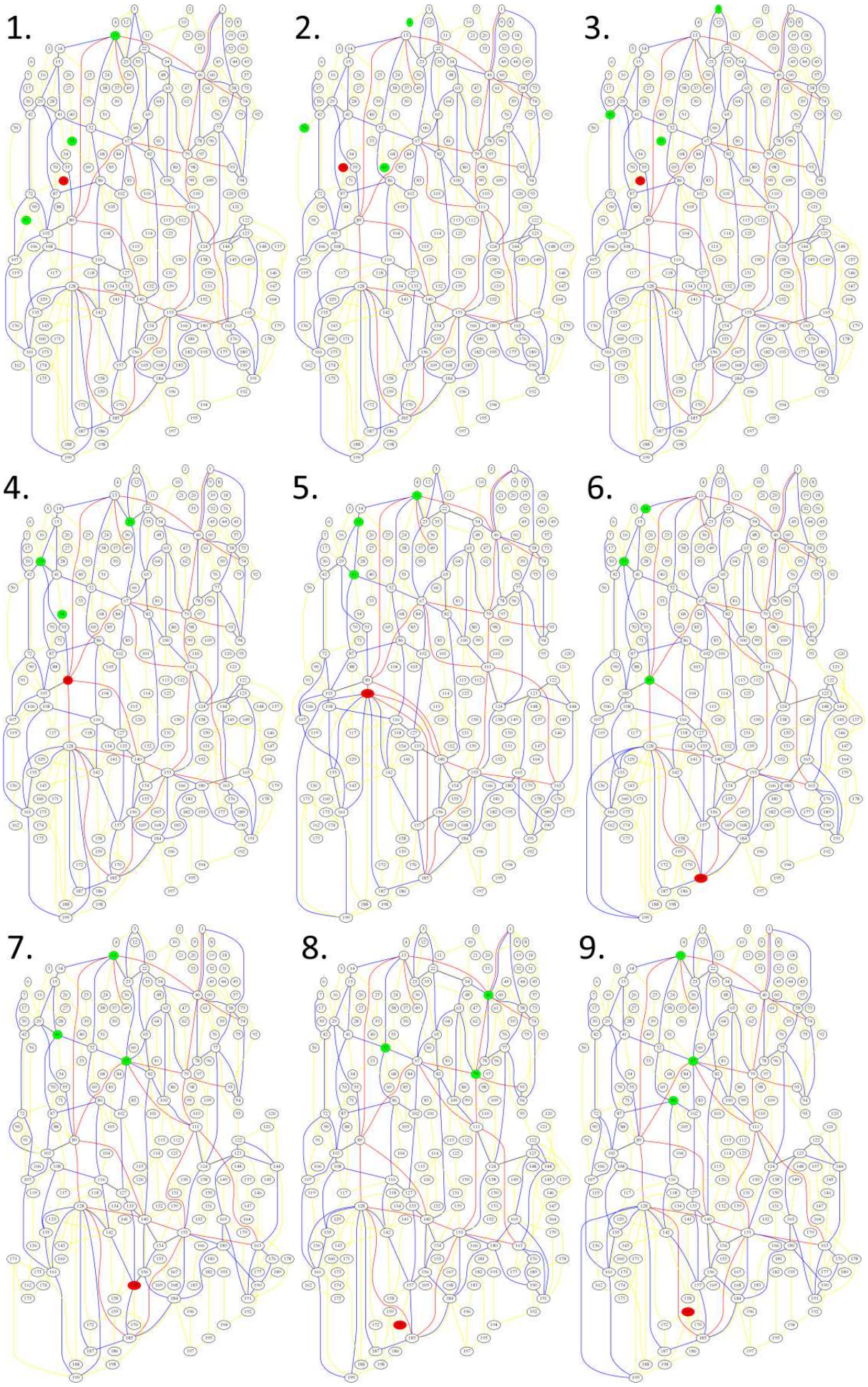
Calculate M_k
if $M_k = \emptyset$ **then**
    Stop Detectives win
else
    $LM_k = \arg\min_{v \in M_k} \sum_{j=1}^{3} (3^{-distance(v, LD^j_k)})$
    Return LM_k

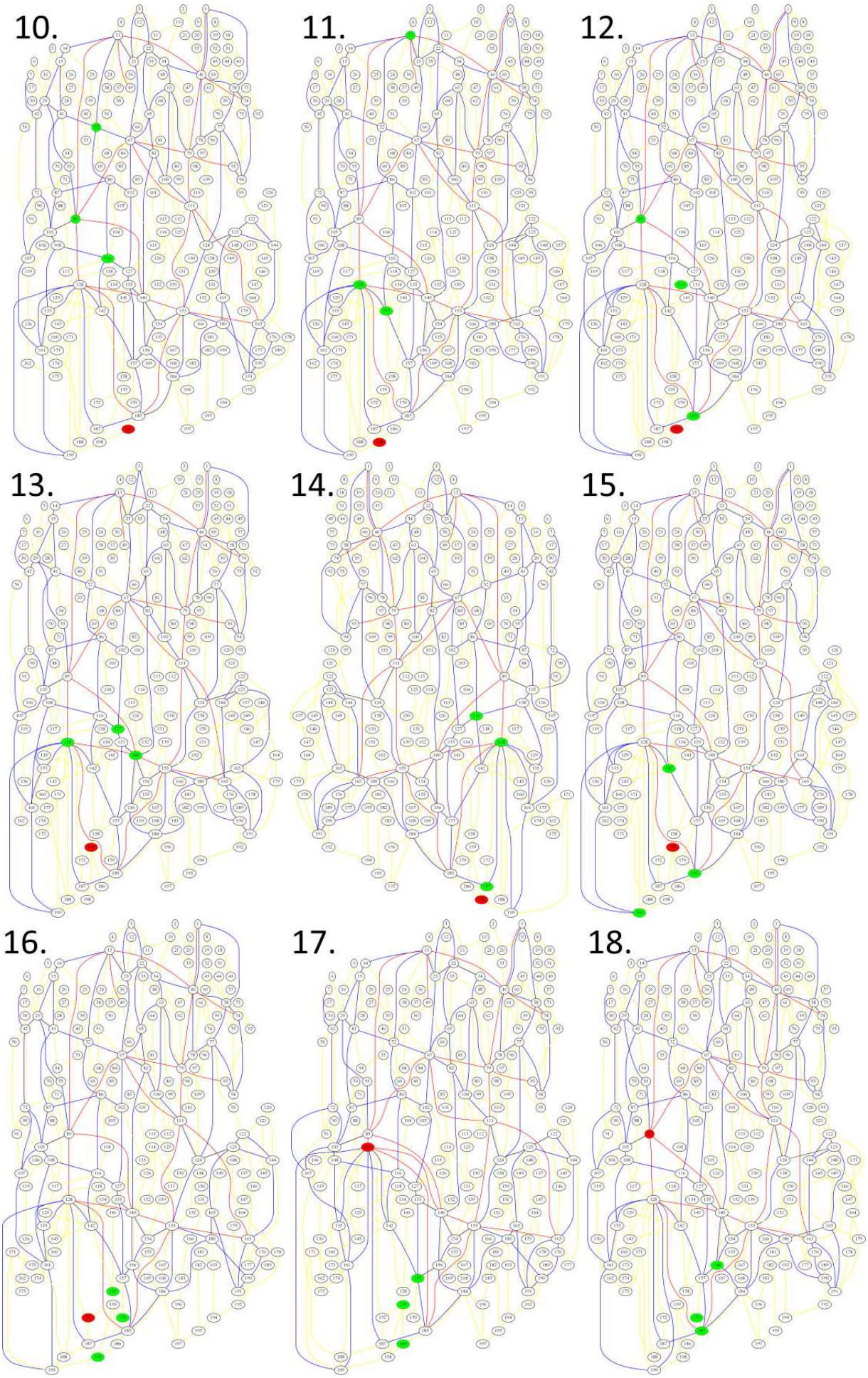
Results

We ran the game 100 times with the MisterXRun strategy for MisterX and DetectiveRush strategy for the Detectives. MisterX won 21 times and the Detectives won 79. We expected MisterX to win a greater portion of the time, however this is still a significant improvement from the MisterXRandom strategy.

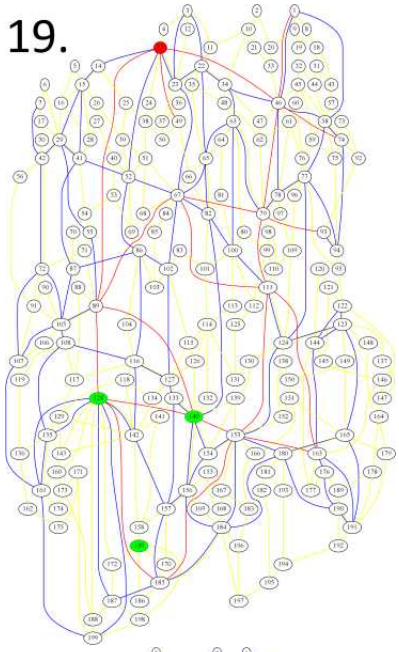
```
Detective wins: 79
Mister X wins: 21
(.venv) PS C:\Users\edchu\ComplexSystemsProject>
```

A Graphviz visualisation of a sample game shows an example of MisterX evading the Detectives. The following figure shows the player locations at the end of turns 1-24.

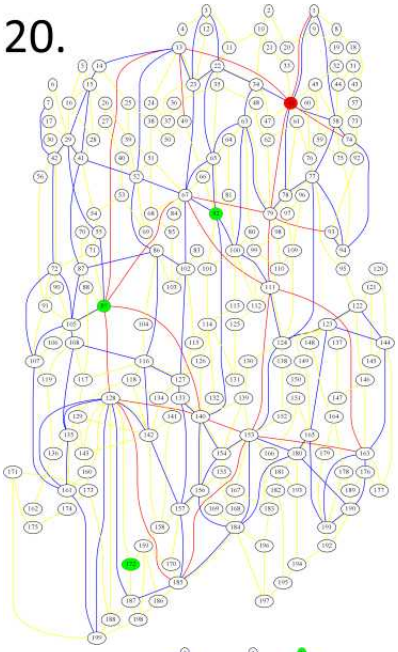

1.
2.
3.
4.
5.
6.
7.
8.
9.


10.
11.
12.
13.
14.
15.
16.
17.
18.

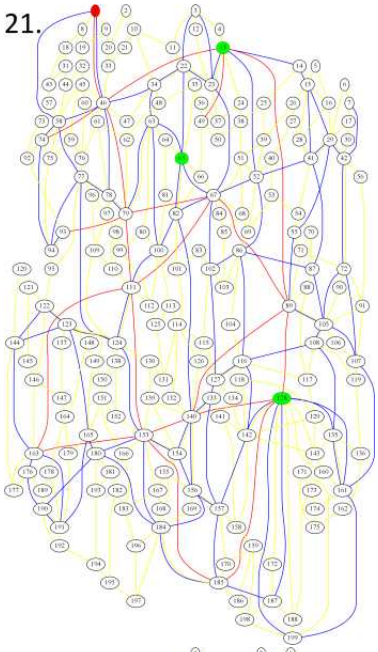
19.



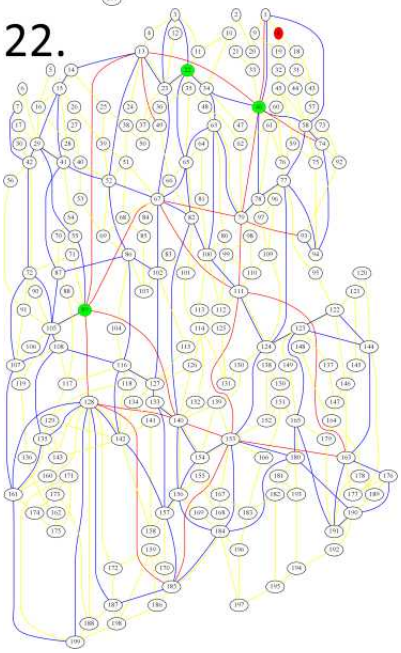
20.



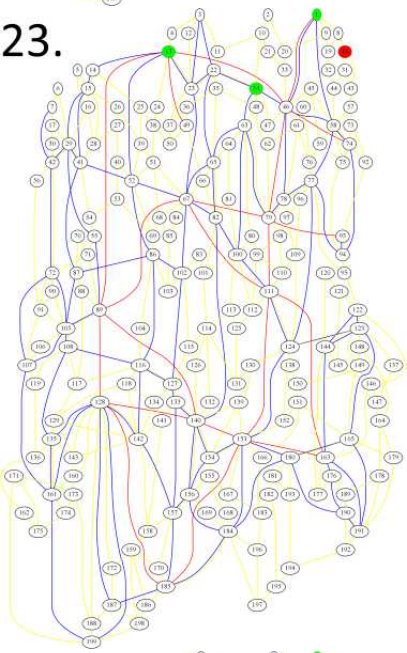
21.



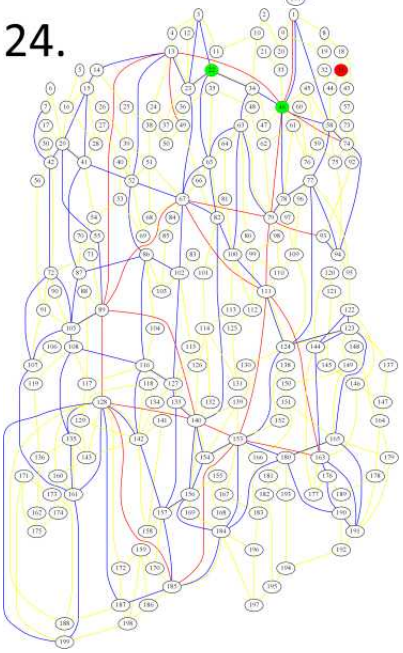
22.



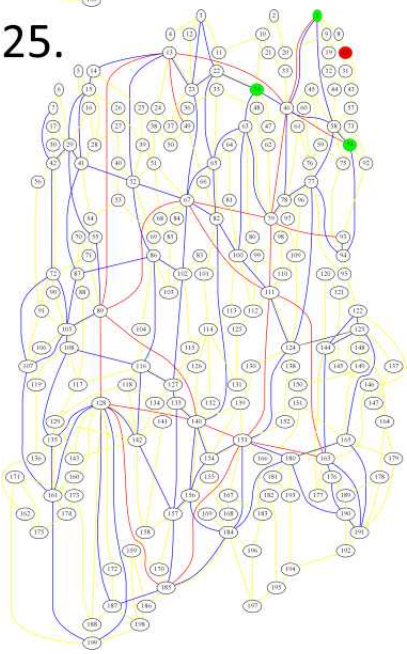
23.



24.



25.



6 Conclusion and Further Possibilities

We have modelled the game Scotaland Yard and created a relatively balanced pair of strategies for MisterX and the Detectives.

To continue this project, we would improve the model of the game to include transport tickets for the Detectives and power moves for MisterX. We would then make strategies from the DetectiveRush and MisterXRun algorithms to include the new rules. New strategies would then be developed for MisterX and Detectives to capitalise on these new rules. As this is a much closer model of the game, we would expect to achieve a win ratio closer to 50:50.

While we managed to improve MisterX's performance in Section 5, we would like to explore the possibility of improving it even further by applying Machine Learning. To do this, we would carry out the following steps:

- Model the game as a binary classification problem with two possible classes: a MisterX victory or a MisterX loss.
- Use the Detective location data as training data
- Use a binary classification algorithm such as a Decision Tree or a Neural Network to train MisterX to evade the Detectives.