Bachelor Project

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Heuristic Evaluation in the Scotland Yard Game

Daniel Borák

Supervisor: RNDr. Vojtěch Kovařík, Ph.D.
January 2021

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Borák Daniel** Personal ID number: **469899**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Branch of study: **Computer and Information Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Heuristic Evaluation in the Scotland Yard Game

Bachelor's thesis title in Czech:

Heuristiky pro hru Scotland yard

Guidelines:

Scotland Yard is a board game where a team of "detectives" moves around a large graph in an attempt to catch a "criminal". An interesting feature of the game is that while the criminal has full information, the detectives only receive hints about his movements. The goal of the thesis is to:
1) Get familiar with the main game-theoretical models relevant to Scotland Yard (extensive form games, one-sided partially-observable stochastic games).
2) Review the solution methods applicable to Scotland Yard.
3) (a) Start with the algorithm "MCTS + distance-to-the-nearest-detective heuristic algorithm" for solving Scotland Yard.
(b) Design a substantial extension of (a) based on (2).
4) Compare the empirical performance of (3b) to that of (3a) and other available baselines.

Bibliography / sources:

[1] Nijssen, Pim, and Mark HM Winands. "Monte carlo tree search for the hide-and-seek game scotland yard." IEEE Transactions on Computational Intelligence and AI in Games 4.4 (2012): 282-294.
[2] Erik Vaknin, "Strategy Generation for Partially Observable Stochastic Games" (2019). FEL CVUT.
[3] Zinkevich, Martin, et al. "Regret minimization in games with incomplete information." Advances in neural information processing systems. 2008.
[4] Kovařík, Vojtěch, et al. "Rethinking formal models of partially observable multiagent decision making." arXiv preprint arXiv:1906.11110 (2019).

Name and workplace of bachelor's thesis supervisor:

RNDr. Vojtěch Kovařík, Ph.D., Artificial Intelligence Center, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **10.01.2020** Deadline for bachelor thesis submission: **05.01.2021**

Assignment valid until: **30.09.2021**

RNDr. Vojtěch Kovařík, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt

Student’s signature

Acknowledgements

I would like to thank my professors and instructors of Czech Technical University in Prague for the knowledge and experiences that I gained here. I am grateful to all my friends for their assistance throughout my studies, and the wonderful times we spent together. At least but not last I would like to thank my family for all the support they gave me during my studies, for their continuous help and love.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses. Prague, 5. January 2021

Abstract

Scotland Yard can be characterised as 2-player one-sided partially observable stochastic game. We will take a look at current research used in this and similar games, and design our own improvements and heuristics. At the end we will evaluate results of experiments and evaluate their efficiency.

Keywords: Game theory, Monte-Carlo Tree Search, Scotland Yard, board game, partially observable stochastic games

Supervisor: RNDr. Vojtěch Kovařík, Ph.D.

Abstrakt

Scotland Yard se dá charakterizovat jako jednostraně částečně pozorovatelná stochastická hra. Podíváme se na současný výzkum v této a podobných hrách, a navrhneme naše vlastní vylepšení a heuristiky. Na závěr posoudíme výsledky experimentů a vyhodnotíme jejich účinnost.

Klíčová slova: Teorie her, Monte-Carlo Tree Search, Scotland Yard, deskové hry, částečně pozorovatelné stochastické hry

Překlad názvu: Heuristiky pro hru Scotland Yard

Contents

Figures

Tables

Chapter 1

Introduction

The thought of playing games against a computer has appeared soon after its creation. In 1945, Alan Turing used chess as an example of computer's possibilities, and in 1950 he wrote the first computer chess program. It the same year Alan Turing proposed the Turing Test, predicting that one day, computers could be programmed at level where humans would not know whether they faced human or a machine. In 1997, DEEP BLUE won against Gary Kasparov in a 6 game match, and for the first time computer beat a reigning world champion. In 2015, AlphaGo conquered the game of Go, in 2017 Libratus and Deepstack beat humans in game of no-limit Texas hold'em and in 2019 OpenAI Five defeated world's best team in a game of DotA 2. In this thesis, we will take a look at some of the algorithms behind two of these programs and analyse, whether they can be used in a game of Scotland Yard.

Scotland Yard can be characterised as 2-player one-sided partially observable stochastic game. Chess and Go are perfect information game, with AlphaGo and its successors based on combination of Monte-Carlo Tree Search (MCTS) and neural networks. No-limit Texas hold'em is imperfect information game, and we will take look at Deepstack, which has combined MCTS and Counterfactual regret minimisation (CFR) at its core.

After we choose which algorithms to use, we will consider various heruistics for MCTS, evaluate result of the experiments, and discuss the results.

Chapter 2

Scotland Yard

Scotland Yard is a board game designed for 3-6 players, originally released in 1983 by Ravensburger. There are several versions of the game with slightly different rules, but we shall be considering only the original version of the game [1]. There are 6 agents: 5 detectives and Mr.X. Since detectives work in a team with the same victory condition, it is essentially a 2-player game.

Game is played on a graph with numbered locations, connected by 4 types of edges for transportation: taxi, bus, underground and boat. Individual transportation types differ from each other by possible distance traveled and number of nodes where the transportation type can be used. At start, each detective has 10 taxi, 8 bus and 4 underground tickets. Mr.X has 4 taxi, 3 bus and 3 underground tickets, 2 double move (2x) tickets and as many black tickets as there are detectives in the game, in this case 5. Two detectives are not allowed to stand at the same location at the same time.

At the beginning of the game, all players have their starting location chosen randomly of 18 pre-defined locations and place their game piece at the assigned location, but Mr.X keeps his location hidden. Current position of Mr.X is revealed only on rounds 3, 8, 13, 18 and 24. Every round begins with a turn of Mr.X, who chooses where he wants to travel, and pays with either the appropriate ticket or a black ticket. Black ticket can be used for any type of transportation, allowing Mr.X to keep his choice of transportation secret, and it's the only way to travel via boat. After that, Mr.X may use 2x move ticket (if he still has one), and can play another turn. This counts as a new round, increasing the round counter and therefore round where Mr.X has to reveal his position comes sooner. After Mr.X ends his move, each detective makes their move in fixed order, move their game piece and give the used ticket to Mr.X. Mr.X wins if he is not caught within

24 rounds. Detectives win if one of the detectives is located at the same position as Mr.X at any point of the game.

2.1 Challenges of Scotland Yard

Size of the game
Game of Scotland Yard has a large number of possible states, and as such we have to consider possible simplifications. Concerning the board, there are no symmetries, and the turn order of detectives matters as well, making a difference between one player leaving a square and another entering or blocking the path of a fellow teammate. The number of tickets matters as well, since being unable to use some kind of transportation severely limits detectives mobility and can be abused by Mr.X, and likewise, not having double move or black tickets limits Mr.X's ability to flee.
Let's make an assumption that for each of 200 possible positions, there are 50 or so reasonable spaces for detectives to be at a given time for last revealed position of Mr.X and his subsequent moves. That alone gives us over one hundred million possible states, without even considering remaining tickets among the players and hiding spot of Mr.X.

Cooperation between agents Teamwork is necessary for detectives, both for capturing Mr.X and movement across the map. Without properly surrounding area around Mr.X, there is only a small chance of capturing him during turns when he stays hidden, and during transportation there is a chance of blocking teammates path and wasting their turns.

Imperfect information Since location of Mr.X is hidden most of the time, detectives have to make estimates about his position. We have information set of all possible positions of Mr.X, and as such we are able to narrow down possible hiding locations.

Mr.X's advantages Mr.X has two actions available only to him. The first one is ability to use double move. When used right, Mr.X can escape from a surrounded area or create a gap between him and seekers. Second special action is black move. Black move not only allows Mr.X to use boat transport, it also allows him to confuse his opponents. In situations where multiple types of travel are available across all possible locations of Mr.X, the number of possible locations increases significantly for the following turns.

Chapter 3

Previous research on Scotland Yard and other imperfect information games

As of may 2020, there are currently two directions of research done on Scotland Yard; the first is based on Monte Carlo Tree Search (MCTS) [9], [10] and the second tackles the problem with adversarial neural networks and Q-learning [11].
Strengths of MCTS algorithm are based around its progressively built search tree, which requires only a simple heuristic and does not need any heuristic for the current state. The algorithm can be stopped at any time, and since it is domain independent algorithm, it can be modified quickly for use on many problems. However, it requires a lot of computational resources during its run, and with a higher branching factor the state space may not be searched thoroughly enough to give a reasonable estimate in reasonable time. It has been successfully used in games with complete information like Go [18], shogi [4] and chess [4], real-time games such as Ms. Pac-Man [17], and in games with incomplete information like Settlers of Catan [3], Kriegspiel [2] and poker.
Current state of the art algorithm is AlphaGo [4] by Deepmind, which combines MCTS with machine learning, and has managed to beat the best human and computer players in Go, shogi and chess.
Adversarial neural networks and Q-learning is a new approach to the problematic of Scotland Yard, published as recently as fall 2018.[11] However, this model has not implemented full rules of Scotland Yard which limits strengths of Mr.X, and yet achieve lower win rate for detectives than MCTS approach. As such we have decided to use results from MCTS Scotland Yard model from [9] as our baseline.

Chapter 4

Theory

4.1 Normal and extensive-form game

Normal-form game

Normal-form game is a tuple (N, S, U), where N = 1..n is a set of players, S_i is a set of strategies for each player and U is a set of utility functions for each player, which assign utility value $u_i(s)$ based on chosen strategy of players $s = (s_1..s_n)$. It is usually represented in a tabular form.

Extensive-form game

Extensive-form game is used to model games with a finite and predetermined horizon. EFG are usually showcased as a tree, with nodes of the tree representing game state and outgoing directed edges as actions that can be taken by a player, leading to a new game state.
Unlike normal-form game, in extensive-form game we can represent randomness and sequential games.

	R	P	S
R	$(0,0)$	$(-1,1)$	$(1,-1)$
P	$(1,-1)$	$(0,0)$	$(-1,1)$
S	$(-1,1)$	$(1,-1)$	$(0,0)$

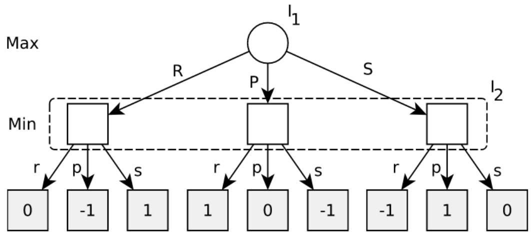


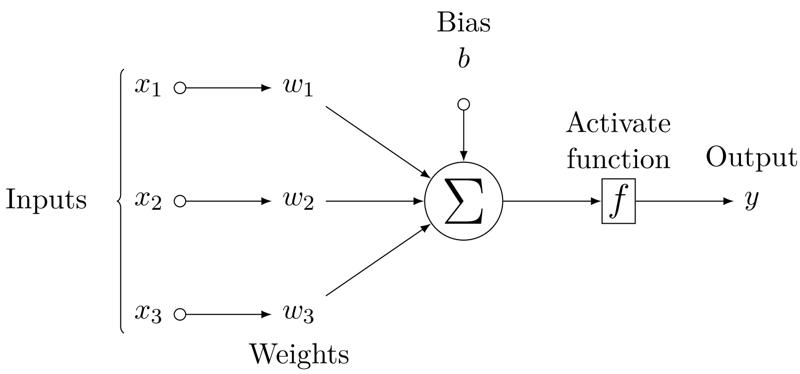
Figure 4.1: Example for normal-form game and imperfect information extensive-form game. Source:[20]

Imperfect information extensive-form game

Imperfect information extensive-form game (EFG) is a tuple $G = ((N, A, H, Z, \rho, \chi, \varphi, u), I)$, where N = 1..n is a set of players, A is set of actions, H is a finite set of all possible histories of actions taken from the root, Z is set of all terminal states of the game where $Z \in H$ represent leaves of a game tree, ρ: H -> N is a player function which returns current player for a given node, φ function returns available actions for current state, utility function u_i: Z -> R assigns value of the terminal node. Information set I is a set of equivalence on decision nodes of a player I with the property ρ(h) = ρ(h') = I and χ(h) = χ(h'), for $h, h' \in I$ for an information $I \in I_i$.

4.2 Artificial neural network

Artificial neural network (ANN) is graph composed of nodes and edges, where nodes are called neuron and edges are named synapses. Every node is composed of three basic components:



Weight vector **w**, which assigns weights to incoming edges **x**, summation function and activation function, which maps result to the outgoing edge. This function is usually non-linear and differentiable, for example ReLu units, sigmoid or SoftMax.

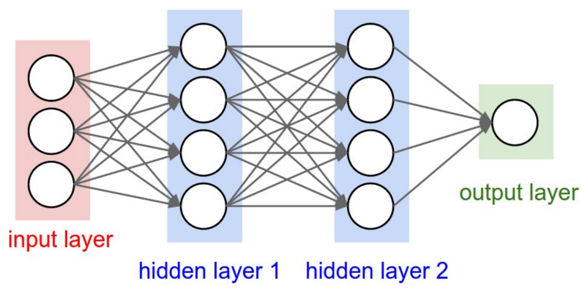


Figure 4.2: Example of neural network with 4 layers. Source:[19]

In general, neural network has several layers. First is called input layer, last is output layer and layers in between are called hidden layers.

4.3 Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is a best first search algorithm, which does not require a state evaluation function. From current state, the algorithm builds a search tree, using results from previous iterations of the

MCTS algorithm to guide growth of the search tree. The values of the nodes of the search tree are used to estimate values of moves, which get progressively more accurate with the growing size of the search tree.

MCTS is composed of four basic steps:

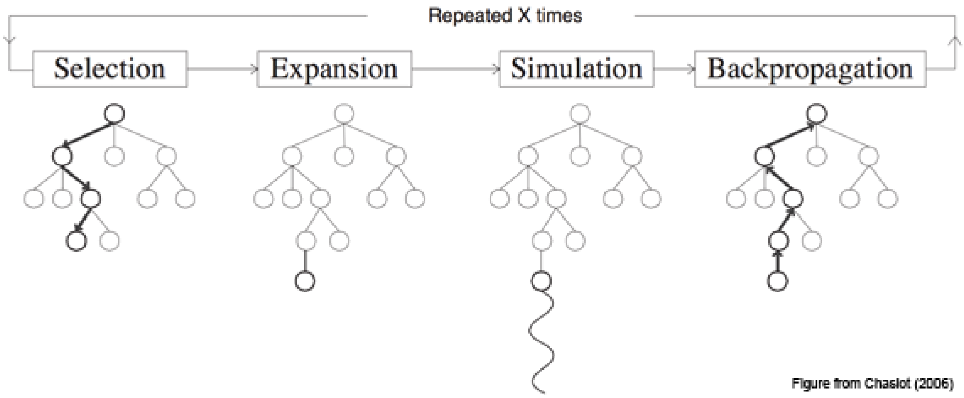


Figure 4.3: Four steps of MCTS algorithm

Selection
Starting from the root node, the search tree is traversed using a strategy until we reach a node L that is not fully expanded. For selection of the child nodes to traverse, we use the Upper Condidence Bounds for Trees (UCT) selection strategy [6], which is a variant of Upper Confidence Bound [7] algorithm used in multiarmed bandit problem. It is represented by formula

$$UCB_i = x_i + C \cdot \sqrt{\frac{lnN}{n_i}}$$

where UCB_i is Upper Confidence Bound of the child node, x_i is estimated value of the child node, n_i is number of times child node has been visited and N is total number of visits of the parent node. C is a bias parameter (also called exploration parameter), which is tuned during testing. This formula combines exploration of known reward value x_i and exploitation of relatively less visited nodes to increase their chances of being visited.

Reward value is based on playout, which is not a reliable evaluation function and as such has to be used many times to increase reliability. Usually first searches are far from terminal nodes, and with distance from terminal nodes getting lower the estimate converges to become more reliable.

Expansion
If L is not a terminal node, and L is not yet fully expanded, then for one of the possible moves, which have not been yet expanded, a new child node C is created, which we then use for the next step.

Simulation/Playout
Run a simulated playout from C until a result is achieved. Playout can be either random or use simple heuristic. Basic MCTS algorithm assumes that it has all information available, which is not the case in our problem. We will elaborate on the playout later in section 5.1.

Backpropagation
Result from simulation is used to update value of all nodes which have been visited in current iteration.

After either time runs out or fixed number of iterations is reached, the child of the root that is considered to be the best out of them is then selected. From [16], some possible options are such as:

- Max child: The child of the root with highest reward is chosen as best move.
- Robust child: The child of the root with highest number of visits is chosen as best move.
- Max-Robust child: The child of the root with both highest reward and highest number of visits is chosen as best move. If no child node satisfies these conditions, search continues until such child node exists.

In this project, Robust child is used.

Chapter 5

MCTS improvements for Scotland Yard

Domain of possible locations of Mr.X
We keep a list L of Mr.X‘s possible positions. Every time Mr.X moves with ticket t, we create a new list L‘ based on following rule: all locations from any location in L that can be reached by using ticket t are added to list L‘. After that we replace list L with L‘. Every time detective moves, whenever they step on location S that is in L, remove S from L.

Coalition reduction [9]
As stated before, all detectives are cooperating and can therefore be considered as 1 player. When a detective wins, it is considered as a win of every detective and is backpropagated as such. However, that can lead to situations, where a particular detective might rely too much on other detectives and not involve themselves in the search; for example, the detective can be on the other side of map to where all other agents, including Mr.X, are, and yet they'd earn the same score as those who are close to Mr.X. To prevent that, we use Coalition reduction. If detective that is currently a root player captures Mr.X, they earn a score of 1. However, if another detective captures Mr.X, they earn lower score, $1 - r$, where $r = [0, 1]$. If value of r is too small, detective might rely too much on his companions. However, if value of r is too large, they might cease cooperation with other detectives. The parameter r can be fine tuned through experimentation.

Move filtering [9]
Mr.X has only limited amount of double move tickets and black tickets, and

as such we don't want him to use them when there's no benefit in it. In the current version, the following is implemented:

- When from all of Mr.X's possible locations lead only taxi paths, black ticket won't be used, since it would not change the number of possible locations of Mr.X
- No black ticket or double move before turn 3. Due to sheer number of possible locations, there is no sizeable benefit.
- Double move is used only when average distance of detectives from Mr.X is below threshold

$\mathcal{E}$-greedy playout strategy [9] [14]
During playout stage of the MCTS algorithm, we can make either random actions or implement simple heuristic. In the multiarmed bandit problem, one such heuristic selects maximum possible reward among all actions according to our current knowledge, which is called greedy strategy. However, that would mean that no other actions are visited. Because of that, small probability $\mathcal{E}$ is used to make random action among all possible actions. As number of iterations continues to grow, all actions keep on converging to their true reward value.

At the beginning of the search, Mr.X is placed to a set location; for Mr.X this position is his real one, for Detectives it is estimated position. For our agent, following heuristics are used:

Mr.X's strategy: Minimum distance
Move to locations that gives maximum possible distance between Mr.X and closest detective.

Detective's strategy: Minimum distance to estimated position
Move to locations that gives minimum possible distance between estimated position of Mr.X and closest detective.
Position of Mr.X is estimated after each move by using Weighted roulette-wheel selection 5.1.1, which is discussed is later section. If detective reaches estimated position of Mr.X, but Mr.X is not there, Weighted roulette-wheel selection is used to determine his new estimated position. On rounds where Mr.X has to reveal his position, the estimated position is set to his real position.

5.1 Playout phase improvement

Perfect Information Monte Carlo Tree Search (PIMCTS) [12]
PIMCTS is a technique for playing imperfect information games with a search tree too large to be optimally searched. In order to turn imperfect information into perfect, we have to change all unknown variables into known variables. It can be done randomly, so that unknown variables are set to one of the possible values, or we can use an algorithm to reduce randomness.

Incomplete information playout
We're using $\mathcal{E}$-greedy playout strategy to estimate value of a node. Since it requires current player to know location of all players, including Mr.X, before we start building the search tree with MCTS, we have to estimate Mr.X's current position.

5.1.1 Localisation of Mr.X

The basic MCTS algorithm was designed for perfect information games. However, in most cases the current location of Mr.X is unknown.

Weighted roulette-wheel selection [9]
Each possible position of Mr.X is given a weight w_i based on minimum distance from a nearest detective, and then we randomly choose one of them with chance $\frac{w_i}{\sum w}$ These weights can be fine-tuned through experimentation.

MCTS search localisation of Mr.X
For each possible position of Mr.X we perform a small number of iterations of MCTS algorithm, setting these positions as location of Mr.X, and we count number of wins in these playouts. The position which has the lowest possible number of wins may be considered as weak from detectives perspective, and therefore strong from Mr.X's view, and we set this as a location of Mr.X with remaining number of iterations.

5.1.2 Estimating first two moves with machine learning

During first two moves, we can try to estimate what is the usual move of detectives and play it, saving us time, and possibly improving our chance to win as detectives can sometimes make suboptimal moves when the amount of Mr.X's possible locations is too high and the estimated position is off too a side.

5.1.3 Finding best move through multiple searches - Mr.X

Just like we use multiple searches for localization of Mr.X, we can use multiple searches for a double move. First we portion total amount of iterations for next move between normal search and double move. First we find the best action for the current state, which gives us an estimate of value of next state. After we perform the action, we do another search from new position, and if value of the supposed double move state is higher by certain portion than the state after first action, we perform a double move. We have written the formula as following:

$$DoubleMoveEstimate >= NormalMoveEstimate * (1 + multiplierValue)$$

If the equation holds true, we perform a double move.

Chapter 6

Experiment

We have used following settings in all experiments unless stated otherwise: MCTS iterations = 10000, following 3 parameters from [9]: detective's exploration parameter C = 2, hider's exploration parameter C = 0.2, coalition reduction parameter = 0.25 and when choosing position of Mr.X with the roulette-wheel selection method 5.1.1 we use parameters [0.196, 0.671, 0.540, 0.384, 0.196] assigned to distances 1, 2, 3, 4 and more than 4, $\mathcal{E}$-greedy playout strategy for $\mathcal{E} = 0.2$, limit usage of black tickets as stated in Chapter 5, use double move on condition that average distance of all detectives from Mr.X is smaller than 3. Each experiment is run 900 times, with each starting position used the same number of times for Mr.X to reduce bias of starting locations.

6.1 Experiment 1: Comparison with previous work

For this experiment we use all parameters from [9] agent and compare our results. When we disable double move option, we get a win rate for detectives of $62.2\% \pm 3.1$. In quoted article with same settings they reached win rate of $66.0\% \pm 1.9$, however, some parameters might be different. There are several versions of Scotland Yard map with small differences, and in our playout we still use tickets and assume they do not.
For a version with double move enabled the final win rate was $46.8\% \pm 3.3$

6.2 Experiment 2: MCTS localisation of Mr.X

In this experiment we test whether localisation of Mr.X can be improved by using winrate from MCTS, where position with lowest win rate of detectives is used as a position of Mr.X in subsequent search which selects action to be used

	Split (localisation/action search)	Win rate (95 % confidence)
1	10/90	59.8% $\pm$ 3.2
2	25/75	57.6% $\pm$ 3.2
3	50/50	56.9% $\pm$ 3.2
4	25/75	55.3% $\pm$ 3.3

Table 6.1: Experiment 2 - Detective win rate of various splits of 10,000 MCTS iterations

The best option of localisation improves our win rate from 46.8% $\pm$ 3.3 to 59.8% $\pm$ 3.2 and gets better result for each tested setting.

6.3 Experiment 3: Influence of number of iterations on win rate

We use different amount of iterations for seeker, hider or both. In detective/Mr.X column use number of iterations in the left most column, Mr.X/detective uses 10.000 iterations, and in final column both agents use the same number of iterations. Default settings are used along with MCTS localisation using 25/75 split.

Iterations	Detective	Hider	Both
40000	60.5% $\pm$ 3.2	50.1% $\pm$ 3.3	57.6% $\pm$ 3.2
20000	58.3% $\pm$ 3.2	52.4% $\pm$ 3.3	57.2% $\pm$ 3.2
5000	55.3% $\pm$ 3.3	58.3% $\pm$ 3.2	53.9% $\pm$ 3.3
2500	44.7% $\pm$ 3.2	59.4% $\pm$ 3.2	50.3% $\pm$ 3.2
1000	33.1% $\pm$ 3.1	61.1% $\pm$ 3.2	41.0% $\pm$ 3.2

Table 6.2: Experiment 3 - Detective win rate against number of MCTS iterations

We can compare these results with that of 57.6% $\pm$ 3.2 from previous experiment, when both agents used 10.000 iterations. We can see that detectives reached only a mild improvement with increase of iterations, while hider reached a more significant improvement. With low iterations, hider algorithm seems to perform slightly better than seeker algorithm, which might be due to a lower rate of cooperation with low amount of iterations.

6.4 Experiment 4: Usage of suboptimal moves for Mr.X

We always assume that each agent will take the best action they can. Because of that, not selecting the best move might have a positive effect. We use two variables - percentageDifference and moveChance. If worse action than the best one has its value lower higher than $100\% - percentageDifference$, it has a chance equal to moveChance to use suboptimal move. Default settings are used along with MCTS localisation using 10/90 split.

$Move\ chance$	10%	30%	50%
$Move\ \%\ value$			
90%	59.1% ± 3.2	59.6% ± 3.2	67.2% ± 3.1
70%	64.1% ± 3.1	76.0% ± 2.8	85.9% ± 2.3
50%	71.6% ± 3.0	80.1% ± 2.6	90.5% ± 1.9

Table 6.3: Experiment 4 - Usage of suboptimal moves for Mr.X

Best result of 59.1% ± 3.2 is worse than result of 57.6% ± 3.2 without using suboptimal moves.

6.5 Experiment 5: Usage of double move based on MCTS

Hider uses MCTS to select first action, and then uses MCTS to look at value of possible double move. If value of double move is above threshold, double move is used. This allows for more flexible usage of double move in bad situations, instead of waiting for detectives to come close. Default settings are used along with MCTS localisation using 10/90 split.

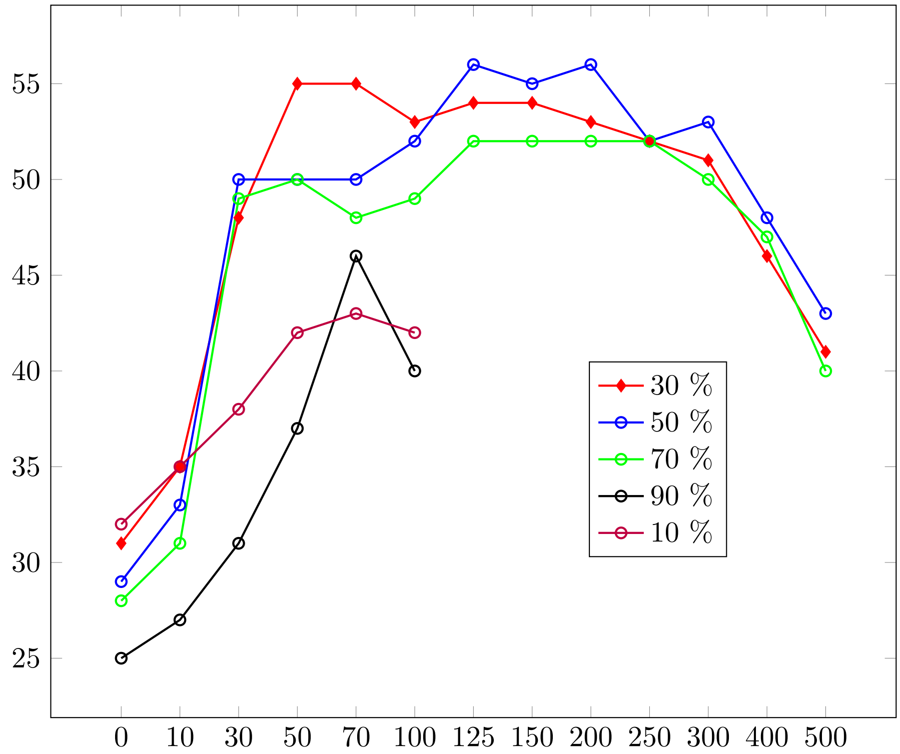


Figure 6.1: Mr.X's win rate for values of win percentage difference between first and second move of double move.

Between values 30% to 300% , hider has a win rate of up to 56 % ± 3.2, which is better win rate than hider's 42.4% ± 3.2 from experiment 2. We can see that this result is fairly consistent

Chapter 7

Conclusions

In this work we focused on improving of heuristics, of which we focused on two - localisation of Mr.X and double move.
We have implemented localisation method based on MCTS, which brought an increase in win rate for detectives from $46.8\% \pm 3.3$ to $59.8\% \pm 3.2$. The main benefit of this method is non-reliance on any new domain knowledge, which let's us use this method in similar games with hidden information. By selecting the most threatening possible position of Mr.X, we can reduce win rate for node with maximum estimated win rate, effectively minimizing maximum win chance. This is very efficient in Scotland Yard, because the number of strong locations for Mr.X is limited. However, this also means seekers will not commit to a certain area, which might be beneficial during end game when there is not enough time to look everywhere.
Mr.X has also gained an improvement. Just like detectives, by using MCTS to decide whether or not to use double move, we improved win rate from $42.4\% \pm 3.2$ to $56\ \% \pm 3.2$. Just like previous method, no new domain knowledge has been introduced. This allows for a big improvement in double move usage - before this improvement, double move usage has been decided by average distance from detectives, which meant that while some detectives could be close by, if there were detectives far away, double move could not be used, which allowed for losses even while double move ticket was still in possession.

Bibliography

[1] Rules of Scotland Yard,

`ravensburger.com/spielanleitungen/ecm/Spielanleitungen/Scotland\_Yard\_W\_And\_B\_GB.pdf`

[2] Ciancarini, Paolo and Favini, Gian, *Monte Carlo tree search in Kriegspiel,* in Artificial Intelligence, vol. 174, pp. 670-684, July 2010

[3] Szita, Istvan and Chaslot, Guillaume and Spronck, Pieter, *Monte-Carlo Tree Search in Settlers of Catan,* in Ethical Theory and Moral Practice, vol. 6048, pp. 21-32, May 2009

[4] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, D. Hassabis, *A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play* in Science, vol. 362, pp. 1140-1144, Dec. 2018

[5] G. Broeck, K. Driessens, Kurt, J. Ramon, *Monte-Carlo Tree Search in Poker Using Expected Reward Distributions,* in Conference Paper, DOI: 10.1007/978-3-642-05224-8-28, Nov. 2009

[6] Kocsis, Levente and Szepesvári, Csaba, *Bandit Based Monte-Carlo Planning,* in Machine Learning: ECML, PP. 282-293, Sep. 2006.

[7] P. Auer, N. Cesa-Bianchi, P. Fischer, *Finite-time Analysis of the Multi-armed Bandit Problem,* in Machine Learning, vol. 47, pp. 235-256, May 2002

[8] Website collecting research on MCTS, `http://mcts.ai`

[9] J. A. M. Nijssen and M. H. M. Winands, *Monte-Carlo Tree Search for the game of Scotland Yard,*. at 2011 IEEE Conference on Computational Intelligence and Games (CIG'11), Seoul, 2011, pp. 158-165.

[10] J. A. M. 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, vol. 4, pp. 282-294, Dec. 2012.

[11] T. Dash, S. Dambekodi, P. Reddy, A. Abraham, *Adversarial neural networks for playing hide-and-search board game Scotland Yard,*. in Neural Computing and Applications, Sep. 2018.

[12] M. Ginsberg, *GIB: Imperfect Information in a Computationally Challenging Game,*. in Journal of Artificial Intelligence Research, vol. 14, June 2011.

[13] P. I. Cowling, D. Whitehouse, E. J. Powley, *Emergent bluffing and inference with Monte Carlo Tree Search,*. at 2015 IEEE Conference on Computational Intelligence and Games (CIG), pp. 114-121

[14] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction,*. MIT Press, Cambridge, MA, USA, 1998. ISBN-13: 978-0262193986

[15] C. B. Browne et al., *A Survey of Monte Carlo Tree Search Methods,* in IEEE Transactions on Computational Intelligence and AI in Games, vol. 4, no. 1, pp. 1-43, March 2012.

[16] G. Chaslot, M. Winands, H. Herik, J. Uiterwijk, B. Bouzy, *Progressive Strategies for Monte-Carlo Tree Search,* in New Mathematics and Natural Computation, vol. 04, pp. 343-357, Nov. 2008

[17] T. Pepels, M. H. M. Winands and M. Lanctot, *Real-Time Monte Carlo Tree Search in Ms Pac-Man,* in IEEE Transactions on Computational Intelligence and AI in Games, vol. 6, no. 3, pp. 245-257, Sept. 2014.

[18] G. Sylvain, D. Silver, *Achieving Master Level Play in 9x9 Computer Go,* inProceedings of the 23rd National Conference on Artificial Intelligence, vol. 3, no. 3, pp. 1537-1540, July 2008.

[19] Andrej Karpathy.Convolutional Neural Networks for Visual Recognition.url:http://cs231n.github.io/.

[20] Course BE4M36MAS, Czech Technical University, Faculty of Electro Engineering, Prague. url:https://cw.fel.cvut.cz/b191/courses/be4m36mas/start

Appendix A

Source code

Source code is based on Java. Java codebase consists of Monte-Carlo Tree Search algorithm and Scotland Yard implementation. Most of this source code is borrowed with some modification. In each source file, author is credited at the top, and new files and new functions created by us have comments about their purpose.