Opleiding Informatica

Applications of Monte Carlo on the Board Game Six

Galen Deering

Supervisors:
Walter Kosters and Jeannette de Graaf

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl 23/12/2018

Abstract

This thesis outlines the effectiveness and interplay of a few simple variations of the Monte Carlo approach, applied to the game simulation SixExe, a virtual adaptation of the board game Six: a two-player strategic tile-laying game in two phases. In Six, two players attempt to construct one of three winning shapes using tiles of their own colour. The second phase of the game introduces tile removal and tile capturing. We explore and discuss the different behaviours of the Monte Carlo-based algorithms applied to SixExe. In addition, we experiment with various alterations to the second phase for further insight into these behaviours. Finally we provide some ideas about how this research can be expanded further.

Contents

Chapter 1

Introduction

Six is an abstract strategy board game designed by Steffen Mühlhäuser as Steffen Spiele [Ste] made for two or four players. First published exclusively in German in 2003 [gam], and later republished with multi-lingual manuals (Fig. 1.1), the game uses a set number of red and black hexagonal tiles, and has no standardised playing field or physical board, meaning it can be played on any flat surface with adequate space.

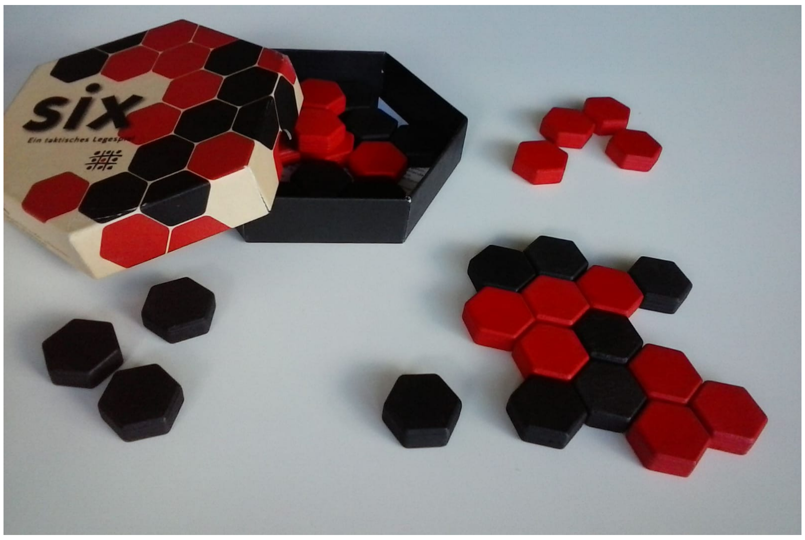


Figure 1.1: A later edition of Six.

The two players, or teams of two players, choose or are assigned one of the two colours, and attempt to construct one of three winning shapes by taking turns to place one tile of their colour per turn whilst trying to block the opponent from achieving the same. See Chapter 2 for a detailed explanation of the rules.
This paper aims to demonstrate the relative competence of a small number of low-level AI techniques applied

to the ruleset of Six. For that purpose we will utilise the functionally identical SixExe, a framework specifically designed to emulate the Six environment such that AI techniques can be applied.

Thesis Overview

After introducting the overall concepts of this thesis (Chapter 1), Chapter 2 will further define the rule set and environment of Six and SixExe. Chapter 3 will divert to present related articles and further reading. Chapter 4 expands on the definitions of the players, the experiments run on SixExe, and outlines the raw results of those experiments, after which Chapter 5 discusses the observations and speculations on those results. The final chapter, Chapter 6, summarises these findings in a comprehensible manner, listing the various conclusions that can be drawn from the experiments performed.

This document was created as a bachelor thesis under the supervision of the LIACS institute, Walter Kosters, and Jeannette de Graaf.

Chapter 2

Rules of Six

The rules of Six are fairly straightforward. The goal of the game is to construct one of three designated shapes consisting of six tiles of the player's colour (Fig. 2.1), whilst simultaneously preventing the opponent from achieving the same. The first to construct one such a shape on the playing field wins the game.

2.1 Standard Rules

There are 19 red tiles and 19 black tiles. Once both players have agreed on which player uses the red tiles (player Red) and which uses the black tiles (player Black), the starting player is determined at random and the first phase of play begins. That player then lays a tile of their colour in the (approximate) centre of the playing field. Placing a tile ends the turn, calling the next player to lay a tile adjacent to any of the tiles on the playing field. Any new tile must be lain such that it touches sides with at least one other tile.

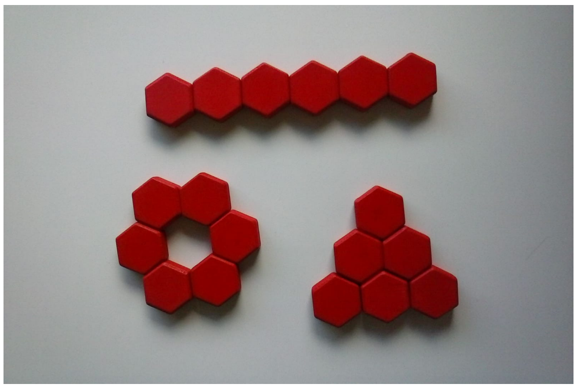


Figure 2.1: Winning shapes

Victory conditions

As soon as a player manages to construct one of the three goal shapes (Fig. 2.1) on the playing field using tiles of their own colour, that player has won the game and the game is over. The tiles adjacent to the completed shape do not influence this. For example, a completed triangle shape made of only red tiles will end the game in a red victory regardless of whether it is adjacent to more red tiles, or surrounded by either black ones or empty spaces. Similarly the closed ring of six tiles counts as a victory condition regardless of whether the centre space is empty, or is filled with a tile of either colour. In the event that multiple of the target shapes occur at once, e.g., the last tile to be laid completes both a triangle shape and a line shape of the same colour, there are no special rules and the game is ended in a victory for the colour of those shapes as normal. Since the game ends as soon as any of the winning shapes have been completed, it is physically impossible for winning shapes of different colours to exist on the playing field at the same time, and any winning shapes completed simultaneously will necessarily be of the same colour.

Second Phase

In the event that all tiles have been placed without any winning shapes being constructed, Six enters its second phase. In this phase, a player's turn consists of removing one of the tiles of their colour from the playing field and then placing that tile adjacent to the remaining tiles on the field just as in the first phase. Again, placing the tile ends the turn, and if it does not end the game, allows the next player to make their move. If a tile that is removed results in the playing field consisting of two[1] distinct disconnected groups of tiles, the smaller of the two groups is removed from the playing field and those tiles will remain out of play until the end of the game. This is called *Capturing* (Fig. 2.2). For the purposes of capturing, single disconnected tiles also count as their own groups.

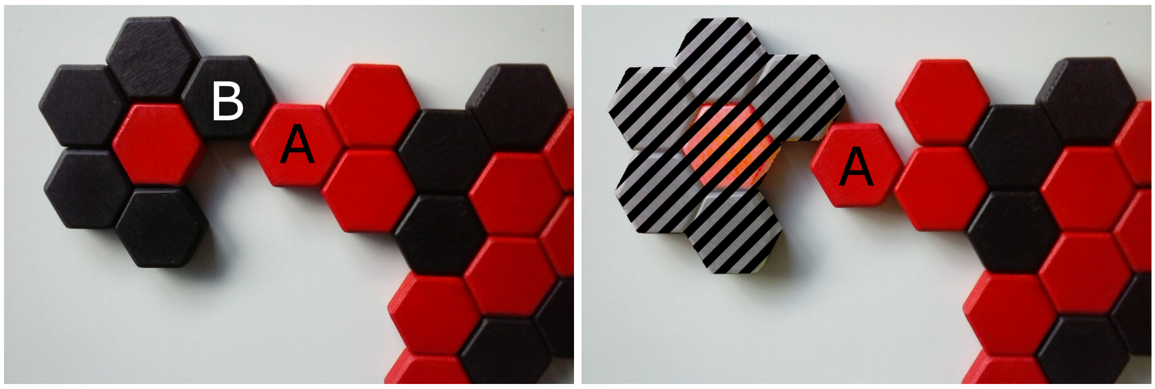


Figure 2.2: Red captures the group of six tiles counting from tile B by removing tile A, resulting in those tiles being removed.

When removing a tile results in the playing field being divided into two distinct groups of equal size, the player who removed the tile may choose which of the two groups to discard, leaving the other group as the remaining playing field. The second phase will continue until the game reaches one of its victory conditions

[1]Refer to Section 2.2.3 for the case where it results in three groups.

discussed in the section below. There are no rules in place to prevent an infinite game in the scenario where both players try to deliberately extend the game by avoiding any winning or capturing moves, as briefly discussed in Section 2.2.4.

Victory conditions (continued)

Just as in the first phase, the game ends as soon as either player places a tile that completes one of the winning shapes (Fig. 2.1), that player being the winner.
In addition, the second phase also allows victory by shut-out: if a turn with a capture results in a player having fewer than six tiles of their colour on the playing field, i.e., too few to construct a winning shape of six tiles, they have lost, and the game ends. If the other player still has six or more tiles on the playing field, they are pronounced the winner. If they do not, neither player has enough tiles left to win, and the game ends in a tie. During subsequent games, the loser of the last game may place the first tile. In cases of a tie, the starting player is determined at random as usual.

Four-player variant

It is also possible to play Six with four players instead of two. The game plays much the same as before, with two players per colour working in teams. The turn order progresses clockwise, assuming the players are seated by alternating the teams. E.g., if players A and B are team red, and players C and D are team black, turn progression will follow A C B D.

When both players in each team are governed by the same algorithms as their teammates, this distinction becomes moot, as every scenario becomes identical to the two-player equivalent with the same algorithms in play. Therefore we will not be exploring this variant using SixExe. However, there is an opportunity for further study of the four-player variant while using different AI techniques for each player in a team.

2.2 SixExe divergence

SixExe is the digital game simulation built to approximate Six for the purposes of AI experimentation. For the sake of this study some of the rules defined in the section above will have to be further refined, simplified, or otherwise modified. Any such differences are listed in this section.

2.2.1 Starting player

Whilst in Six the starting player is not tied to the player's chosen colour, SixExe rigorously keeps the red colour reserved for the starting player. This is done primarily for consistency in data when testing for whether the starting player has an advantage, and it has no effect on the actual gameplay as the different colours are functionally identical.

2.2.2 Half-SixExe

Half-SixExe is a simplified version of SixExe which consists of the first phase only. Players compete to place a winning tile combination and play the game as usual, with the major difference that when the players run out of tiles, instead of proceeding to Phase Two, the game ends.
If no players have laid a winning tile combination at this point, the match is pronounced a draw.

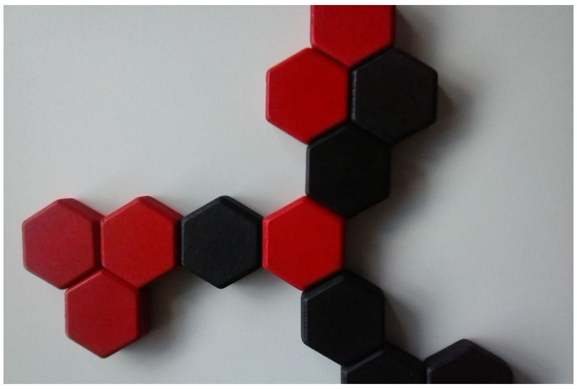


Figure 2.3: Removing the lone red tile will create three disconnected groups.

2.2.3 Anomalous captures

There are a few scenarios which can arise during the capture phase which SixExe handles differently than the standard rule set:

- When a capture produces two distinct tile groups of equal size, SixExe will select a group at random to be removed, rather than giving the player a choice of groups. This is done for programming simplicity and performance streamlining. Whilst this does change the game state in a way that can affect or prevent a victory for a given player, it is the author's assumption that these scenarios are sufficiently rare that this approach will not affect the results in any non-negligible way in the cases of the agents used in this paper.
- There exists an extremely rare scenario where a single tile can be the sole connecting factor between three tile groups that would otherwise be distinct (Fig. 2.3). The event where this tile is removed, resulting in three separate groups[2], is not defined in the standard ruleset. SixExe handles these situations by comparing the sizes of all three groups and removing the two smallest ones. In the event that these groups (or just the two largest) are of equal size, the choice of which of the largest to remove is made

[2]Considering the structure of the hexagonal tiles, three is the maximum number of groups that can be created in this way before the groups become non-disjunct.

arbitrarily, based on the order they are encountered in. As above, it is assumed that these scenarios are sufficiently not to affect the results in any meaningful way.

2.2.4 Absence of human tendencies

Whilst it does not fundamentally change the game in any way, there are some notable ways in which a physical game played between human players will statistically differ from those played digitally by AI agents. Mainly, there are the issues of removing fully enclosed tiles and potentially infinitely repeating moves. Both can occur only in the second phase.

The matter of enclosed tiles is purely a physical one. There are no rules to indicate that, during the second phase, choosing to remove a tile that has been completely surrounded by other tiles is anything other than a legal and valid move. However, in practice, it has been observed that when two human players reach this stage they are unlikely to choose a fully enclosed tile to remove, partly because removing such a tile will take considerable more effort than removing any other tile, which can be easily slid away, and partly because such a move is not likely to be strategically advantageous. The AI agents have no such qualms and will consider these moves as they will consider any other move.

The matter of infinitely repeating moves can occur when a situation arises in the second phase where both players can reliably remove and replace some tiles without capturing or otherwise significantly changing the scenario. It is entirely possible, although not statistically likely, to repeat non-impactful moves ad infinitum. Where human players might instinctively avoid this, the AI agents might under certain conditions repeat moves fruitlessly for a considerable time. For authenticity, no safeguards have been placed to prevent this in the experiments performed.

Chapter 3

Related Work

This chapter will discuss a number of relevant and related articles on similar subjects. This paper deals with the game Six and a number of Monte Carlo methods. An expansion on the concept of the Monte Carlo approach is available in the book "An introduction to sequential Monte Carlo methods" [DDFG01] by Doucet et al. It deals with the outlining of sequential versions of the typical Monte Carlo algorithm and a variety of its applications. Further expansion on and deeper exploration of techniques of the Monte Carlo family can be found in the works of Cameron Browne et al. For example, the paper "A Survey of Monte Carlo Tree Search Methods" [BPW$^+$12] explores the current state of research into Monte Carlo Tree Search, which aims to be more efficient than the standard version used in this research on Six. In doing so it outlines the further potential of the field.

The game Hex [hex] bears some similarities to Six. Both are in their basic form two-player tile-laying games where the object is to complete a certain abstract or shape on the board by laying single hexagonal tiles during the player's own turn. The methods applied to Six could be adapted for Hex, and vice versa. C. Browne explores this in his book "Making the right connections" [Bro00]. The article "On a decomposition method for finding winning strategy in Hex game" [YLP01] details a method to find strategies for the game, as the title implies. "The game of Hex and the Brouwer fixed-point theorem" [Gal79] takes a direct mathematical approach, providing proof of equivalence between Hex strategies and the Brouwer fixed-point theorem.
To see Monte Carlo Tree Search applied to Hex, refer to the article "Monte Carlo tree search in Hex" [AHH10].

Chapter 4

Experiments

This chapter will detail the AI players and the various experiments they are applied to, and shows the results of those experiments in a number of different forms with the intent to clearly demonstrate the parts relevant to the conclusions in Chapter 5.

4.1 Players

This section describes the various agents used in the experiments detailed in this chapter

Random player

The Random player will, as its name implies, act with entirely random moves. At each juncture where the Random player is called upon to make a move, it will select a valid move from the list, of course, uniformly at random, and proceed with that move. It is the most straightforward algorithm and is liable to only ever win a game by accident.
In the second phase, the Random player will be tasked to *remove* a tile from the board, again uniformly at random, after which, if the game has not ended, it will choose a random valid move to place it in just as in the first phase. The already copious list of valid moves is in the second phase likely to expand almost exponentially, whilst the amount of moves with any kind of tactical advantage is barely likely to expand at all, effectively further shrinking the already small chance that the Random player will affect any advantageous moves at all in this phase.

Weak Monte Carlo player

Expanding on the Random player, a typical Monte Carlo algorithm seeks to use the random element in an intelligent way. It does this by introducing a *hypothetical* element. In practice, within the context of SixExe, the Weak Monte Carlo player observes the current state of the original or "real" game board, and copies this state onto a new parallel board. We will refer to this new board as the "hypothetical" game board. Once an identical game state has been established, the Weak Monte Carlo player chooses a single random move, here referred to as the **cursor**, performs it on the hypothetical board, and allows this parallel game to complete using the Random algorithm for both players. The cursor move is then given a score based on whether the hypothetical game has been won or not, and the result is saved and compared to that of the next cursor, which is another completely random move[1]. The process continues for a set number of loops, each loop evaluating a new random cursor move from scratch. Once all loops have been completed, the Weak Monte Carlo algorithm executes the best cursor move it found onto the original game board. Since this version of Monte Carlo only performs a single Random vs Random game per cursor move, the best value of any such move can never be greater than one single win. In the event where all evaluated moves were losses, Weak Monte Carlo will execute the first move it tried, as none of the following options were better than that one. In essence, Weak Monte Carlo chooses and executes the first random move it finds where the performed Random vs Random game coincidentally ends in a win in its own favour. The algorithm needs not continue to evaluate other moves once it has found one which it has evaluated as a winning move, but it does so anyway due to the algorithm's structure.

Basic Monte Carlo player

A more typical version of the Monte Carlo approach will apply hypothetical random games numerous times per valid available (cursor) move, thereby ensuring that every possible move will be evaluated and furthermore giving each cursor move a more tractable evaluation, E.g., take the scenario where Random vs Random games engaged after move 2 might first win once and then lose four times, whilst Random vs Random games engaged after move 5 might win four times and tie once. Weak Monte Carlo would not differentiate between these two and would in this scenario choose move 2, since it is the earliest move it found where the first result was a win, whereas Basic Monte Carlo could compare the scores $1-1-1-1-1=-3$ and $1+1+1+1+0=4$ and choose move 5.
The step by step process of Basic Monte Carlo is as follows: Upon being called, Basic Monte Carlo reads the current state of the game board, creates an identical "hypothetical" game board, takes the first valid move of the list of valid moves as a cursor move, executes that move on the new board, and then performs a predetermined number of games of Random vs Random that start on that board state, evaluating the cursor move as this process completes. It increments the value score of the cursor for every win, and decrements for every loss. The result after these loops are completed for this move is compared to the result of the previous cursor move, if applicable, and the move with the most favourable value is saved. If the current cursor move is not the last valid move, it will repeat this process with the next valid move as the new cursor. Effectively, Monte Carlo will evaluate each possible move with a precision that grows in correlation to the amount of Random vs Random games performed per move, and pick the move with the best value it found. There are

[1]This can result in the same move being evaluated more than once, but since it will be treated as a new scenario, it will not affect the previous evaluation of the move, nor will the previous evaluation affect the current one.

many different ways in which this Monte Carlo application can be expanded or enhanced. A number of these methods are discussed in Chapter 6.

4.2 Half-SixExe

This section shows the results of various experiments applied on Half-SixExe using the agents detailed above. These experiments are performed purely on the first phase. If there is no winner by the time the first phase ends, the game is counted as a draw. This means that, considering each player begins with 19 tiles in the standard version, a draw in this case means any game that reaches 38 tiles placed without either player achieving a winning shape. The subsections below show the results of each performed experiment.
All experiments have been run with Red as the starting player.

Random vs Random

Fig. 4.1 shows the results of Random versus Random. Random ties often, and the starting player seems to beat the second player more than vice versa.

Results	
Red victories	128,477
Black victories	91,486
Tied games	780,037
Total runs	1,000,000

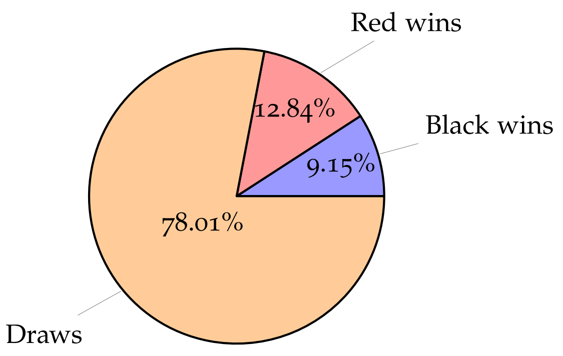


Figure 4.1: One million games of Random vs Random.

Weak Monte Carlo vs Random

The Weak Monte Carlo versus Random experiment (Figures 4.2, 4.3, 4.4, and 4.5) has been repeated numerous times each with a tweaked number of random cursors for Weak Monte Carlo to test. It has been run for the values 1, 5, 10, 20, 30, 40, 50, 100, and 150.

Results	
Red victories	1,356
Black victories	895
Tied games	7,749
Total runs	10,000

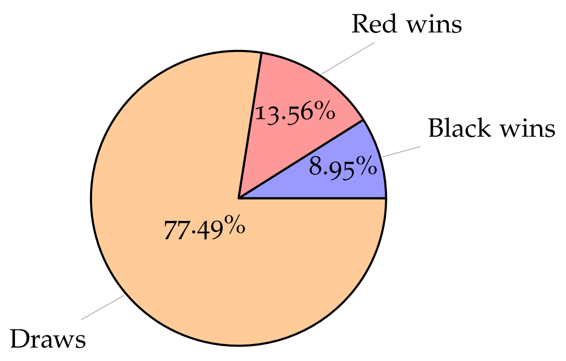


Figure 4.2: Weak Monte Carlo (Red) with one random cursor; effectively Random.

Results	
Red victories	4,582
Black victories	687
Tied games	4,731
Total runs	10,000

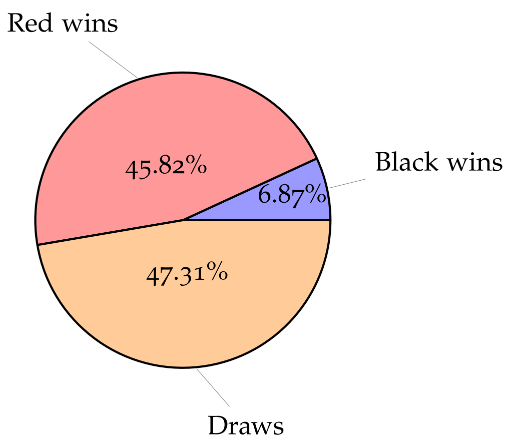


Figure 4.3: Weak Monte Carlo (Red) with 5 random cursors.

Results	
Red victories	6,237
Black victories	584
Tied games	3,179
Total runs	10,000

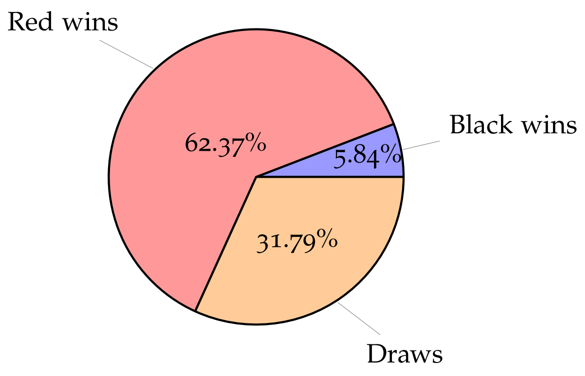


Figure 4.4: Weak Monte Carlo (Red) with 10 random cursors.

For brevity, the win rates will be distilled into a line graph, see Fig. 4.5.

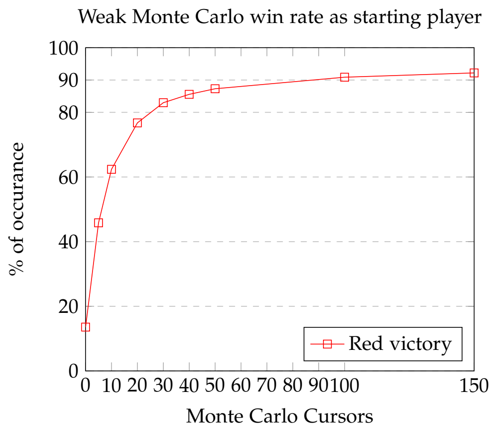


Figure 4.5: Every win percentage in this above graph is determined from a 10000-fold repeated experiment. Random wins and ties have been omitted.

Random vs Weak Monte Carlo

The Random vs Weak Monte Carlo experiment (Figures 4.6 and 4.7) is identical to the above Weak Monte Carlo vs Random experiment, with the key difference that Random is the starting player. Weak Monte Carlo plays as black. For brevity, only the results of the iteration with 10 random cursors are shown in full.

Results	
Red victories	1,009
Black victories	5,211
Tied games	3,780
Total runs	10,000

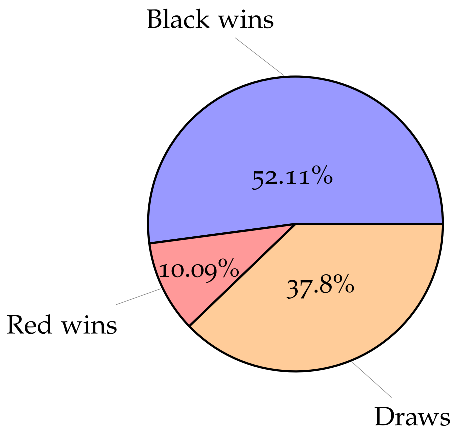


Figure 4.6: Weak Monte Carlo (Black) with 10 random cursors.

Figure 4.7 shows the win rates for the various random cursor counts used in this experiment. Random wins and ties have been omitted.

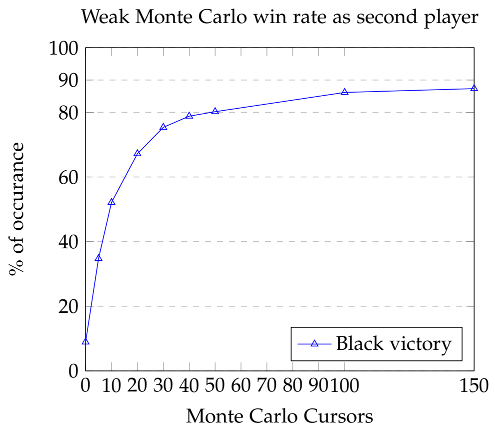


Figure 4.7: Every win percentage in this graph is determined from a 10000-fold repeated experiment.

Weak Monte Carlo vs Weak Monte Carlo

The Weak Monte Carlo vs Weak Monte Carlo experiment (Fig. 4.8) pits two identical Weak Monte Carlo algorithms against each other. Just as the experiments above, this has been repeated with the cursor values 1, 5, 10, 20, 30, 40, 50, 100, and 150.

Results	
Red victories	5,763
Black victories	4,208
Tied games	29
Total runs	10,000

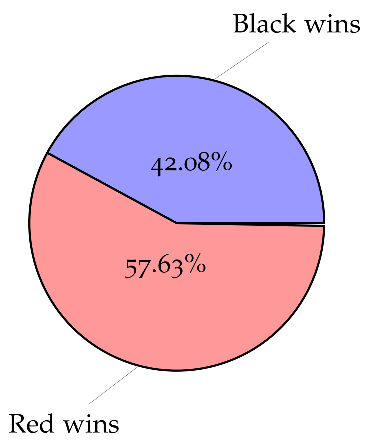


Figure 4.8: Weak Monte Carlo versus Weak Monte Carlo each using 100 loops. 0.29% Draws.

Basic Monte Carlo vs Random

Figure 4.9 shows the results for the Basic Monte Carlo versus Random experiments in Half-SixExe. Basic Monte Carlo generally wins against Random, but there is the rare occasion where Random triumphs even at the higher numbers of cursors.

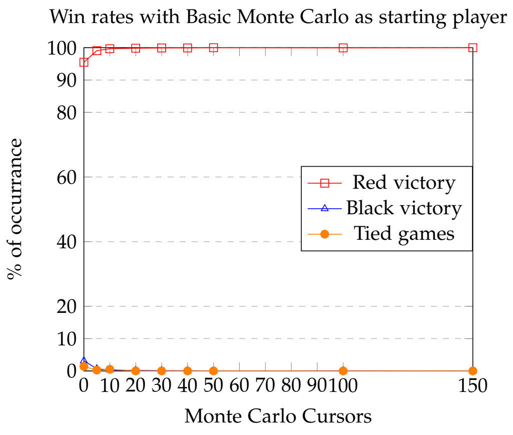


Figure 4.9: Every win percentage in this graph is determined from a 10000-fold repeated experiment.

Basic Monte Carlo vs Weak Monte Carlo

Figure 4.10 shows the results for the Basic Monte Carlo versus Weak Monte Carlo experiments in Half-SixExe. Weak Monte Carlo provides a slightly greater challenge to Basic MC than Random does.

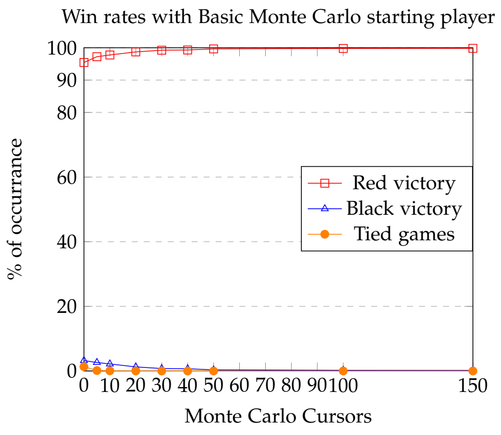


Figure 4.10: Every win percentage in this graph is determined from a 10000-fold repeated experiment.

Basic Monte Carlo vs Basic Monte Carlo

Figure 4.11 shows the results for the Basic Monte Carlo versus Basic Monte Carlo experiments in Half-SixExe. Basic Monte Carlo plays relatively even when its opponent is itself.

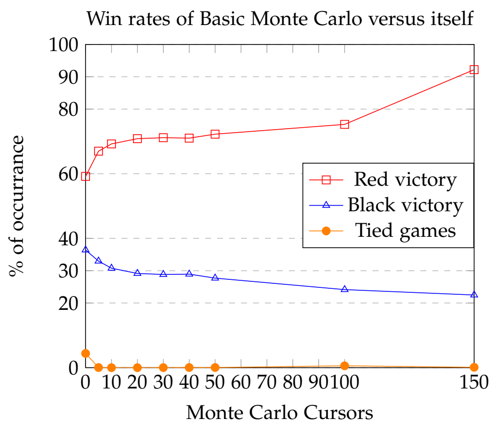


Figure 4.11: Every win percentage in this graph is determined from a 10000-fold repeated experiment.

4.3 SixExe

Here we unleash the agents on the full scope of the game, from start to finish, and gather the results. This experiment set introduces the second phase, which will at the default number of tiles begin after the 38th move, assuming neither player has won the game in phase one. To illustrate, we will also show the average number of moves per game alongside the victory rates. Since it is no longer possible to reach a draw simply by placing 19 tiles each, draws in this experiment are any situation where a sufficient amount of tiles of both colours have been captured such that neither player can possibly lay any winning shapes any more. All experiments have been run with Red as the starting player.

Random vs Random

Figure 4.12 shows the results of the Random versus Random experiment in SixExe. The games where Random plays against itself often end in the second phase, bearing one of the highest average moves per game.

Results	
Red victories	4,711
Black victories	4,126
Tied games	1,163
Average moves per game	53.01
Total runs	10,000

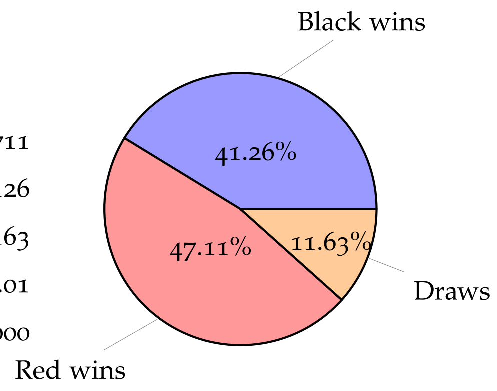


Figure 4.12: There are significantly fewer ties than in HalfSix, and the starting player seems to win slightly more often than the other.

Weak Monte Carlo vs Random

Figure 4.13 shows the results of the Weak Monte Carlo versus Random experiment in SixExe. Weak Monte Carlo reaches the second phase frequently and seems to plateau around a 70% win rate after 20 or more cursors.

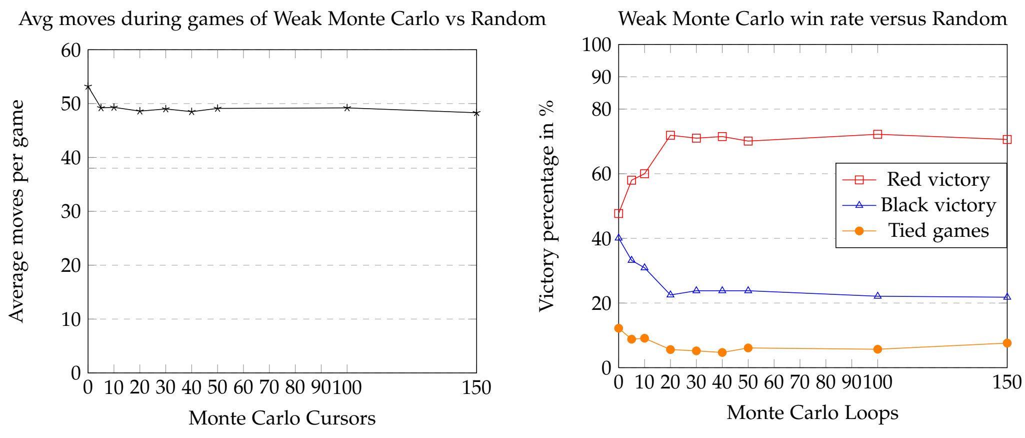


Figure 4.13: Every win percentage in this graph is determined from a 1000-fold repeated experiment.

Weak Monte Carlo vs Weak Monte Carlo

Figure 4.14 shows the results of the Weak Monte Carlo versus Weak Monte Carlo experiment in SixExe. The behaviour of Weak Monte Carlo in the second phase appears to be erratic when playing against itself. This experiment appears to be the most evenly matched.

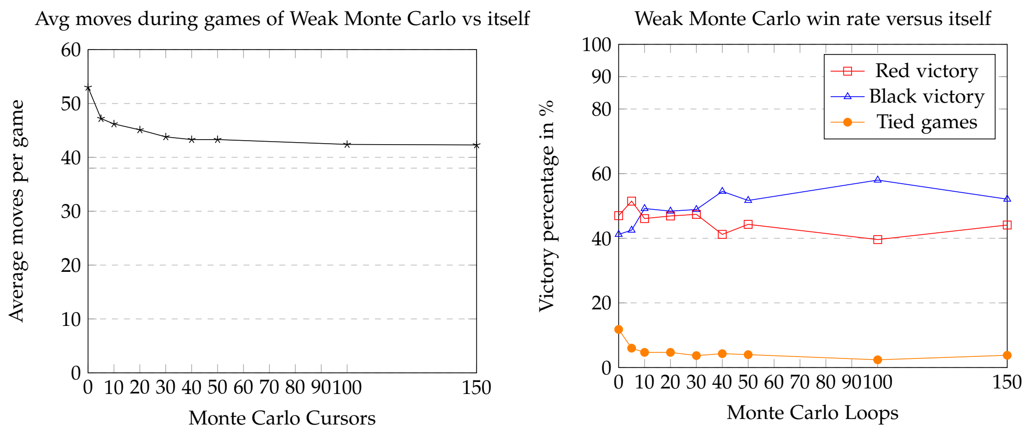


Figure 4.14: Every win percentage in this graph is determined from a 1000-fold repeated experiment.

Basic Monte Carlo vs Random

Figure 4.15 shows the results of the Basic Monte Carlo versus Random experiment in SixExe. As could be expected, Basic Monte Carlo has a high win rate versus the Random player. However, there appears to be a curious spike of extra activity around the 5 play-outs mark. Despite this, Basic Monte Carlo manages to end the game before the second phase with remarkable consistency.

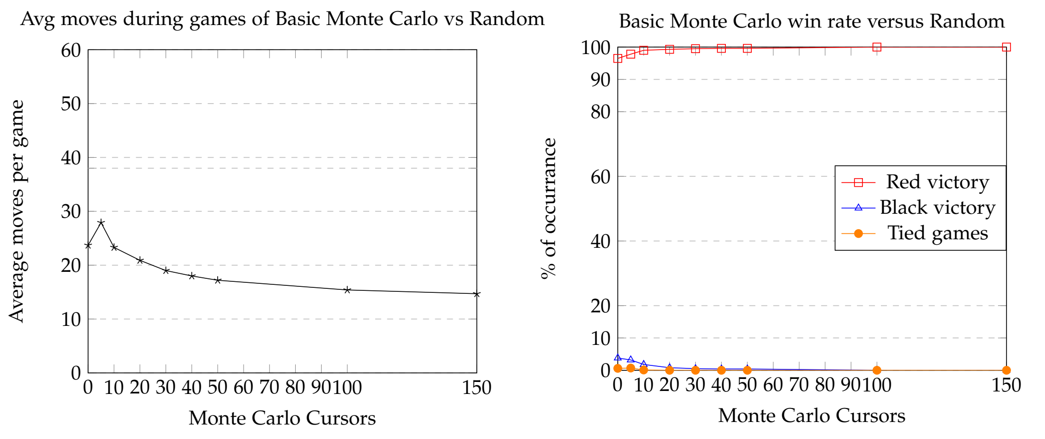


Figure 4.15: Every win percentage in this graph is determined from a 1000-fold repeated experiment.

Basic Monte Carlo vs Weak Monte Carlo

Figure 4.16 shows the results of the Basic Monte Carlo versus Weak Monte Carlo experiment in SixExe. The same burst of activity can be seen when Basic MC faces Weak MC, coinciding with a small gain in wins for Weak Monte Carlo.

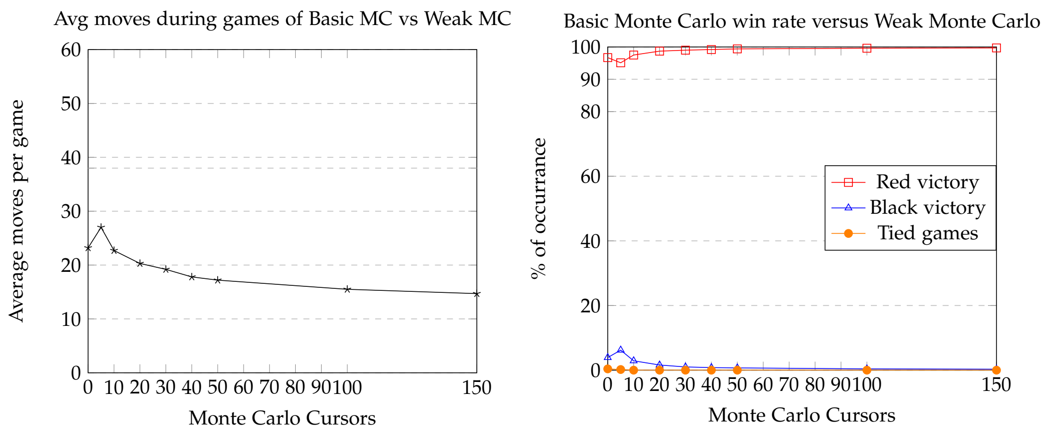


Figure 4.16: Every win percentage in this graph is determined from a 1000-fold repeated experiment.

Basic Monte Carlo vs Basic Monte Carlo

Figure 4.17 shows the results of the Basic Monte Carlo versus Basic Monte Carlo experiment in SixExe. Basic Monte Carlo seems to consistently end the game in the first phase even when up against an equal opponent.

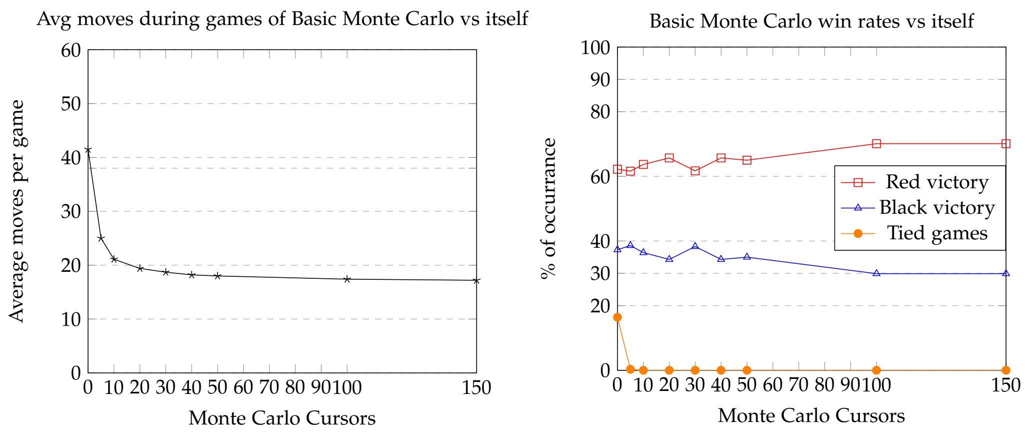


Figure 4.17: Every win percentage in this graph is determined from a 1000-fold repeated experiment.

4.4 SixExe with Diminished Monte Carlo

The SixExe with Diminished Monte Carlo experiment (hereafter referred to as Diminished SixExe) is a variant on the SixExe experiment. The agents are again released on the full scope of the game, but the agents of the Monte Carlo family have been modified not to consider the second phase of play in their hypothetical playouts when the real game is still in its first phase. In other words, since the second phase typically begins after 38 moves, i.e., when all tiles have been placed, any Monte Carlo hypothetical game played before the 38th move will terminate after the 38th tile has been placed. If at that point there is no winning shape on the board, where ordinarily it would begin the second phase of play, the hypothetical game will instead count as a draw. This cuts out a significant portion of the computation, and we will be comparing the results to those in Section 4.3. Since Random vs Random is unaffected by this change, that particular experiment will not be run again. Again, all experiments have been run with Red as the starting player.

Weak Monte Carlo vs Random

Figure 4.18 shows the results of the Weak Monte Carlo versus Random experiment in Diminished SixExe. Weak Monte Carlo appears to gain higher win rates quite steadily as the amount of cursors approaches 100.

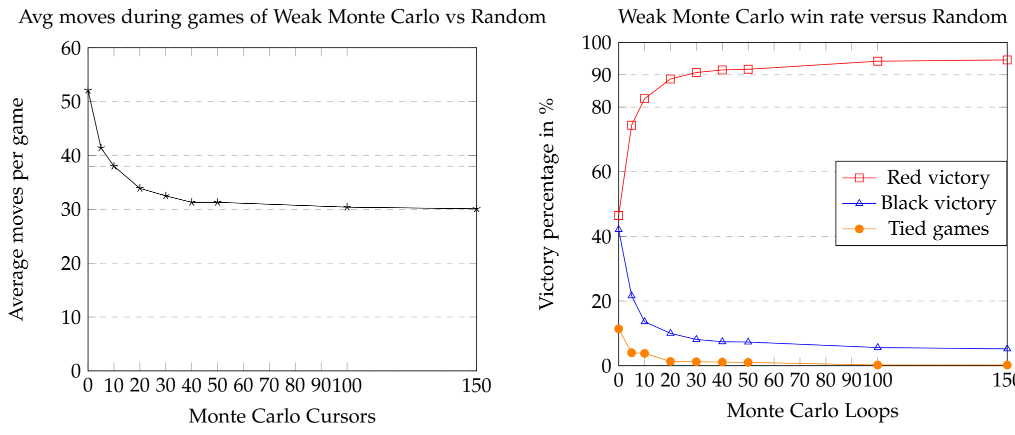


Figure 4.18: Every win percentage in this graph is determined from a 1000-fold repeated experiment.

Weak Monte Carlo vs Weak Monte Carlo

Figure 4.19 shows the results of the Weak Monte Carlo versus Weak Monte Carlo experiment in SixExe. When up against itself, Diminished Weak Monte Carlo manages to maintain the advantage as the starting player now that it perceives reaching the second phase to be worth as much as a draw.

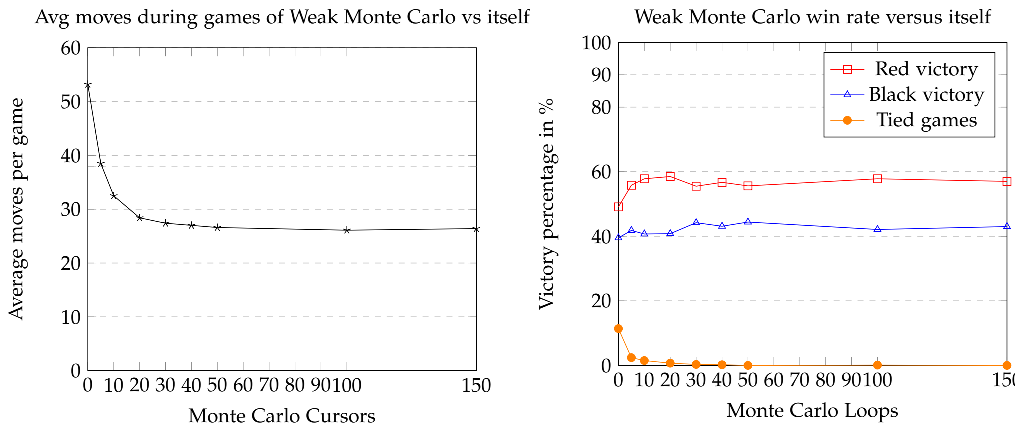


Figure 4.19: Every win percentage in this graph is determined from a 1000-fold repeated experiment.

Basic Monte Carlo vs Random

Figure 4.20 shows the results of the Basic Monte Carlo versus Random experiment in SixExe. Basic Monte Carlo manages to avoid the second phase completely using this criteria.

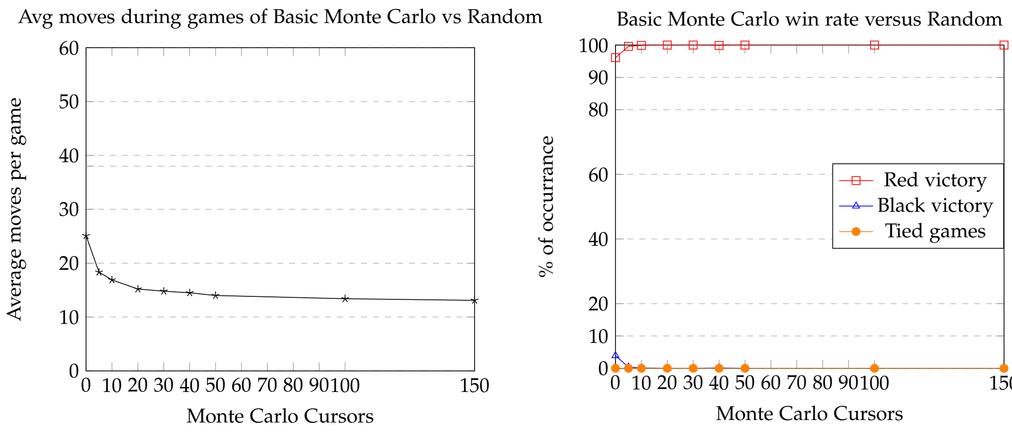


Figure 4.20: Every win percentage in this graph is determined from a 1000-fold repeated experiment.

Basic Monte Carlo vs Weak Monte Carlo

Figure 4.21 shows the results of the Basic Monte Carlo versus Weak Monte Carlo experiment in SixExe. Weak Monte Carlo manages to win slightly more often against Basic Monte Carlo around the 5 cursor/5 play-out mark.

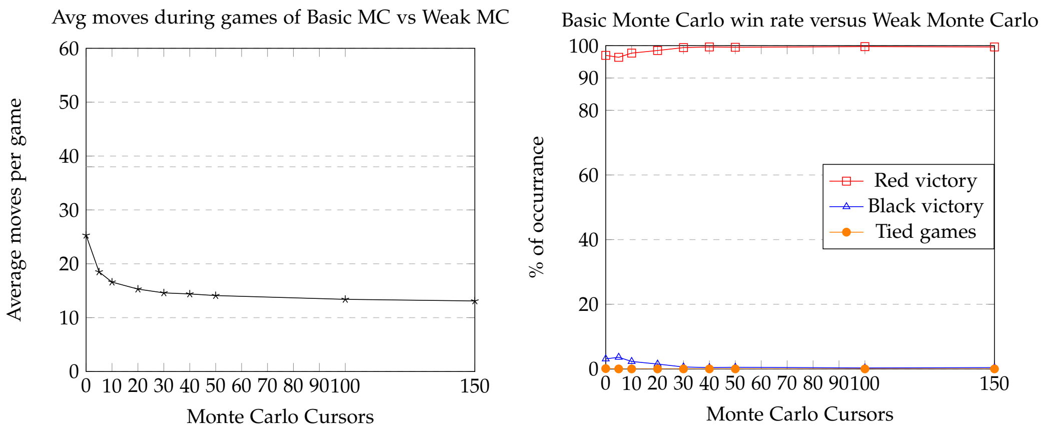
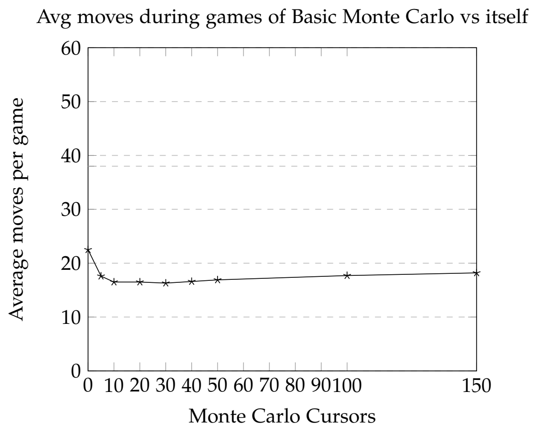


Figure 4.21: Every win percentage in this graph is determined from a 1000-fold repeated experiment.

Basic Monte Carlo vs Basic Monte Carlo

Figure 4.22 shows the results of the Basic Monte Carlo versus Basic Monte Carlo experiment in SixExe. Diminished Basic Monte Carlo versus itself yields very similar results to the regular Basic Monte Carlo versus itself experiment (Fig. 4.17), with the notable differences that the average game no longer reaches the second phase with singular play-outs, and that the disparity in win rates of the first and second player is slightly greater in the Diminished Monte Carlo experiment.



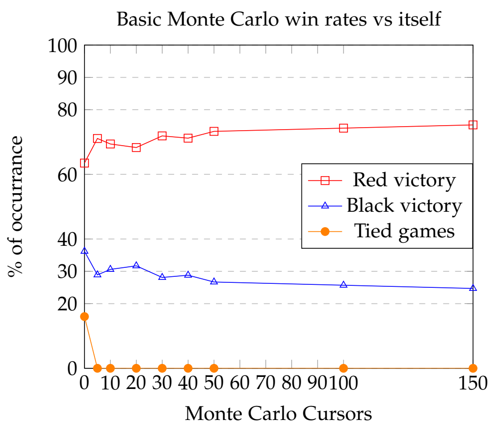


Figure 4.22: Every win percentage in this graph is determined from a 1000-fold repeated experiment.

4.5 SixExe with Monte Carlo exclusive to the Second Phase

In the final experiment, SixExe with Monte Carlo exclusive to the Second Phase, hereafter referred to as Second-Phase Focused SixExe, the agents are adjusted such that they only affect random moves during the first phase of play. In essence, for the first 38 moves, the game plays out with one Random player opposing another Random player. Then, when the last tile has been placed and the second phase begins, the agents resume play by their normal methods. For example, a Weak Monte Carlo versus Random match would start first with both players performing random moves from the start of the first phase until the start of the second phase. At that point, the first player (Weak Monte Carlo) would begin playing as Weak Monte Carlo as normal, and the second player would also resume playing their usual way, which in this example is still random. Just as in Diminished SixExe (Section 4.4), this experiment has no effect on the Random versus Random setup, and that experiment will not be repeated. All experiments have been run with Red as the starting player.

Weak Monte Carlo vs Random

Figure 4.23 shows the results of the Weak Monte Carlo versus Random experiment in Second-Phase Focused SixExe. Naturally, the average number of moves always remains well beyond the 38-moves mark, despite the tendency for Random versus Random to end the game in the first phase approximately 22% of the time (See Fig. 4.1).

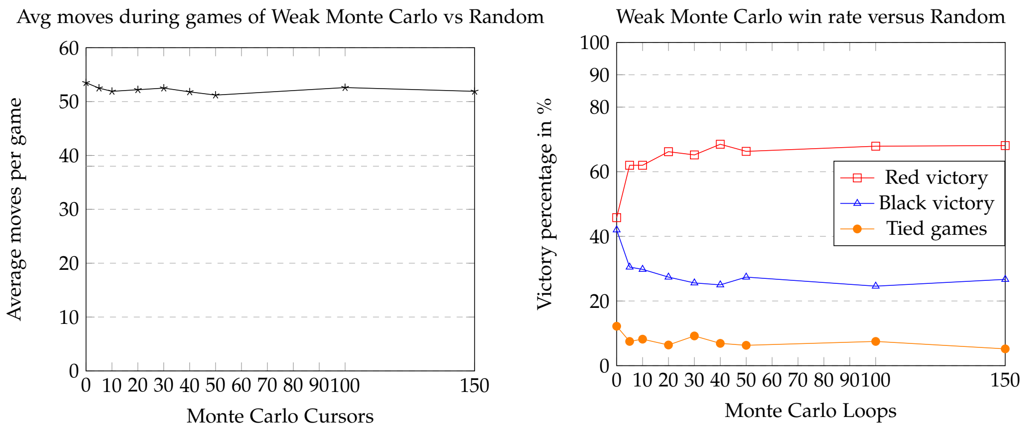


Figure 4.23: Every win percentage in this graph is determined from a 1000-fold repeated experiment.

Weak Monte Carlo vs Weak Monte Carlo

Figure 4.24 shows the results of the Weak Monte Carlo versus Weak Monte Carlo experiment in Second-Phase Focused SixExe. Here Weak Monte Carlo seems to show great consistency as the results seem highly comparable to those of Random versus Random in the SixExe experiment (Fig. 4.12) regardless of the amount of cursors used. This could be a fine example of additional cursors past a certain point having very little meaningful effect on this algorithm.

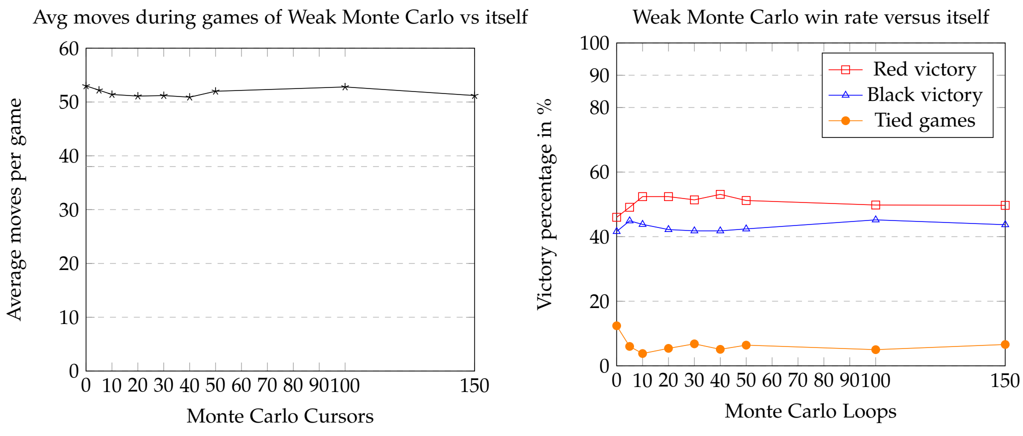


Figure 4.24: Every win percentage in this graph is determined from a 1000-fold repeated experiment.

Basic Monte Carlo vs Random

Figure 4.25 shows the results of the Basic Monte Carlo versus Random experiment in Second-Phase Focused SixExe. The average number of moves remains high, but lies significantly closer to the second phase border mark compared Figures 4.23 and 4.24. Surprisingly, Basic Monte Carlo does not hit the 100% win rate mark even at 150 cursors. This could simply be due to the boards built by the Random versus Random phase having an equal chance to be structured in one of either player's favour, granting either player regular opportunity over the course of one thousand games for an easy (or coincidental) win.

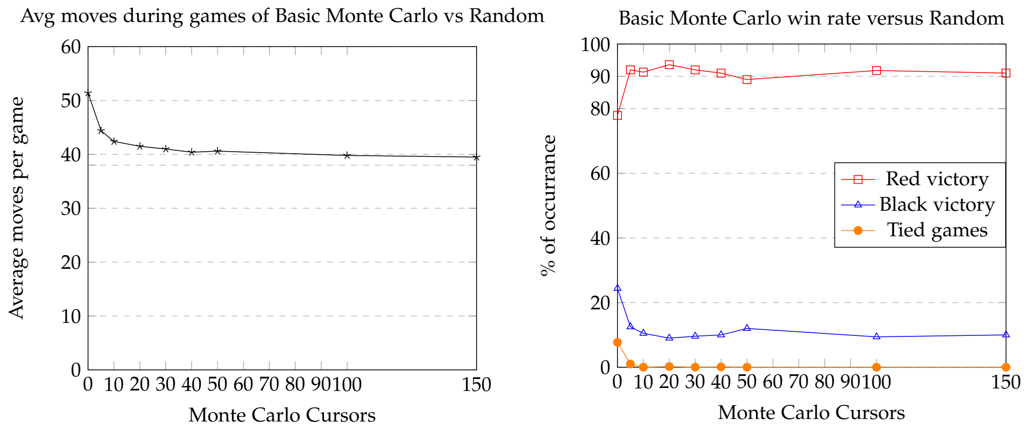


Figure 4.25: Every win percentage in this graph is determined from a 1000-fold repeated experiment.

Basic Monte Carlo vs Weak Monte Carlo

Figure 4.26 shows the results of the Basic Monte Carlo versus Weak Monte Carlo experiment in Second-Phase Focused SixExe. Just as in the Second-Phase Focused SixExe Weak Monte Carlo versus Random experiment (Fig. 4.24), judging by the results, the Weak Monte Carlo player seems almost interchangeable with the Random player, observing the similarities between figures 4.26 and 4.25. Despite these similarities, figure 4.23 shows that the behaviour of Weak Monte Carlo is still far from identical to that of Random.

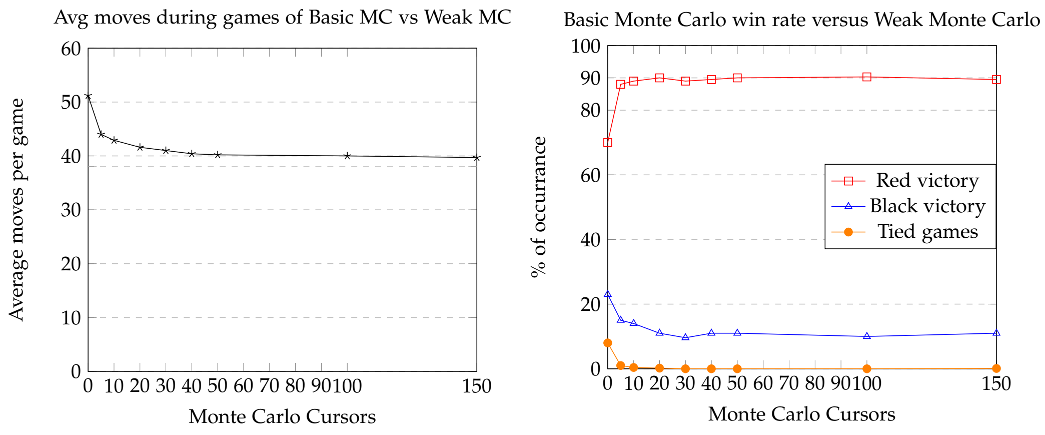


Figure 4.26: Every win percentage in this graph is determined from a 1000-fold repeated experiment.

Basic Monte Carlo vs Basic Monte Carlo

Figure 4.27 shows the results of the Basic Monte Carlo versus Basic Monte Carlo experiment in Second-Phase Focused SixExe. Basic Monte Carlo manages to end the game quickly with increasing consistency in correlation to the amount of cursors, with the average number of moves at 150 cursors reaching 38.5, which is very close to the place where the second phase begins (38 moves).

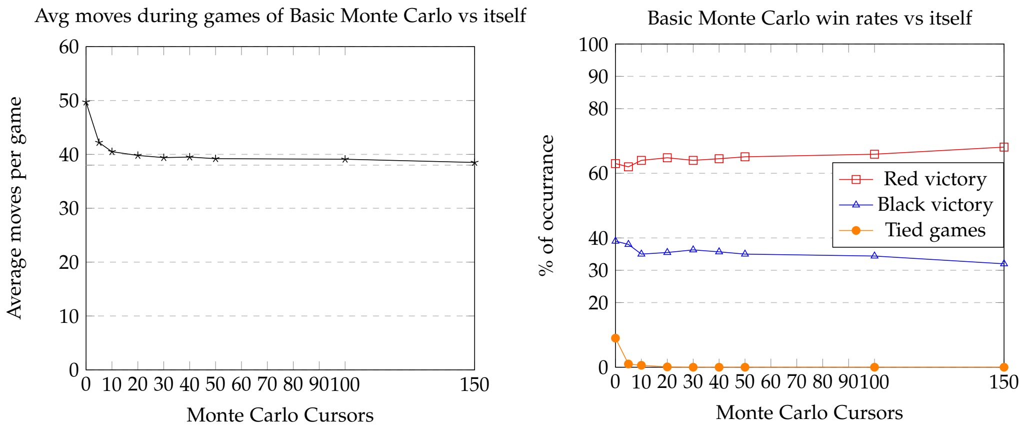


Figure 4.27: Every win percentage in this graph is determined from a 1000-fold repeated experiment.

Chapter 5

Evaluation

When observing the results of the various experiments, a number of notable phenomena arise. At a glance, there is the trivial observation that any form of Monte Carlo far outperforms the Random player (Fig. 5.2).

5.1 Starting player advantage

Figure 5.1 shows that, when the winrates of Weak Monte Carlo as a starting player and Weak Monte Carlo as the second player are shown in the same graph, the latter clearly beats the opponent (in this case the Random player) with less frequency. This alone is enough to suggest that the starting player has an advantage over its opponent. To confirm this, we turn to the experiments where an algorithm is opposing itself; where both the Red and the Black player are guided by the same rules and probabilities. Figures 4.1 and 4.8 in Chapter 4 show the results of Random versus Random and the Weak Monte Carlo versus itself results. In both cases, the Red player clearly has a greater win rate than the Black player. And as stated in Sections 4.2 and 4.3 in Chapter 4, all experiments have been set up as such that Red is always the starting player. Thus, the starting

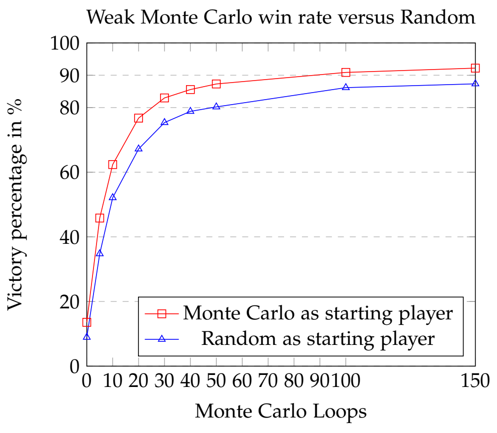


Figure 5.1: Weak Monte Carlo shows a notably higher win rate as the starting player.

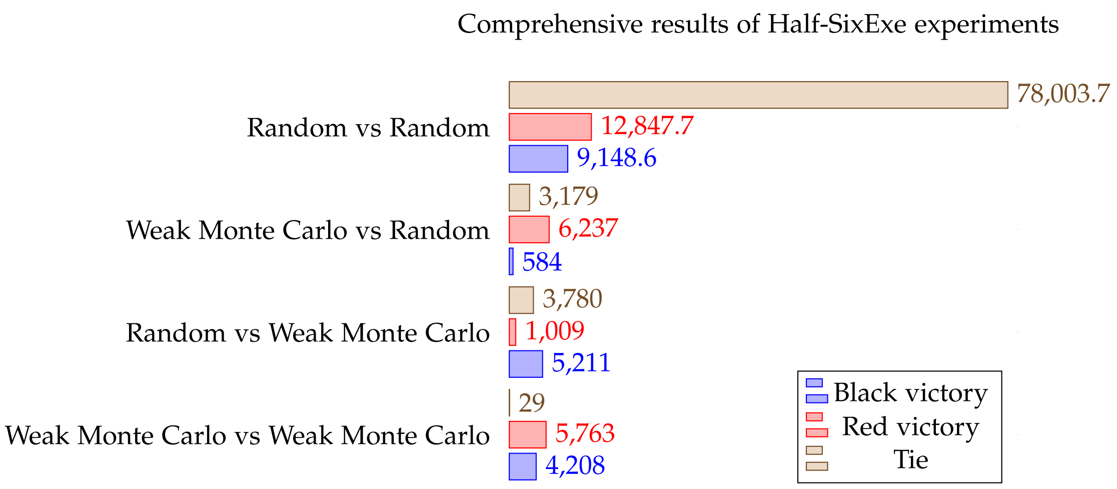


Figure 5.2: An excerpt of the results of the Half-SixExe experiments in comprehensive form. Note: Random vs Random results in this graph have been scaled down by a factor of 10 for ease of viewing.

player receives an increased win rate regardless of algorithm efficiency. Figure 5.2 demonstrates this holds true across all performed Weak Monte Carlo experiments in Half-SixExe, as even the Random player's win rate increases almost twofold when it is the starting player in a match versus Weak Monte Carlo. The theory holds true outside of Half-SixExe as well, as seen in figures 4.17, 4.22, and 4.27. Strangely, figure 4.14 appears to contradict this. We discuss a possible reason for this in Section 5.2.

5.2 Weak Monte Carlo's weakness

Comparing the results of Weak Monte Carlo versus Random shown in Figures 4.5, 4.13, and 4.18, it becomes clear that the Weak Monte Carlo algorithm seems to behave relatively erratically when it is allowed to run hypothetical games over the second phase. This is further reinforced by the seemingly anomalous victories of the second player in the Weak Monte Carlo versus Weak Monte Carlo SixExe experiment (Fig. 4.14), which would otherwise contradict the Starting Player Advantage theory discussed in Section 5.1. It is likely that this effect is caused by the way Weak Monte Carlo goes for the first move that results in a win when played out with a Random versus Random game. It is worth noting that any given scenario in the second phase has a greater magnitude of possible moves than in the first phase, making any specific single move far less likely to be picked at random. It immediately follows that any given chain of random moves that might result in the hypothetical win is extremely unlikely to be repeated by the following moves on the real board, as Weak Monte Carlo will evaluate another completely random move every step of the way, and is very unlikely to find the optimal path to victory by chance.

5.3 Efficiency of Diminished Monte Carlo

When the results of the SixExe experiments (Section 4.3) are compared to their equivalents in the SixExe with Diminished Monte Carlo (Section 4.4) there are two items that appear significant.
Firstly, the average moves per game is consistently lower in all the Diminished Monte Carlo experiments. It is conceivable that this is caused by the way that reaching the second phase is now considered to be a draw by Monte Carlo's hypothetical games. A draw will be scored as zero; neither negative nor positive. Relative to any moves that might result in a win, this makes moves that are likely to result in the second phase being reached *less desirable.* This in turn creates an increased likelihood for any of the Monte Carlo algorithms to choose a move that has a chance to grant a victory sooner (i.e., within fewer moves) compared to the equivalent scenario in non-diminished SixExe.
Secondly, the win rates in the SixExe with Diminished Monte Carlo experiments are more consistent than their SixExe counterparts. The same barrier created by reaching the second phase being undesirable might also reinforce the tendency to avoid risky states where there are a great number of opportunities for the opponent to turn the game around, be it by blocking or capturing. The risk of one's tiles being captured specifically is greatly decreased when the game ends before the second phase.
In addition to these matters the Diminished Monte Carlo technique is far more computationally efficient. The second phase introduces a new mechanic which might potentially affect large portions of the board with a single move. Since the Monte Carlo hypothetical games need to keep track of all possible potential effects, this makes any single move in the second phase more computationally intense than any given move in the first phase. There is also the fact that a single move in the second phase consists of two separate decisions (which tile to remove and where to place it after the potential capture resolves) which are each decided based on another round of comparing hypothetical moves however many cursors or hypothetical play-outs Monte Carlo is allowed to compute. This means any given second phase move produces almost double the usual amount of hypothetical games being played out depending on how many legal moves are available. With Diminished Monte Carlo all the extra computations relating to the second phase are ignored until the second phase is reached on the real board. This combined with the increased avoidance of the second phase in general discussed above results in a greatly decreased number of computations per game.

5.4 Attacking versus defending

Observing the behaviour of Basic Monte Carlo in figures 4.22 and 4.27, where Basic Monte Carlo is up against itself, there appears to be a steady increase in win rate for the starting player as the amount of cursors increases. Since in increase in cursors can be equated to an increase in strategic accuracy, this could infer that this version of Basic Monte Carlo has a greater aptitude for building towards winning shapes than it does for blocking the opponent's shape building attempts, as one might expect a stronger defence to extend the game and prevent a loss. The relatively low numbers for average moves shown in figures 4.22 and 4.27 prove that Basic Monte Carlo ends the game far earlier than its Weak Monte Carlo counterparts in the respective experiments (Figures 4.19 and 4.27).

5.5 Lack of infinite games

Whilst it has been noted in Section 2.2.4 that games of Six could technically play out with an infinite number of moves, it is worth noting that throughout all experiments performed for this paper, not a single game has reached a thousand moves. In fact, observing the average move charts in Sections 4.3, 4.4, and 4.5, it seems that even games exceeding 100 moves are extremely rare. It is imaginable that stronger algorithms that play more defensively could increase the amount of moves quite drastically.

Chapter 6

Conclusions and Further Research

This chapter summarises the conclusions of our various experiments, comparing the performance of our various AI players against one another. Observing the accumulated data within context as in Chapter 5, it becomes clear that, assuming all adaptations of the Monte Carlo approach are allowed to perform their individual loops more than once, the final hierarchy of performance is as such, from best to worst:

1. Basic Monte Carlo
2. Weak Monte Carlo
3. Random player

Furthermore the same data detailed in Section 5.1 clearly shows that there *is* a correlation that implies the starting player has a clear advantage over the second player (Fig. 5.1). The consistency of this increase in win rate over the various experiments confirms that this advantage exists beyond reasonable doubt, even if the exact strength of said advantage remains unclear. Determining the exact strength of the starting advantage is a problem left for further study.

Section 5.3 explains how the diminished adaptations of Monte Carlo are more efficient than the standard style of the Monte Carlo players that have been used throughout this paper. As Section 4.4 shows, the results of using this style seems to enhance the results rather than detriment them in any way, making the Diminished Monte Carlo approach an advantageous and attractive method.

Section 5.4 suggests that the Basic Monte Carlo algorithm used for these experiments is more adept at aggressive tactics than it is at defensive ones. For further research, one could research the effects of adjusting Basic Monte Carlo to value defensive moves as well as winning ones.

Finally, Section 5.5 notes that the experiments performed for this paper have rarely reached 100 moves, and never hit 100.

There are more opportunities for further research in the following:

- Explore different algorithm interactions in the Four-Player Variant (Section 2.1).
- Unleash more powerful algorithms upon SixExe, e.g., Monte Carlo Tree Search [BPW$^+$12].
- Explore the effect on the starting player's advantage when the starting player is given one tile less, or otherwise study the effect of altering the amount of tiles in the game.
- Research the various performance spikes such as those in Figures 4.16 and 4.17 with greater rigour.
- Research the cut-off and balance points of the use of adding more play-out loops or cursor moves by pitting different versions of Monte Carlo against each other giving their opponents a different amount of play-outs or cursor moves.
- Apply advanced heuristics to the various Monte Carlo approaches by e.g. adding partial negative values for draws or incorporating the length of the game (the amount of moves made) in the value of an outcome.
- Investigate different methods by differentiating shape-building victories from tile-capturing victories, and experiment with more targeted heuristics.
- Explore the tactics around choosing which groups to capture in the cases of groups of equal size.
- Research more defensive algorithms in conjunction with harder move limits, e.g. ending games that reach 100 moves as a draw.

Bibliography

[AHH10] Broderick Arneson, Ryan B. Hayward, and Philip Henderson. Monte Carlo Tree Search in Hex. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(4):251–258, 2010.

[BPW+12] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo Tree Search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[Bro00] Cameron Browne. *Hex Strategy: Making the right connections*. AK Peters Natick, MA, 2000.

[DDFG01] Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential Monte Carlo methods. In *Sequential Monte Carlo Methods in Practice*, pages 3–14. Springer, 2001.

[Gal79] David Gale. The game of Hex and the Brouwer fixed-point theorem. *The American Mathematical Monthly*, 86(10):818–827, 1979.

[gam] Versions of Six. BoardGameGeek LLC. `https://boardgamegeek.com/boardgame/20195/six/versions` (accessed: 10.10.2018).

[hex] The game of Hex. Edinburgh University. `https://www.maths.ed.ac.uk/~csangwin/hex/index.html` (accessed: 27.11.2018).

[Ste] Steffen Mühlhäuser. Steffen Spiele. `https://www.steffen-spiele.de/` (accessed: 10.10.2018).

[YLP01] Jing Yang, Simon Liao, and Mirek Pawlak. On a decomposition method for finding winning strategy in Hex game. In *Proceedings ADCOG: Internat. Conf. Application and Development of Computer Games*, pages 96–111, 2001.