

Modelling Scotland Yard

Fredderf and Ed0G

December 28, 2021

1 Introduction

We concentrate on a popular search game called Scotland Yard involving one player, 'MisterX' and 1-5 'Detective' players. 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 an adjacent vertex. The aim of the Detectives is to 'capture' MisterX by landing on the same vertex as him or not allowing him to move and the aim of MisterX to stop this from happening. If MisterX is captured the detectives win, whereas if MisterX can last 24 turns without being captured they win.

The game is begun by all players choosing at random a starting tile. The detectives locations are at all times common knowledge however the location of MisterX is only revealed 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 adjacent vertex and records it covering it up with the ticket used to get there (taxi, bus, underground). If this is not possible due to surrounding 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 MisterX to hide his mode of transport on a turn, and 2 double move tokens, which allow him to take 2 turns before the detective get a chance to move.

There are always at least 4 searchers in the game. When there are less than 4 detective players then up to 2 of the 4 players are 'Policemen' and do not have the ticket restriction that detectives have. These players are controlled jointly by the searcher players.

2 Model of the game

We will model a simplified version of the game in order to improve the performance of the player algorithms. We will set the searchers to 4 remove

of rules involving travel tickets for the detectives, essentially making all detectives policemen (we will continue to refer to all searchers as detectives throughout). In balance, we will further simplify the game by removing all double move tokens and blank tokens.

We make an undirected graph with vertex set $V = \{1, 2, \dots, 199\}$ and 3 edge sets E_t , E_b and E_u to represent the board and taxi, bus and underground edges respectively.

Definition 2.1. We define the following variables for $k \in 0, 1, 2, \dots, 24$ and $j \in \{1, 2, 3, 4\}$:

- for a finite set S , we call $random(S)$, the element chosen uniformly at random from S .
- M_k is the set of possible moves 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 .
- LM_k is the location (vertex) of MisterX at the end of turn k .
- LD_k^j is the location (vertex) of Detective j

The game runs as follows:

Algorithm 1: SCOTLAND YARD GAME

Input: $LM_0, LD_0^1, LD_0^2, LD_0^3, LD_0^4$
Output: Winner

```

1 for  $TURN$  in  $1, 2, \dots, 24$  do
2   MisterXMove(Input =  $TURN$ )
3   DetectivesMove(Input =  $TURN$ )
4   if  $LM_k \in \{LD_k^1, LD_k^2, LD_k^3, LD_k^4\}$  then
5     STOP: Detectives win
6 STOP: MisterX Wins

```

2.1 Modelling in Python

We use python to model the game. The file `generate_graphs.py` contains functions relating to drawing out the graph in the form of a Graphviz object. When it adds nodes, it also highlights the positions of MisterX and the Detectives.

The `player.py` file makes the classes `MisterX` and `Detective` which includes initialising each by choosing an initial location from the starting locations. The each class also includes a function to find a list of possible moves from their current location.

We will use `play_game.py` to run the model of a single game. The `initialise_game` function creates a MisterX and 4 Detectives and ensures all Detectives have different starting locations. It then contains the code to run the game.

3 Initial Strategies

As an initial strategy we will select the move of every player by choosing a vertex uniformly at random from the set of unoccupied vertices adjacent to his current vertex location.

Algorithm 2: MISTERX RandomMove STRATEGY

Input: $k=\text{TURN}$, LM_{k-1} , LD_{k-1}^1 , LD_{k-1}^2 , LD_{k-1}^3 , LD_{k-1}^4
Output: LM_k

- 1 Calculate M_k
- 2 **if** $M_k = \emptyset$ **then**
- 3 **Stop** Detectives win
- 4 **else**
- 5 **Return** $\text{Randomchoice}(M_k)$

Algorithm 3: DETECTIVES RandomMove STRATEGY

Input: $k=\text{TURN}$, LD_{k-1}^1 , LD_{k-1}^2 , LD_{k-1}^3 , LD_{k-1}^4
Output: LD_k^1 , LD_k^2 , LD_k^3 , LD_k^4

- 1 $\text{move} = []$
- 2 **for** j in 1,2,3,4 **do**
- 3 Calculate D_k^j
- 4 **if** $D_k^j = \emptyset$ **then**
- 5 $LD_k^j = LD_{k-1}^j$ to move
- 6 **else**
- 7 $LD_k^j = \text{Randomchoice}(D_k^j)$
- 8 **Return** LD_k^1 , LD_k^2 , LD_k^3 , LD_k^4

We ran the game with the RandomMove strategy for both MisterX and Detectives 100,000 times. Detectives won 36309 times and MisterX won 63691. We expected MisterX to win the majority of the time however were expecting a larger majority.

4 Updated Strategy for Detectives

We next improve the strategy of the detectives to improve their performance. To do this we will introduce a set of possible positions for MisterX based on common knowledge in the game. We will call the set '*PossXLocation*'. The following algorithm will be run by Detectives just before their turn.

Algorithm 4: PossXLocation SET UPDATE

Input: $k=\text{TURN}$, *PossXLocation*, LD_{k-1}^1 , LD_{k-1}^2 , LD_{k-1}^3 , LD_{k-1}^4 , *XTransMode*
Output: *PossXLocation*

```

1 if  $k \in \{3, 8, 13, 18\}$  then
2    $PossXLocation = \{LX_k\}$ 
3 else
4    $temp = \{\}$ 
5   for vertex in PossXLocation do
6     Add to temp all vertices adjacent to vertex in
        $G(V, E_{XTransMode})$ 
7    $PossXLocation = temp \setminus \{LD_{k-1}^1, LD_{k-1}^2, LD_{k-1}^3, LD_{k-1}^4\}$ 
8 Return PossXLocation

```

Following their turn the Detectives will run the single operation

$$PossXLocation = temp \setminus \{LD_k^1, LD_k^2, LD_k^3, LD_k^4\}$$

Using this we update the detectives strategy on their turn to one where all detectives move towards the possible location of MisterX. For this we use the function $distance(x, y)$ to be the graphical distance between 2 sets of vertices (ie. the minimum length of a path from a vertex of set X to a vertex of set Y).

Algorithm 5: DETECTIVES RUSH STRATEGY

Input: $k=\text{TURN}$, PossXLocation , LD_{k-1}^1 , LD_{k-1}^2 , LD_{k-1}^3 , LD_{k-1}^4

Output: $\{LD_k^1, LD_k^2, LD_k^3, LD_k^4\}$

```
1 if  $k < 3$  then
2    $LD_k^1 = \text{RandomMove}$ 
3    $LD_k^2 = \text{RandomMove}$ 
4    $LD_k^3 = \text{RandomMove}$ 
5    $LD_k^4 = \text{RandomMove}$ 
6 else
7   for  $j$  in  $1, 2, 3, 4$  do
8      $d = \text{distance}(LD_{k-1}^j, \text{PossXLocation})$ 
9     for  $\text{move}$  in  $D_{k-1}^j$  do
10      if  $\text{distance}(\text{move}, \text{PossXLocation}) < d$  then
11         $LD_k^j = \text{move}$ 
12 Return  $\{LD_k^1, LD_k^2, LD_k^3, LD_k^4\}$ 
```
