Universiteit Leiden
The Netherlands

Opleiding Informatica

Finding a way through the
changing maze of Labyrinth

Job van der Zwaag

Supervisors:
Walter Kosters & Jeannette de Graaf

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl 27/12/2020

Abstract

The subject of this thesis is the board game Labyrinth. In this board game players need to find treasures in a maze. Players cannot only move through the maze, but they can change the entire layout of the maze by moving the different passages. This way a treasure that the player needs to find can become available or the treasure that the opponent needs to find can become unavailable. The object of this thesis is to create a digital version of the game and build different agents that can play this version. In particular we examine three kind of agents: a random agent, a rule-based agent and agents that use basic Monte Carlo search. These agents are experimented with to find out which agent can play the game the best. Experiments suggest that an agent called the S-RB-AS-MCAgent performs the best. This is an agent that uses Monte Carlo search and rule-based agents to predict the rest of the game after every possible move and change of the board. It also takes a better look at the choices with the highest scores to make sure which one is the best.

Contents

1 Introduction

LABYRINTH, previously THE aMAZEing LABYRINTH, is a board game published by Ravensburger. It was designed by game maker and psychologist Max J. Kobbert in 1986 and was originally called DAS VERRÜCKTE LABYRINTH. The game is known as DOOLHOF in The Netherlands, previously DE BETOVERDE DOOLHOF.

Figure 1: The board game LABYRINTH [Hamleys].

In the game LABYRINTH players are expected to change the maze to their favor. This makes it easier to move towards their objective. But there is an opponent who wants to win as well. So changing the maze to favor yourself is not enough, you need to find a way to block your opponent too. The agents created to play the game need to find their way through the maze by making a path to their objectives, while blocking the paths of their opponent and adapting to the changes to the maze by the other player. These agents will be tested on other board sizes than just the board of the original board game to determine the best one. Experiments suggest that an agent called the S-RB-AS-MCAGENT performs the best on all the boards. This is an agent that uses Monte Carlo search and rule-based agents to predict the rest of the game after every possible move and change of the board. It also takes a better look at the choices with the highest scores to make sure which one is the best.

Thesis overview

The rules of the game LABYRINTH will be explained in Chapter 2. Chapter 3 contains information about related work to this thesis. In Chapter 4 the visualization of the digital version of the game will be explained. The different definitions throughout the thesis are defined in Chapter 5. Chapter 6 will list all the different boards that will be used and Chapter 7 lists all the different agents that will play the game. The results and evaluation of the various experiments are presented in Chapter 8, and Chapter 9 contains the conclusions of these experiments.

This thesis is written as a bachelor thesis supervised by Walter Kosters and Jeannette de Graaf at LIACS, the Leiden Institute of Advanced Computer Science at Leiden University.

2 Rules of the original game

In this chapter the rules of the original LABYRINTH board game [KLL+95] are explained to give an insight in the original game. In a digital version of the game, rules can be changed very easily to test different versions and scenarios. However, the main rules are always the same and all rules are based on the original game. In Chapter 5, the definitions used, are explained more generally.

2.1 Materials and setup

The board game LABYRINTH is played on a square board that fits 49 of the 50 available MAZE CARDs in rows and columns of 7 cards. These MAZE CARDs have various maze passages on them; 16 of the MAZE CARDs are already stuck on the board (see Figure 2). Note also the symmetrical pattern of the MAZE CARDs fixed on the board.

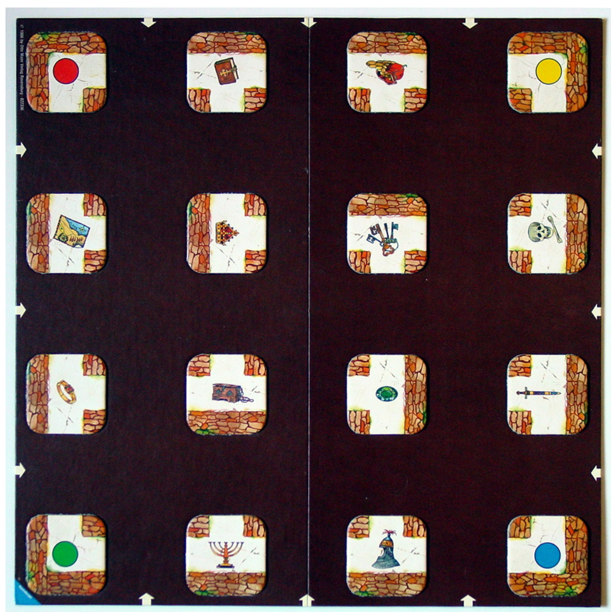


Figure 2: The fixed MAZE CARDs on the board [Magisterrex12a].

By placing the MAZE CARDs in a random order on the board, a maze is formed. The one maze card left is used to change the maze throughout the game, and is referred to as the EXTRA MAZE CARD (see Figure 3).
Out of all the MAZE CARDs, 24 contain special OBJECTs. Each of these OBJECTs is also found on one of the 24 OBJECT CARDs (see Figure 4). At the start of the game these OBJECT CARDs are divided between the players and placed face down in one stack for each player. Each player chooses one of the four game pieces in the colors red, yellow, blue and green. The game pieces are placed on the MAZE CARDs with the same color as the game piece. These MAZE CARDs are located in the

Figure 3: All MAZE CARDs on the board with the EXTRA MAZE CARD [Magisterrex12a, Magisterrex12b].

corners of the game board. This is the starting position for each of the players. The game is played with two, three or four players.

Figure 4: The OBJECT CARDs [Magisterrex12c].

2.2 Goal of the game

For each player, the goal of the game is to move through the maze and find all the OBJECTs on the OBJECT CARDs in front of them. Their stack of OBJECT CARDs is placed face down. Each player can only look at his or her first OBJECT CARD. The OBJECT on that card will become his or her CURRENT GOAL. Because of the placement of the MAZE CARDs, not all OBJECTs are attainable right away. The player can only move on the white open passageways. That is why they need to change the maze using the EXTRA MAZE CARD. The first player that finds all the OBJECTs on his or her OBJECT CARDs and returns to the right starting position is the winner.

2.3 Game play

Every turn consists of two parts: changing the maze and moving one's game piece. Changing the maze is mandatory and this part must always come first. Moving the game piece, however, is not always required.

Changing the maze

The first part of the turn is changing the maze by inserting the EXTRA MAZE CARD into one of the two ends of a movable row or column. This moves every card in this row or column by one position, pushing the last card out of the maze (see Figure 5). This card will be the next EXTRA MAZE CARD, so the next MAZE CARD to be added into the maze by the opponent. In this way all the passageways in this row or column are changed. This allows certain OBJECTs to be directly obtainable for one player and other OBJECTs directly unobtainable for the next player. A player must always use the EXTRA MAZE CARD to change the maze.

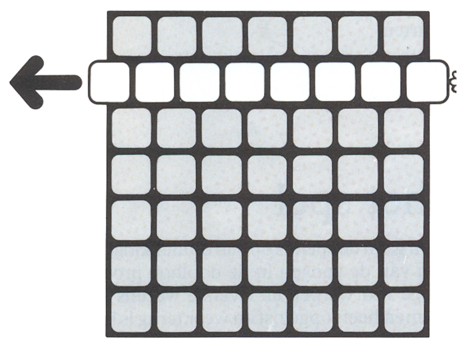


Figure 5: Inserting a MAZE CARD into a row, pushes out the MAZE CARD at the other end [KLL+95].

When a playing piece is located on the MAZE CARD that is being pushed out, it is moved to the MAZE CARD that has just been inserted at the other end (see Figure 6). It is not allowed to insert the MAZE CARD in a way that undoes the last change. This means a player cannot push the EXTRA MAZE CARD back at the same position in the row or column it just came out. Because of the placement of the fixed MAZE CARDs on the board, only every other row and column is able to move, which means one can only insert MAZE CARDs in these rows and columns.

Moving through the maze

The next step is moving the game piece. One is allowed to move everywhere over the open white pathways of the maze. So the game piece cannot move through a wall or on the outside of the game board. There are no rules against two game pieces located on the same MAZE CARD and one can also move through a passage where another game piece is located. When a player ends on the MAZE CARD with the CURRENT GOAL on it, the OBJECT is found and the player can show everyone the

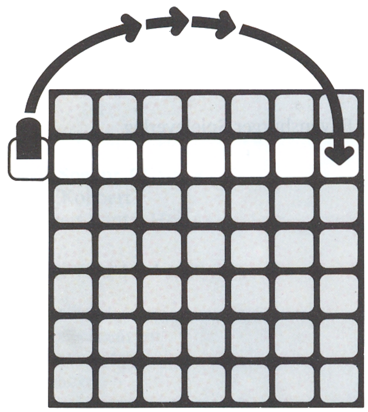


Figure 6: A playing piece located on the MAZE CARD being pushed out, jumps to the MAZE CARD on the other side [KLL+95].

OBJECT CARD and is allowed to look at the next one. A player can only move once on his or her turn, so once the next OBJECT CARD is revealed, the turn is over. When a player has found the last of his or her OBJECTs, the new goal is to return to his or her starting position. The first player to return to his or her starting position after finding all of his of her OBJECTs, is the winner.

3 Related work

In this chapter other work that is similar to the subject of this thesis will be discussed. While there appear to be no other papers about the game LABYRINTH specifically, we can still mention some of the techniques that were used or could have been used.

LABYRINTH is a board game with a changing maze. There are many algorithms for solving non-changing mazes like the Lee Algorithm [Lee61], but the problem in LABYRINTH is not solving the maze. As soon as there exists a path to the point in the maze you are trying to go, you can go there. It does not matter what path you take or how short it is. The problem is creating the path.

An algorithm that could have been used is Monte Carlo Tree Search (MCTS). This is a more advanced version of the Monte Carlo method. The paper "A Survey of Monte Carlo Tree Search Methods" by Browne et al. provides various tools from different publications about MCTS to handle new problems [BPW+12]. While MCTS was not one of the techniques used for the agents that use the Monte Carlo method, it is something that could enhance the current Monte Carlo agents. MCTS has been a successful method for agents playing board games.

An example is AlphaGo, described by by Silver et al. in "Mastering the Game of Go with Deep Neural Networks and Tree Search" [SHM+16]. AlphaGo is the first agent that defeated the human world champion. The agent combines MCTS and a neural network to determine the best moves. A more advanced version of AlphaGo is described in "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm" by Silver et al. is called AlphaZero [SHS+17]. This algorithm is much more generic than its predecessor. In fact it is able to play chess and shogi as well, with only knowing the basic rules of the game, unlike AlphaGo that was trained with games against both other agents and humans. The newest version of this algorithm is MuZero, described by Schrittwieser et al. in “Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model” [SAH+19]. It can match the performance of AlphaZero on the same board games without any knowledge about the game. MuZero can also play Atari games, so playing LABYRINTH should not be a problem.

4 Visualization

To visualize the game, our implementation allows to print the game board to the terminal. Here, every MAZE CARD is represented in a textual way. The terminal can print text in several colors and can also change the background of the text. Clear purple and cyan backgrounds are used to print the walls of the maze. These colors are interchanged to show the different MAZE CARDs more clearly. The pathways of the maze do not have a background, so they are just black.
When a MAZE CARD contains an OBJECT, the letter corresponding to this OBJECT (a, b, c, ...) is printed in the middle of the card. The capital letters R, Y, G and B are used to represent the starting positions of the red, yellow, green and blue player, respectively. See Figure 7 for an example.

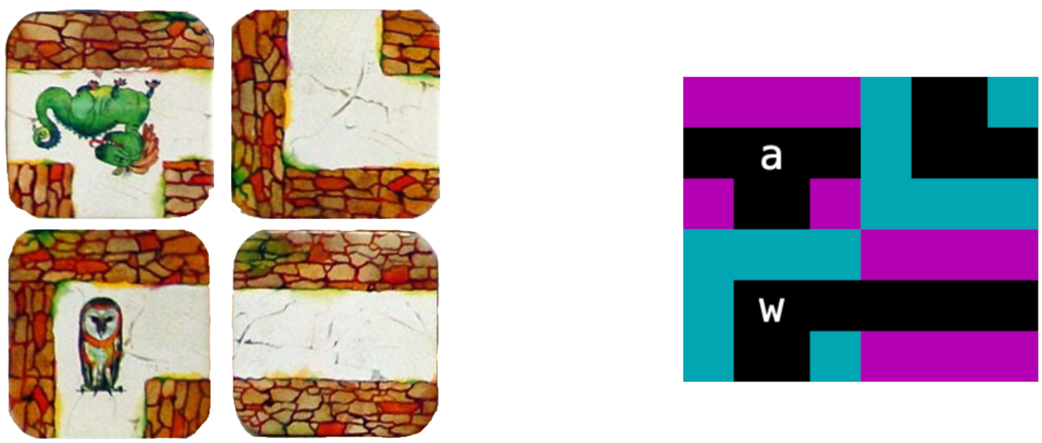


Figure 7: The visualization of some MAZE CARDs [Magisterrex12b].

To show the location of a player’s piece, the background of the middle of the MAZE CARD is changed into the player’s color. See the location of the red and the yellow player in Figure 7 for an example. Both players are located on their starting positions. When there are multiple players on the same MAZE CARD, the background color is changed to white. This means the colors of the individual players are gone, so it is not clear which players are located on this MAZE CARD, but in most cases this can be concluded from the locations of the other players. On the sides of the board the row or column numbers are printed. When there is an arrow next to the number it means this row or column can be moved. Under the board, the extra MAZE CARD is printed in all the different rotations. The visualization of the game board in Figure 3 is shown in Figure 8.

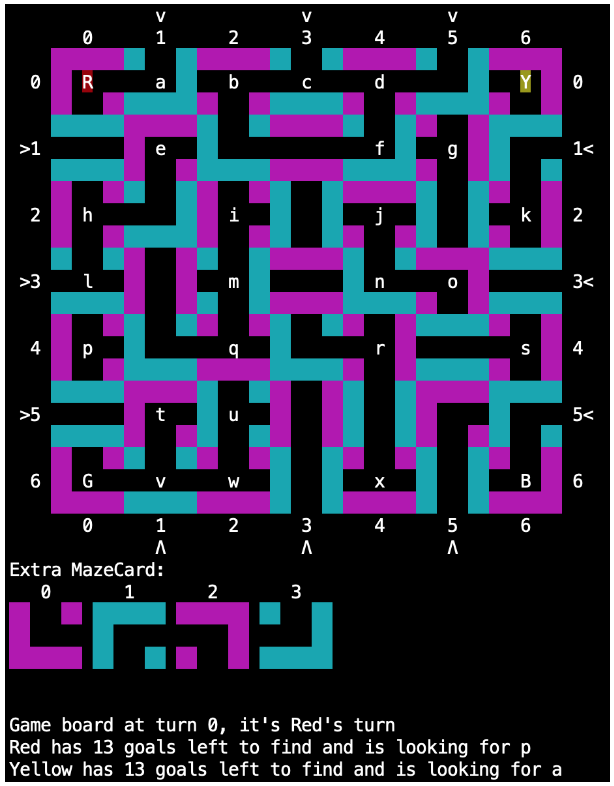


Figure 8: The visualization of the game board.

5 Definitions

The board game LABYRINTH consists of a game board with multiple MAZE CARDs that make up the maze. This maze contains multiple OBJECTs for the players to find. A player can only obtain an OBJECT, when the player has the corresponding OBJECT CARD face up. In this section all these definitions are explained.

5.1 The game board

The game board consists of MAZE CARDs. These square MAZE CARDs contain a part of the maze and have passages on some of the sides of the cards. When two MAZE CARDs are next to each other and both have a passage on the sides that touch each other, the player is allowed to move between these two cards (see Figure 9). During the game the maze is changed by moving the MAZE CARDs on the board, but some rows and columns are fixed.
When setting up the board with the MAZE CARDs, there should be one MAZE CARD left. This MAZE CARD is called the EXTRA MAZE CARD and is used to change the board in the next turn.

5.2 Maze cards

The game consists of three different types of MAZE CARDs:

Figure 9: Left: No passage between the maze cards. Right: A pathway between the maze cards.

- The L-type maze card, with two open passages on connected sides. See Figure 10 (left).
- The T-type maze card, with three open passages. See Figure 10 (middle).
- The I-type maze card, with two open passages on opposed sides. See Figure 10 (right).

Figure 10: From left to right: the L-type maze cards, the T-type maze card and the I-type maze card.

Rotations of the L-type maze card

All the different rotations of the maze card have distinct names. The L-type maze card has four different rotations (see Figure 11):

- Rotation 0, with an open passage on the top and right side of the maze card.
- Rotation 1, with an open passage on the right side and the bottom of the maze card.
- Rotation 2, with an open passage on the bottom and the left side of the maze card.
- Rotation 3, with an open passage on the left side and the top of the maze card.

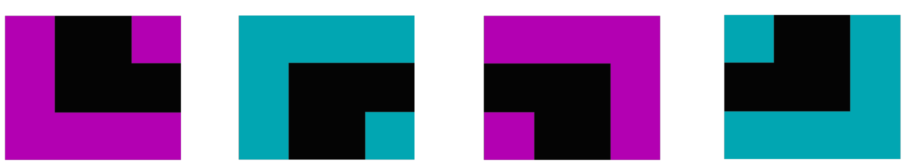


Figure 11: From left to right: the L-type maze card in Rotation 0, the L-type maze card in Rotation 1, the L-type maze card in Rotation 2 and the L-type maze card in Rotation 3.

Rotations of the I-type maze card

The I-type maze card has two different rotations (see Figure 12):

- Rotation 0, with an open passage on the top and bottom of the maze card.
- Rotation 1, with an open passage on the left and right side of the maze card.

Figure 12: From left to right: the I-type maze card in Rotation 0 and the I-type maze card in Rotation 1.

Rotations of the T-type maze card

The T-type maze card has four different rotations (see Figure 13):

- Rotation 0, with an open passage on all sides of the maze card except for the top.
- Rotation 1, with an open passage on all sides of the maze card except for the right side.
- Rotation 2, with an open passage on all sides of the maze card except for the bottom.
- Rotation 3, with an open passage on all sides of the maze card except for the left side.

Figure 13: From left to right: the T-type maze card in Rotation 0, the T-type maze card in Rotation 1, the T-type maze card in Rotation 2 and the T-type maze card in Rotation 3.

5.3 Objects

Some maze cards have objects on them. These objects are connected to special object cards. For every object in the maze there exists an object card, and the other way around. The players must find these objects on the board. The object cards are equally divided between the players. Players can only see one object card at a time. The object on this card is the object they must find in the maze, their current goal.

6 Different boards

To make the testing more interesting, there are different boards that can be created. This way the players can be tested on different boards. In this section all these different boards are explained.

6.1 Original board

The so-called original board is identical to the board of the real life board game. It consists of 49 maze cards in 7 rows and 7 columns and one extra maze card. Every other column and row is not able to move starting with unmovable rows and columns at the ends.

The fixed maze cards

The fixed pieces are from left-to-right, then top-to-bottom:

1. one L-type in rotation 1
2. two T-types in rotation 0
3. one L-type in rotation 2
4. two T-types in rotation 3
5. one T-type in rotation 0
6. one T-type in rotation 1
7. one T-type in rotation 3
8. one T-type in rotation 2
9. two T-types in rotation 1
10. one L-type in rotation 0
11. two T-types in rotation 2
12. one L-type in rotation 3

The four corner pieces all contain a starting point for a player piece. The other fixed maze cards all contain an object.

The other maze cards

The rest of the board is filled with the other cards in a random order. The one maze card that is left, is the extra maze card. The available cards are:

- 6 T-type maze cards with an object
- 6 L-type maze cards with an object

- 9 L-TYPE MAZE CARDs without an OBJECT
- 13 I-TYPE MAZE CARDs without an OBJECT

In practice, every one of these cards is given a number from 0 to 33. These numbers are placed inside a vector that is then shuffled. When a MAZE CARD is placed, the next number is pulled from the vector. This determines the type of the MAZE CARD and the presence of an OBJECT. All cards are placed on the board in a random rotation. All possible rotations are equally likely. See Figure 3 for an example of an ORIGINAL BOARD.

6.2 Random board

The so-called RANDOM BOARD has the same size as the ORIGINAL BOARD and the same rows and columns can and cannot move. The difference is the layout of the MAZE CARDs on the board. All MAZE CARDs on the board have a random type and rotation and the OBJECTs and starting points are placed on the MAZE CARDs randomly too. For the location of the OBJECTS and the starting points, the numbers 0 to 49 are placed in a vector that is then shuffled.
For every position of the game board, a MAZE CARD is created. Every MAZE CARD (even the ones that cannot move), has an equal probability to either be a T-TYPE, L-TYPE or I-TYPE MAZE CARD. All three types are equally likely. The rotation is random for all MAZE CARDs as well and is determined in the same way. Once a MAZE CARD is created, the next number is pulled from the vector. The first 25 numbers correspond to the 25 OBJECTs, the next 4 numbers to the 4 starting points and the other numbers to an empty MAZE CARD. An assigned OBJECT or starting point is placed upon the created MAZE CARD.
See Figure 14 for an example of a RANDOM BOARD. Notice that the starting positions of the red and yellow player are not located on the corner pieces, but on very different places.

6.3 3×3 board

The 3×3 BOARD is a board with three rows and three columns where all rows and columns are able to move. The corner pieces are the same as for the ORIGINAL BOARD, but the rest of the board is filled as a RANDOM BOARD. The board contains 4 OBJECTs. See Figure 15 (Right) for an example of a 3×3 BOARD.

6.4 5×5 board

The 5×5 BOARD is a board with five rows and five columns where only the second and fourth row and column are able to move. Just like the 3×3 BOARD, the corner pieces are the same as for the ORIGINAL BOARD, but the rest of the board is filled as a RANDOM BOARD. The board contains 12 OBJECTs. See Figure 15 (Left) for an example of a 5×5 BOARD.

6.5 Input board

To test agents on very specific boards, there exists a board that is completely based on text file input, the so-called INPUT BOARD. The first row of the file contains the number of rows and the

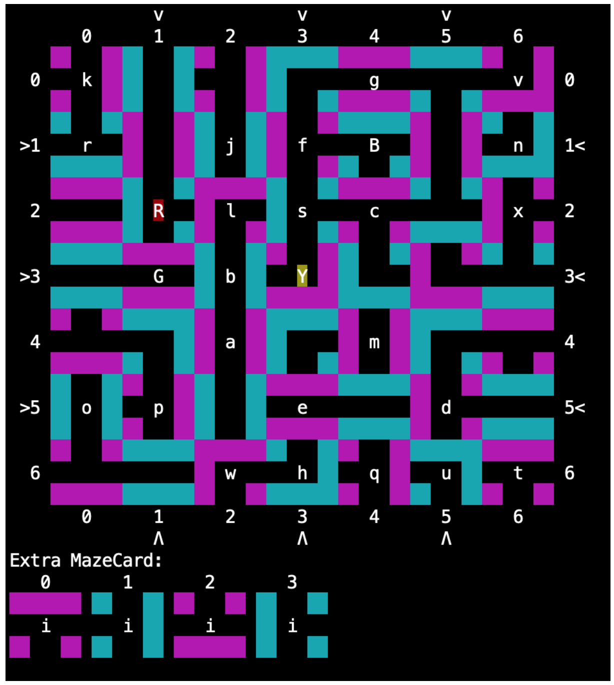


Figure 14: An example of the random board.

second row the number of columns of the board. The next rows contain information about the maze cards of every row with a space between every maze card.
For every maze card there must be three characters. The first character corresponds to the type and is either a L, T or I. The second character corresponds to the rotation and is a number between 0 and 3. The last character is either a lower case letter, to signify an object, or either the letter R, Y, G, B or A, to signify the red player's starting point, yellow player's starting point, green player's starting point, blue player's starting point or an empty maze card, respectively. Since there are 26 different lower case letters, there is a maximum of 26 objects.
The last maze card description is the extra maze card. The second to last row contains the row numbers of the rows that are able to move with spaces in between. The last row has the same information about the movable columns.
Of course, the text file input needs to conform to the rules of the game. For example, there cannot be a player without a starting position, there needs to be at least one object for every player to find and at least one row or column needs to be able to move.
An example of an input board text file is:

```
3
6
L1a L3b T0A I1A T3c T2A
I0R I0A I0d I0e I0A I0Y
T3A I0f T3g T3h I0i T3A
L3A
1 2
0 2 3 5
```

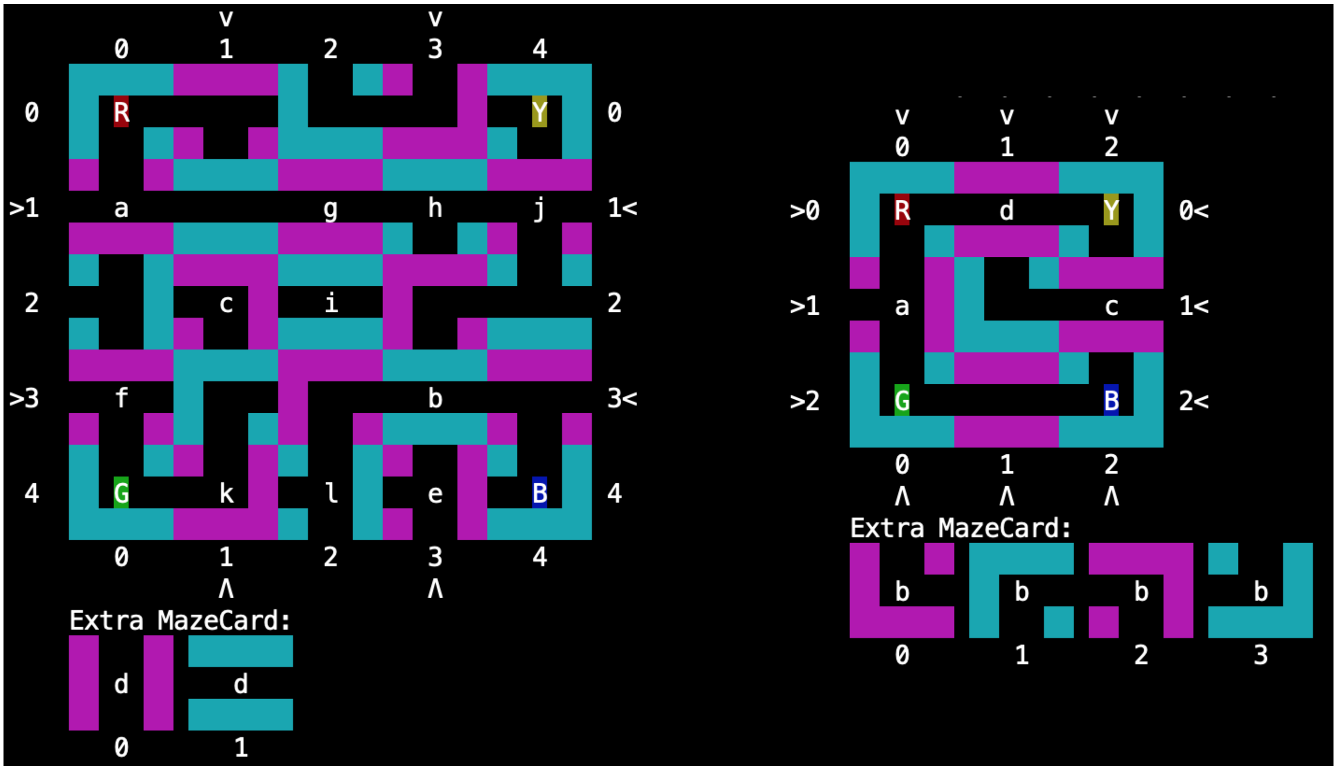


Figure 15: Left: An example of the 5×5 BOARD. Right: An example of the 3×3 BOARD.

See Figure 16 for the corresponding INPUT BOARD.

7 Agents

Three kind of agents are created to play the digital version of the game. These agents all have different characteristics that allow them to use more or less information about the game. The agent that uses the least information is the RANDOMAGENT. The agent that uses the most information about the game is the RULEBASEDAGENT. In the middle are the various MCAGENTs. There are different versions of the MCAGENT that are parameterized to highlight their differences.

7.1 RANDOMAGENT

The first agent will look at every option in the game and chooses one of these options randomly. We will refer to this agent as the RANDOMAGENT. This agent will first choose an insertion of the EXTRA MAZE CARD randomly. So both the rotation, the location of the insertion. Then the RANDOMAGENT will choose a move location for the game piece randomly.

Insertion of the EXTRA MAZE CARD

In general the number of possible insertions is equal to the number of possible insert locations times the number of possible rotations. The number of possible rotations is either four (for a L-TYPE or

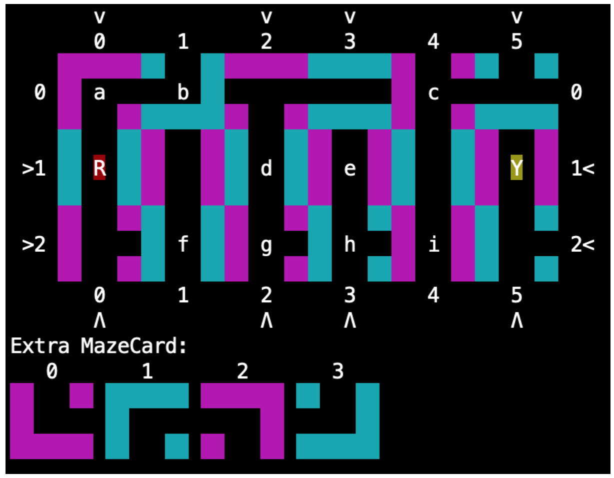


Figure 16: An example of the input board.

T-type maze card) or two (for an I-type maze card). The number of possible insertions is equal to twice the number of moveable rows plus twice the number of moveable columns. So the number of possible insertions for a L-type or T-type maze card would be equal to

$$8 * (r_m + c_m)$$

with r_m the number of moveable rows and c_m the number of moveable columns. The number of possible insertions for an I-type maze card is equal to

$$4 * (r_m + c_m)$$

with r_m the number of moveable rows and c_m the number of moveable columns. But for both maze cards one of these possibilities is not allowed because it would place the maze card back in the same position it just came out of, with the exception of the first move. So the chosen insertion is not completely random. Therefore, the actual number of possible insertions is

$$t * (r_m + c_m) - b$$

with $t = 8$ for a L-type or T-type maze card, $t = 4$ for an I-type maze card, r_m the number of moveable rows, c_m the number of moveable columns and $b = 0$ for the first move of the game and $b = 1$ for every other move. All allowed maze card insertions have an equal probability for the RandomAgent. Every one of these possible insertions is given a number between 0 and

$$t * (r_m + c_m) - b - 1$$

and a random number is generated, corresponding to one of the insertions. This way all possible insertions have an equal chance to be chosen, a chance of 1 in $t * (r_m + c_m) - b$.

Move location of the game piece

After inserting the maze card, the RandomAgent looks at all the reachable maze cards. If one of these maze cards contains the current goal, the agent will move to this maze card.

Otherwise the RandomAgent will choose a random reachable maze card to move to, with all locations being equally likely. All reachable maze cards (including the current location) are assigned a number between 0 and “the number of reachable maze card”-1. Then a random number is generated, corresponding to one of the reachable maze cards.
So the selected move is not always selected at random, which makes the RandomAgent not completely random. When the RandomAgent is not able to move to the current goal the chance of landing on one of the reachable maze cards is always equal to 1 divided by “the number of reachable maze cards”.

7.2 RuleBasedAgent

The next agent is referred to as the RuleBasedAgent and uses more information about the game than the RandomAgent. This agent chooses its insertions and movements based on a few more advanced game tactics.

Insertion of the extra maze card

Every turn, the RuleBasedAgent looks at every possible insertion and checks all the reachable maze cards after this insertion. All these maze cards are checked for the current goal. If there exists an insertion that makes it possible to obtain the current goal immediately, the RuleBasedAgent will make this insertion. If there are several insertions that make it possible to obtain the current goal immediately, the agent will choose the first one that it comes across. If there is no such insertion possible, the RuleBasedAgent will make a random insertion in the same way as the RandomAgent.

Move location of the game piece

After insertion, the RuleBasedAgent will move to the current goal, if this is possible. If it is not, the agent will move to a maze card that is close to the object. This means the RuleBasedAgent will check the distance between the current goal and each reachable maze card. The distance between two maze cards is defined as the sum of the vertical difference and the horizontal difference. The agent will move to the maze card with the smallest distance to the current goal. When there is more than one reachable maze card with the same shortest distance to the current goal, the RuleBasedAgent will choose the maze card it comes across first.

7.3 MCAgent

The Monte Carlo agents, or MCAgents, use for every turn a Monte Carlo method. The Monte Carlo method is a way to determine which possible action has the best chance of being a good one. Every time the MCAgent is triggered, it will examine all the possible options for this turn and will predict what would happen if it would choose this option. For all MCAgents, a number of games is played for each of the options for the initial turn, the so-called number of play-outs. These play-outs are played randomly and the option with the most wins at the end out of all the play-outs will be chosen in the actual turn. In case of a tie, the MCAgent will choose the turn it comes across first. Playing the entire game can be too long, so instead a certain depth of turns is set. After this number of turns is reached in a playout, the current state of the game will be given a score to reflect the MCAgent's chance of winning this game and the option with the highest total score out of all the play-outs will be chosen in the real turn. In case of a tie, the MCAgent will choose the turn it comes across first.

Different MCAgents

There are eight different versions of the MCAgents, that differ in three attributes:

- First every MCAgent is either the Single version, and sees the entire turn as one, or the Double version, and sees the turn as two smaller turns.
- For all MCAgents there also exists a special RuleBased version, that uses the RuleBasedAgent to predict the rest of the game, instead of the RandomAgent.
- Lastly there is also an AllStar version of all agents that uses a Monte Carlo algorithm to take a closer look at the best choices already determined using a Monte Carlo algorithm.

So in conclusion the eight different MCAgents are:

- S-MCAgent, the Single version of the MCAgent.
- D-MCAgent, the Double version of the MCAgent.
- S-RB-MCAgent, the Single RuleBased version of the MCAgent.
- D-RB-MCAgent, the Double RuleBased version of the MCAgent.
- S-AS-MCAgent, the Single AllStar version of the MCAgent.
- D-AS-MCAgent, the Double AllStar version of the MCAgent.
- S-RB-AS-MCAgent, the Single RuleBased AllStar version of the MCAgent.
- D-RB-AS-MCAgent, the Double RuleBased AllStar version of the MCAgent.

Single version

The Single version of the MCAgent combines the two parts of the turn, the insertion of the extra maze card and the movement of the game piece, into one Monte Carlo algorithm. For every possible combination of the insertion of the extra maze card and movement after this insertion, the number of play-outs is set to be 100. For these play-outs, the agents are all replaced by RandomAgents. The unknown objects to find are replaced by a random selection of all the objects yet to be found. The current goal of the MCAgent is known, so this object is not replaced. All the other objects that are yet to be found, are divided in the same manner as in the creation of a new game. The depth is set to be four turns. After these turns, the number of objects found by each of the agents is used to create a score for every combination of insertion and movement. This score is based on the number of wins, the number of objects found by the MCAgent and the number of objects found by the other agents. A won game receives 50 points and a lost game receives -50 points. For every object found the score is increased by 15 points and for every object found by the opponent the score is reduced by 10 points. This scoring method is determined by small experiments of 25 games with various scoring methods. The best method was selected by comparing the number of wins, the number of objects found by the agent and the number of objects found by the opponent. The depth of four is determined in the same way. The combination of insertion and movement with the highest score is used in the real game. When there is more than one combination of insertion and movement with the same highest score, the MCAgent will choose the combination of insertion and movement it comes across first.

Double version

The Double version of the MCAgent uses two Monte Carlo algorithms for the two parts of the turn. For every possible insertion of the extra maze card the number of play-outs is set to be 100. Analogous to the Single version, the agents in these play-outs are all replaced by RandomAgents and the unknown objects to find are replaced by a random selection of all the objects yet to be found. These play-outs are all set to be five turns to see how many objects each of the agents can obtain. This information and the number of wins is used to create a score. Wining the game means the score increases by 50 points, while losing the game means the agent loses 50 points. For every object the agent finds, the score increases by 10 points. For every object the opponent finds, the score decreases by 10 points. The score method and depth of the play-outs are determined in the same way as for the Single version. The insertion with the highest score is used in the real game. When there is more than one insertion with the same highest score, the MCAgent will choose the insertion it comes across first.
After this insertion a 100 play-outs are played for every possible movement of the agent. These play-outs have the same replacements of agents and objects to find as the play-outs for the insertion. These play-outs are set to be five turns to check the number of wins and the number of objects obtained by every agent to calculate a score as well. This score is calculated in the same way as with the selection of the insertion. The MCAgent uses the move with the highest score as the movement in the real game. When there is more than one movement with the same highest score, the MCAgent will choose the movement it comes across first.

AllStar version

The AllStar version of the MCAgent is a special version of the normal MCAgent. In the case of the Single version, there are still play-outs created for every combination of insertion of the extra maze card and movement after this insertion. In this case the number of play-outs equals 10. The play-outs are analogous and the objects obtained by the agents are used to create a score. Then the 10 combinations of insertion and movement with the highest scores are played out 100 times more. The combination with the highest score from these play-outs is used in the real game. In the case of the Double version, the choice of the insertion of the extra maze card is once again separated from the choice of the movement. With the AllStar version, both create 10 play-outs for every choice initially. The choices with the highest scores will be played out 100 times more in the same way.

RuleBased version

The RuleBased version of the MCAgent replaces all the agents by RuleBasedAgents instead of RandomAgents in all the play-outs. This way, all the agents in the play-outs will use more information about the current situation of the game, which will make the agents more like human agents. Since the play-outs are more like the real game, the RuleBased version should be better in predicting the rest of the game. In a game against RandomAgents this might work against the RuleBased version, since the RuleBased version makes predictions based on the RuleBasedAgent while it is playing against a RandomAgent.

8 Results and Evaluation

In the experiments, the agents are compared on three different boards: the 3×3 board, the 5×5 board and the original board. All experiments are conducted in one on one games against the same player. In this way all agents have an equal chance. The agent being tested always starts at the red marker and the opponent at the blue marker. This places the agents the furthest from each other. Each agent plays 100 games on every board. The agents are compared with respect to the total number of wins, the average number of objects found and the average number of rounds played. For all games the game board is reset. So all the maze cards that are placed upon the board randomly are changed between every game.

The most basic agents, the RandomAgent, the RuleBasedAgent, the S-MCAgent and the D-MCAgent, are first tested in games against the RandomAgent. The best players get to continue in games against the RuleBasedAgent.

As seen from the results later in this section, the S-MCAgent outperforms the other agents against the RandomAgent. So to test the S-MCAgent further, the rest of the experiments, played against the RuleBasedAgent, are between the D-MCAgent, the S-MCAgent, the S-RB-MCAgent, the S-AS-MCAgent and the S-RB-AS-MCAgent.

8.1 Time

Since all agents are very different, some agents take more time to complete a game than others. Tables 1 and 2 contain the average running times of one game in the experiments run.

	3×3 board	5×5 board	original board
RandomAgent	<1 ms	4 ms	18 ms
RuleBasedAgent	5 ms	87 ms	270 ms
D-MCAgent	2 s	10 s	39 s
S-MCAgent	2 s	17 s	2 m

Table 1: The average time of one game against the RandomAgent.

8.2 Number of wins

In this experiment the agents play 100 games on all three boards and the total number of wins per board is tracked.

	3 × 3 BOARD	5 × 5 BOARD	ORIGINAL BOARD
D-MCAgent	2 s	-	-
S-MCAgent	3 s	19 s	2 m 29s
S-RB-MCAgent	-	-	1 u 28 m
S-AS-MCAgent	127 ms	5 s	41 s
S-RB-AS-MCAgent	18 s	2 m 23 s	20 m 27 s

Table 2: The average time of one game against the RuleBasedAgent.

Against the RandomAgent

The results of the 100 games against the RandomAgent can be found in Table 3.

	3 × 3 BOARD	5 × 5 BOARD	ORIGINAL BOARD
RandomAgent	49	51	47
RuleBasedAgent	85	82	100
D-MCAgent	85	96	100
S-MCAgent	97	100	100

Table 3: The number of wins against the RandomAgent out of 100 games.

As seen from the results, the RandomAgent wins about half of the games against itself on all three boards, which is expected from two agents that are the same. The RuleBasedAgent and the D-MCAgent perform best on the original board. They both win all 100 games. Given the size of this board, the chance that the RandomAgent obtains enough objects to win the game by luck is very small. This is probably the reason for this big loss by the RandomAgent. The RuleBasedAgent seems to perform around the same on the biggest two boards, while the D-MCAgent performs a bit worse on the 3 × 3 board. Since this board is so small and there are few objects to be found, a RandomAgent can win much easier here, so this would explain the worse result for the D-MCAgent, but not the same result for the RuleBasedAgent.

Against the RuleBasedAgent

The rest of the experiments are 100 games against the RuleBasedAgent. The results of this experiment can be found in Table 4.

	3 × 3 BOARD	5 × 5 BOARD	ORIGINAL BOARD
D-MCAgent	56	72	78
S-MCAgent	84	94	99
S-RB-MCAgent	58	96	100
S-AS-MCAgent	89	96	99
S-RB-AS-MCAgent	94	97	99

Table 4: The number of wins against the RuleBasedAgent out of 100 games.

In these games against the RuleBasedAgent all tested agents win more games on a larger board. So it seems that the tested agents have more of an advantage against the RuleBasedAgent on a larger board. The D-MCAgent performs the worst on all boards. The best agent is harder to determine. On the 5×5 board and the original board all other agents are extremely close to each other, but on the 3×3 board the S-RB-MCAgent performs about as bad as the D-MCAgent, while the S-RB-AS-MCAgent performs much better than the rest of the agents.

8.3 Average number of objects found

In the same games where the number of wins was tracked, the average number of objects found was stored as well. This is the number of objects the agent found in a game on average in these 100 games.
Notice that the number of objects equals 4 on a 3×3 board, 12 on a 5×5 board and 24 on an original board. So every player has 2 objects to find on a 3×3 board, 6 on a 5×5 board and 12 on an original board. However, the last objective of going back to the starting position is seen as a object as well. So the maximum number of objects that can be found in one game equals 3 on a 3×3 board, 7 on a 5×5 board and 13 on an original board.
The winner of the games always finds the maximum number of objects, so the purpose of this experiment is finding out the difference in objects found in lost games.

Against the RandomAgent

Table 5 contains the average number of objects found for the 100 games against the RandomAgent.

	3×3 board		5×5 board		original board	
RandomAgent	2.1	70.0%	5.6	80.0%	11.1	85.4%
RuleBasedAgent	2.7	90.0%	6.4	91.4%	13.0	100%
D-MCAgent	2.9	96.7%	7.0	100%	13.0	100%
S-MCAgent	3.0	100%	7.0	100%	13.0	100%

Table 5: The average number of objects found out of 100 games against the RandomAgent.

Since the RuleBasedAgent and the D-MCAgent won all 100 games on the original board, it makes sense that both have an average of 13 objects found. On the 3×3 board, the objects are easier to find, so it is no surprise that all agents find a high number of objects on average. The RandomAgent finds the fewest objects on all boards, while the other agents have very high averages. So while some games might have been lost, the agents managed to find a number of objects that is close to the number needed to win. This makes sense against a RandomAgent that probably won by accident against the RuleBasedAgent and the D-MCAgent.

Against the RuleBasedAgent

Now we look at the games against the RuleBasedAgent, for which the results are in Table 6. In this metric, as expected, the D-MCAgent still performs the worst. But its number of objects found is much closer to the other agents. All agents still have higher scores on the larger boards. On the largest two boards all agents manage to find almost all objects, with the D-MCAgent finding fewer objects on the 5×5 board. On the smallest board, the differences are somewhat

	3×3 BOARD		5×5 BOARD		ORIGINAL BOARD	
D-MCAgent	2.5	83.3%	6.7	95.7%	12.8	98.2%
S-MCAgent	2.8	94.3%	6.9	98.7%	13.0	99.9%
S-RB-MCAgent	2.5	84.7%	6.9	99.0%	13.0	100%
S-AS-MCAgent	2.9	95.0%	6.9	98.7%	13.0	99.9%
S-RB-AS-MCAgent	2.9	97.7%	7.0	99.6%	13.0	99.9%

Table 6: The average number of OBJECTs found out of 100 games against the RuleBasedAgent.

larger, with once again the D-MCAgent and the S-RB-MCAgent at the bottom and the S-RB-AS-MCAgent at the top.

8.4 Average number of OBJECTs found by the opponent

Just like the average number of OBJECTs found by the agent, the average number of OBJECTs found by the opponent was also tracked. The opponent needs to find the same number of OBJECTs to win a game.

Against the RandomAgent

The average number of OBJECTs found by the opponent in the games against the RandomAgent can be found in Table 7. While every agent plays against the same agent, all these RandomAgents find a different number of OBJECTs because they are playing against different agents.

	3×3 BOARD		5×5 BOARD		ORIGINAL BOARD	
RandomAgent	2.0	66.7%	5.4	77.1%	11.0	84.6%
RuleBasedAgent	1.1	36.7%	3.5	50.0%	3.8	29.2%
D-MCAgent	1.1	36.7%	1.9	27.1%	2.5	19.2%
S-MCAgent	0.8	26.7%	1.1	15.7%	1.4	10.8%

Table 7: The average number of OBJECTs found by the RandomAgent out of 100 games against different agents.

Notice that the in the games of the RandomAgent against the RandomAgent, the two RandomAgent found about the same number of OBJECTs. This makes sense since it is the same agent. The number here is, however, a little bit smaller on all three boards, but that could be because the agent being tested always gets the first turn.
Against the MCAgents the percentage of OBJECTs found decreases for a larger board size, while this does not seem to be the case against the other players.
Also notice that on the 3×3 BOARD the opponent of the RuleBasedAgent and the D-MCAgent find the same number of OBJECTs and the RuleBasedAgent and the D-MCAgent win the same number of games (see Table 3), while the D-MCAgent manages to find a significant number of OBJECTs more than the RuleBasedAgent (see Table 5).

Against the RuleBasedAgent

Table 8 contains the results for the same experiments against the RuleBasedAgent. Like the games against the RandomAgent, the RuleBasedAgents in these experiments find different numbers of objects because they are playing against different agents.

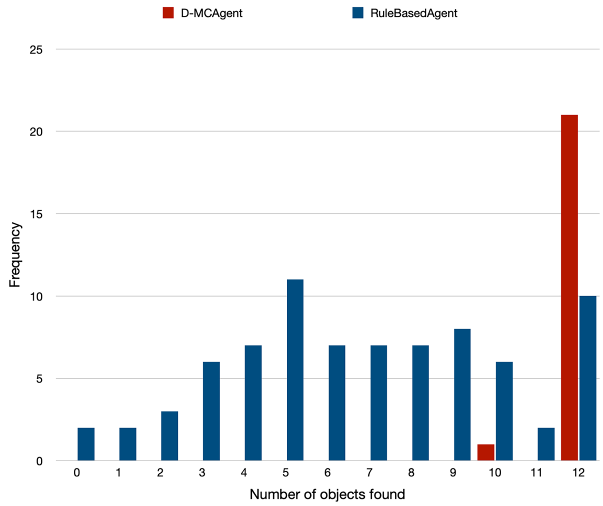
	3×3 board		5×5 board		original board	
D-MCAgent	2.1	68.3%	4.2	59.6%	8.2	62.8%
S-MCAgent	1.3	44.7%	2.8	40.6%	5.5	42.4%
S-RB-MCAgent	1.9	63.0%	2.3	32.9%	4.6	35.0%
S-AS-MCAgent	1.4	45.3%	2.6	37.3%	6.0	46.0%
S-RB-AS-MCAgent	1.1	37.7%	2.7	37.9%	5.0	38.4%

Table 8: The average number of objects found by the RuleBasedAgent out of 100 games against different agents.

While with the other two metrics all agents were pretty close together, in this metric the differences are much bigger. The opponents of the tested agents do not follow the previous metrics of scoring in order of the board sizes. Most agents actually have their opponent perform the worst on the 5×5 board. The opponent of the D-MCAgent manages to find the most objects with a large margin on all three boards. On the 3×3 board the opponent S-RB-MCAgent finds not much less, while the opponent of this agent manages to find the fewest objects on the other two boards. Very close on these board is the opponent of the S-RB-AS-MCAgent which also finds the fewest objects on the smallest board.

8.5 Distribution of objects found

The average number of objects found can be misleading if an agent, for example, either finds the maximum or nothing. In that case the average will be somewhere in the middle whereas that number of objects is never actually found. That is why the distribution of the number of found objects will also be reported on. The following graphs show the distribution of the number of objects found if the game is lost for the 100 games of every agent from the experiments against the RuleBasedAgent on the original board. See Figure 17 for the distribution of the number of objects found by the D-MCAgent and the S-MCAgent, Figure 18 for the distribution of the number of objects found by the S-AS-MCAgent and the S-RB-MCAgent and Figure 19 for the distribution of the number of objects found by the S-RB-AS-MCAgent. Notice that the results only show 0 to 12 objects found on the horizontal axes, since the game would have been won if the agent found 13 objects and we only look at lost games at this point.
The distributions of the RandomAgent against the S-MCAgent and the S-AS-MCAgent are symmetric and unimodal with a peak around the middle. The distributions of the RandomAgent against the D-MCAgent is symmetric and unimodal for the most part as well, but with an extra peak at 12 objects found. The distribution of the RandomAgent against the S-RB-MCAgent is skewed right, but has a lower frequency at 0 and 2 objects found. As expected from Subsection 8.3 most games lost by the agents being tested are lost with 12 objects found.



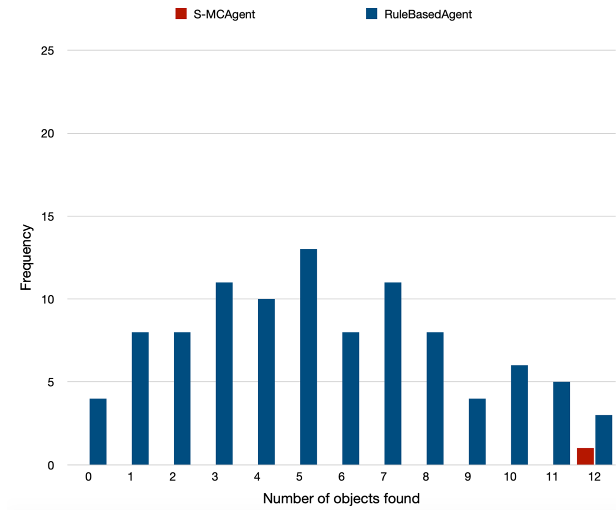


Figure 17: The distribution of OBJECTs found on a lost game against the RULEBASEDAGENT out of 100 initial games on the ORIGINAL BOARD. Left: the D-MCAGENT. Right: the S-MCAGENT.

8.6 Average number of rounds

To determine how long a game was, the number of rounds in a game was tracked. In one round, both players get one turn. There is a specification on the average number of rounds on a game that was won, to look at the difference in length between a lost game and won game.

Against the RANDOMAGENT

Notice that in the 100 games on the ORIGINAL BOARD for the RULEBASEDAGENT, D-MCAGENT and the S-MCAGENT, no games were lost. So there is no information on the number of rounds on lost games for these players. The same is true for the S-MCAGENT on the 5×5 BOARD. Table 9 contains the average number of rounds in the other games against the RANDOMAGENT.

	3×3 BOARD			5×5 BOARD			ORIGINAL BOARD		
	Total	**Won**	**Lost**	**Total**	**Won**	**Lost**	**Total**	**Won**	**Lost**
RANDOMAGENT	22.0	20.2	23.7	146.4	153.7	138.7	366.0	359.7	371.5
RULEBASEDAGENT	12.7	12.3	15.1	78.0	71.1	109.3	108.5	108.5	-
D-MCAGENT	13.5	11.7	23.3	52.8	51.3	88.5	88.4	88.4	-
S-MCAGENT	6.4	6.3	11.7	25.0	25.0	-	41.3	41.3	-

Table 9: The average number of rounds out of a 100 games against the RANDOMAGENT on the total 100 games, the won games and the lost games.

In general it seems that the larger the board, the higher the number of rounds. This makes sense since a larger board contains more OBJECTs to find and has more distance between OBJECTs. So it takes more rounds to get enough OBJECTs to win the game. On all boards the S-MCAGENT seems to need the smallest number of rounds. The RANDOMAGENT needs the most rounds, but this is logical for an agent that plays against itself, especially for one that uses random moves.

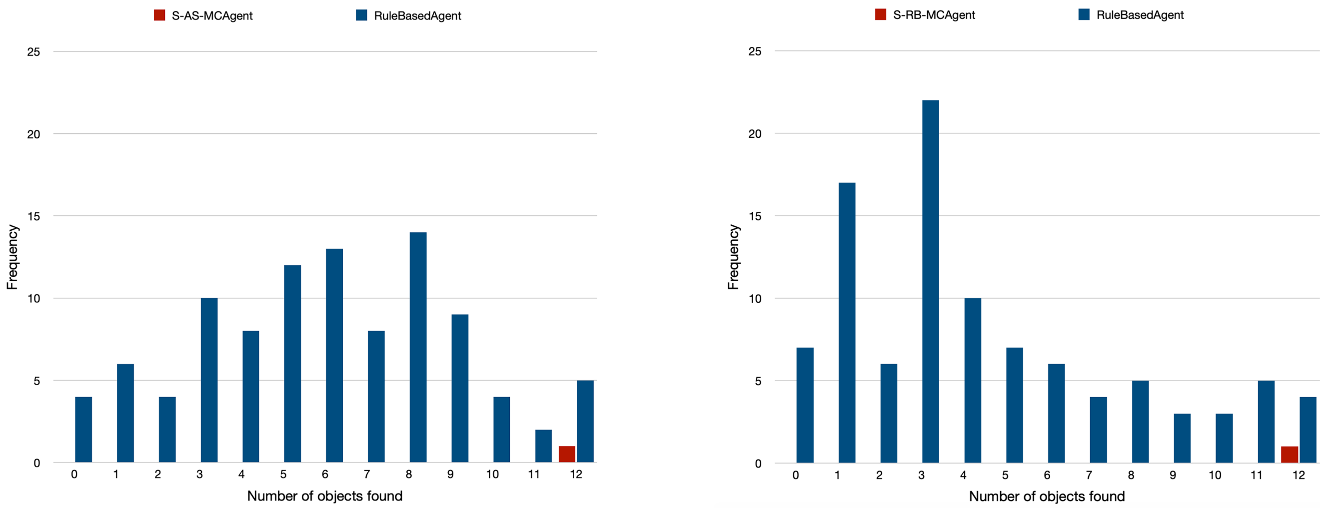


Figure 18: The distribution of OBJECTs found on a lost game against the RULEBASEDAGENT out of 100 initial games on the ORIGINAL BOARD. Left: the S-AS-MCAGENT. Right: the S-RB-MCAGENT.

In most cases the average number of rounds on a won game is smaller than the average number on all 100 games. So it takes fewer rounds if the game is won. This is probably because most agents find many objects in all games. So when a game is lost it was probably close and the game took longer.

It is hard to compare the agents on the lost games for the ORIGINAL BOARD. The S-MCAGENT, with the fewest total number of rounds and the fewest number of rounds on a won game on all boards, has not lost on the largest two boards. For the 3×3 BOARD, the S-MCAGENT seems to have the fewest number of rounds as well, with the RULEBASEDAGENT as second. On the 5×5 BOARD, however, the D-MCAGENT needs fewer number of rounds than the RULEBASEDAGENT. However, this is based on a small amount of data, since there were few games lost for these agents as well.

Against the RULEBASEDAGENT

The S-RB-MCAGENT managed to win all 100 games on the ORIGINAL BOARD, so there is no data on the number of rounds on a lost game of this agent on this board. The average number of rounds of the other games can be found in Table 10.

Just like the experiments against the RANDOMAGENT, all agents need more rounds on a larger board. On the 3×3 BOARD the S-RB-MCAGENT needs the most rounds by far. The S-MCAGENT and the S-AS-MCAGENT both need the fewest rounds. On the other two boards the D-MCAGENT needs the most rounds, while the other agents need about the same number of rounds.

Once again most agents have a smaller number of rounds on a won game and more on a lost game. An agent that stands out is the S-RB-AS-MCAGENT that needs far more rounds on a lost game than on a won game. But like the games against the RANDOMAGENT, there are not many games that were lost, so not much data about them.

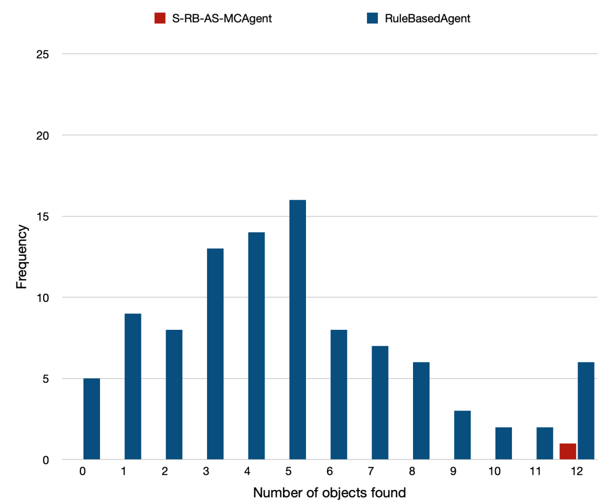


Figure 19: The distribution of OBJECTs found on a lost game by the S-RB-AS-MCAGENT against the RULEBASEDAGENT out of 100 initial games on the ORIGINAL BOARD.

	3 × 3 BOARD			5 × 5 BOARD			ORIGINAL BOARD		
	Total	**Won**	**Lost**	**Total**	**Won**	**Lost**	**Total**	**Won**	**Lost**
D-MCAGENT	8.3	7.6	9.2	43.3	38.8	54.9	69.4	62.7	93.0
S-MCAGENT	5.4	5.1	7.1	24.6	23.6	39.3	41.9	41.8	51.0
S-RB-MCAGENT	10.5	7.8	14.2	24.7	24.1	39.0	38.1	38.1	-
S-AS-MCAGENT	5.4	4.9	8.8	23.8	23.9	23.5	38.7	38.6	47.0
S-RB-AS-MCAGENT	6.5	6.2	11.0	26.5	25.9	46.3	38.5	38.0	85.0

Table 10: The average number of rounds out of a 100 games against the RULEBASEDAGENT on the total 100 games, the won games and the lost games.

9 Conclusions and future work

In summary, this thesis is about the digital version of the board game Labyrinth. We have created three kinds of agents: the RANDOMAGENT, the RULEBASEDAGENT and the MCAGENTs. There are different versions of the MCAGENTs: the SINGLE version, the DOUBLE version, the RULEBASED version and the ALLSTAR version. To determine the best agent, we have done experiments on three different board sizes. The best agent will not only win the most games, but also shows the most promise to win games in the other metrics.

9.1 Games against the RANDOMAGENT

In the experiments against the RANDOMAGENT, the RANDOMAGENT and the RULEBASEDAGENT perform the worst on all metrics and boards. That's why the second part of the experiments is continued with the D-MCAGENT and the S-MCAGENT.
In the original experiments, the S-MCAGENT seems to perform a little better than the D-MCAGENT on most metrics and boards. The second experiments confirm this with a bigger difference. This is the

reason why only the S-MCAgent is further tested with its different versions, the S-RB-MCAgent, the S-AS-MCAgent and the combined S-RB-AS-MCAgent.

9.2 The best agent

Now we look at all metrics combined. The D-MCAgent is easily the worst agent on all three boards. Clearly the separate Monte Carlo algorithms idea does not work as well as a combined Monte Carlo algorithm, probably because the S-MCAgents can see the result of the combined action while the D-MCAgent only sees the result of the movement if the insertion is already locked in.
The other agents are closer together on all metrics combined, except for the 3×3 board, where the S-RB-MCAgent performs way worse than the other players. Out of the other players, the S-RB-AS-MCAgent performs the best. Notice that the S-RB-AS-MCAgent is a combination of the S-AS-MCAgent that performs also pretty good on the 3×3 board and the bad performing S-RB-MCAgent. It seems the mistakes the S-RB-MCAgent makes on the smallest board are filtered out in the AllStar version.
The S-RB-MCAgent surprisingly performs much better on the 5×5 board and the original board. As said, all agents except for the D-MCAgent are pretty close to each other on these boards, but the S-RB-MCAgent performs best on the largest two boards.
So on the 3×3 board, the S-RB-AS-MCAgent performs the best and on the 5×5 board and the original board, the S-RB-MCAgent performs the best. Given the large differences on the 3×3 board and the small differences on the other boards, the best agent on all boards is the S-RB-AS-MCAgent.

9.3 Future work

While the S-RB-AS-MCAgent was determined to be the best agent in these experiments, there are still many possibilities for other kinds of agents and other kinds of experiments.

Other agents

An idea for an other agent is a more advanced RuleBasedAgent. This advanced version could use more information about the game. Right now the RuleBasedAgent chooses the first insert that places the current goal in the reachable maze cards and if it cannot find such an insert it chooses a random insert and moves to the closest position to the current goal. Maybe an advanced version could check all the inserts that places the current goal in the reachable maze cards for the move possibilities of the opponent. And if there is no insert to directly get the current goal, the agent could make an insert with the closest possible move to the current goal.
Another idea is to explore more versions of the MCAgents, for example a version that uses a different depth or a different number of playouts. We could also use more sophisticated versions, like Monte Carlo Tree Search.

Other boards

Another option is to experiment on different boards. The boards experimented on are the board of the original game and two smaller boards. A new option is a bigger board, like 12×12 or 15×15. Another option is a board that is not a square, like 3×5, 3×7 or 3×5.

Different rules

A last option is to explore the same game, but with altered rules. An option is that a player gets another turn if it manages to find its CURRENT GOAL. Another new rule is that players are not allowed to cross a path where another player is currently standing.
An important point of the original game is that players only know their current goal. An option is that all players know all the goals they will need to find. This can also be extended in a rule change where players need to find all their goals but the order in which they do does not matter. Finally, a version of the game where all the players know each other's current goals is of interest. This way it is easier to block another player from reaching their goal.

References

[BPW+12] C.B. Browne, E. Powley, D. Whitehouse, S.M. Lucas, P.I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1-43, 2012.

[Hamleys] Hamleys,
`https://www.hamleys.com/images/_lib/ravensburger-labyrinth-66843-0.jpg`
Visited on: August 6th, 2020

[KLL+95] Max J. Kobbert, Herbert Lentz, Horst Laupheimer, and Wolfgang Scheit. Spelregels De Betoverde Doolhof. *Ravensburger*, 1995.
Found on: `https://www.spelregels.eu/PDF/spelregels-de-betoverde-doolhof.pdf`
Visited on: August 6th, 2020

[Lee61] C.Y. Lee. An Algorithm for Path Connections and its Applications. *IRE Trans. Electron. Comput.*, 10:346-365, 1961.

[Magisterrex12a] Magisterrex Retro Games,
`https://magisterrex.files.wordpress.com/2012/02/blogamazeingboard.jpg`
Visited on: August 6th, 2020

[Magisterrex12b] Magisterrex Retro Games,
`https://magisterrex.files.wordpress.com/2012/02/blogamazeingmazecards.jpg`
Visited on: August 6th, 2020

[Magisterrex12c] Magisterrex Retro Games,
`https://magisterrex.files.wordpress.com/2012/02/blogamazeingtreasurecards.jpg`
Visited on: August 6th, 2020

[SAH+19] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T.P.. Lillicrap, and D. Silver. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *ArXiv*, abs/1911.08265, 2019.

[SHM+16] D. Silver, A. Huang, C.J.. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T.P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529:484-489, 2016.

[SHS+17] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T.P. Lillicrap, K. Simonyan, and D. Hassabis. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *ArXiv*, abs/1712.01815, 2017.