Dark Hex: A Large Scale Imperfect Information Game

by

Mustafa Bedir Tapkan

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Imperfect information games model many large-scale real-world problems. Hex is the classic two-player zero-sum no-draw connection game where each player wants to join their two sides. Dark Hex is an imperfect information version of Hex in which each player sees only their own moves. Finding Nash equilibrium for Dark Hex problems is computationally challenging, as the number of possible strategies for each player is large: for example, for the 4×3 board, the number of pure strategies is around 10^{113}. In this work, we use a variety of methods to find improved strategies for 4×3 Dark Hex. To start, we discuss different versions of Dark Hex. After that we introduce a strategy for both players that improve the current results of the ϵ-Nash equilibrium of Dark Hex on the 4×3 board; previously it was known that the first player can win with a probability 0.112 and second player with 0.732 giving $\epsilon = 0.156$. We first improve these bounds with new handcrafted strategies to 0.1428 and 0.75 respectively in abstract game, $\epsilon = 0.107$. We then further improve the strategies using a player, trained by Monte Carlo Counterfactual Regret Minimization using probability one pruning and *imperfect recall* abstraction. We introduce a new simplification approach where we limit the actions based on their probabilities, the new strategies introduced using simplification over the MCCFR strategies give us an improved $\epsilon = 0.01$ in the abstract game. We then introduce probability smoothing with extra parameters. The final results give use 0.791 and 0.205 for players respectively, giving us $\epsilon = 0.002$, in the abstract game. We discuss the differences between handcrafted and computer-generated strategies.

Lastly, we compare players who were trained using Neural Fictitious Self-Play and Monte Carlo Counterfactual Regret Minimization in different abstraction and pruning settings, we fashion a round-robin tournament and report the results; comparing the strength of all the trained players.

To my grandpa

For teaching me to enjoy learning.

A wizard is never late, nor is he early, he arrives precisely when he means to.

– Gandalf (Lord of the Rings, J.R.R. Tolkien)

Acknowledgements

I started my journey in Computer Science years back on a freshman table. After many discussions and long nights, I have come to the University of Alberta to delve further into the things I am most interested in. Now concluding another endeavour in the form of this thesis I have many to thank for reaching here.

First I would like to tell you how much I appreciate having such supportive supervisors, Ryan Hayward and Martin Mueller. They have provided me with everything they could and further extended their friendships. Thank you for being like second parents to me on this long and rocky road. They are the main reason I have been able to reach the end of this passage.

Having experts on the topic you are trying to research around you is one of the most crucial help. I was really lucky to have such amazing people on my reach; Marc Lanctot and Dustin Morill. I am grateful that I have met these resourceful people who have given me direction whenever I needed it, they have guided me in the best way possible, I could not have wished for better mentors.

These past years have had hard things to deal with; getting used to the Edmonton colds and rough years of COVID. I was lucky enough to have many great friendships that kept me sane during these tough times and made me love this city. They were there when I got sick, failed to defend, or just didn't have enough energy to continue what I did. Having your friendship has been the greatest gift in the past years; Nikoo Aghaei, Kiarash Aghakasiri, Hamza Emra, Amirmohsen Sattarifard, Parnian Mehinrad, Liam Peet-Pare, Alexis Arrigoni, Tom Pinder, Katie Burak, Mahtab Faroukh and Farnaz Kohankhaki. You made every part of my time in Edmonton exceptional.

Support from a distance is not an easy task by any means, when I am in the most troubling situations my family was always there for me. Regardless of their problems they have provided me with eyes and ears to bother on every occasion. Especially my sister Nilgun Tapkan and my brother Tarik Bergstrom, who encouraged me to learn and do more at every corner. My family was not the only distant support I had, many friends along the way gave me fortitude, some of them, in no particular order, are Mikhail Mekhedkhin-Meskhi, Natalie Novak-Olszewski, Hediye Rostami and Ali Kaya.

I owe everything I have today to my teachers along the way. There have been many, but I would like to extend my gratitude towards two in particular: Ekrem Uzun, who thought me to love learning, and do what I love, and Kemal Aydin, who became a lifelong mentor and a valuable friend, and thought me to be organized and dedicated in life. I appreciate every minute you spent with me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The quest to develop machines that can behave in an intelligent manner has captured the imagination of researchers for decades. Research into modern artificial intelligence (AI) traces back to Alan Turing in the mid 1900s. In recent years there has been a surge in interest in AI and, in particular machine learning, as deep learning algorithms proved to be capable of solving problems previously too challenging for machines. The question of what constitutes intelligence is a subject of philosophical debate, and it has been defined in different ways by different people. Rich Sutton, building on John McCarthy's definition of intelligence, states that *"Intelligence is the computational part of the ability to achieve goals in the world"*, and, further, that *"A goal achieving system is one that is more usefully understood in terms of outcomes than in terms of mechanisms."* [1], [2]. The last century has seen enormous progress towards this goal of understanding, and building, the computational ability to achieve goals in the world, and has led to some of the most exciting technological developments in the world today [3]–[9].

The AI research community contains competing paradigms and conceptualizations of what it means for a machine to be intelligent and, more importantly, the best way to achieve this intelligence. One perspective, which has proven useful in reinforcement learning (RL), is to consider an *agent* who acts in an *environment* and makes decisions based on the dynamics of this environment, possibly in the presence of other *agents*. Ultimately we would like to build an agent that can effectively make decisions in the real world, but due to the complexity of real world dynamics, it is often beneficial to study agents in more simple and controlled environments. Games provide an ideal setting for studying intelligence as their well-defined rules and simple dynamics allow for systematic and incremental research. Further, the ability to perfectly simulate games allows us to leverage the computational power of modern computers to train agents in ways that are not possible if we rely on the real world as our environment. It is for these reasons that games, both board games and video games, have played an enormously important role in AI research.

Game theory, developed in its modern form in the 1950s by John Nash, John Von Neumann and others, studies interactions between agents with the use of mathematical models. While not intrinsically connected to research in AI, game theory provides a useful framework for understanding dynamics within games with well defined rules. Within game theory there are a variety of types of games, each with their own distinct properties. Since this thesis focuses on the game of Dark Hex, we restrict our attention to two-player extensive form zero sum games.

Such games can be categorized based on the information the agents have: *perfect-information* and *imperfect-information* games. In perfect information games, a player has complete knowledge of the game state before making a decision. In imperfect information games, the agents have partial information about the environment or the other agents. Chess is a popular example of a perfect information game where both players have perfect knowledge of the board and the possible moves for the opponent. On the other hand, Poker is an imperfect information game where a player only has partial information about the opponent's hand. Research on imperfect information games is more challenging due to this partial knowledge. In this thesis we focus on an imperfect information version of the game Hex.

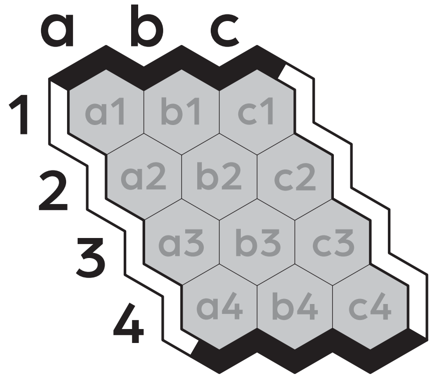


Figure 1.1: An empty Hex board.

Hex is a game with a rich but relatively short history. It was invented by Piet Hein in 1942 [10], and was made popular by the famous mathematician John Nash. Hex is a two-player zero-sum perfect information game with no draw [11]. The game is played on an $n \times m$ board with hexagonal cells where each player tries to connect their sides of the board (east to west or north to south) before the opponent does so. The black player moves first, and players take turns placing a stone of their colour on the board until one player achieves an uninterrupted connection.

Dark Hex is an imperfect information version of the game Hex. It is played on two identical Hex boards. Players sit in a fashion that neither can see their opponent's board. An umpire knows both players' boards. When a player wants to make a move, the umpire lets them place a stone or tells them if the cell is occupied, in which case player chooses another cell to play. Games of this kind are commonly called *phantom games*, *blind games* or as in chess *kriegspiel*. This family of games is, in particular, hard to deal with due to its immense number of games possible. Therefore there is limited research on such games. Even tiny boards have massive complexity. For a 4x3 hex board the number of strategies is roughly 1.506×10^{113} [12].

A *Nash Equilibrium* is an equilibrium value associated with player strategies where no player can benefit from deviating from their strategy unilaterally. Finding a Nash Equilibrium in a game means that both players did the best they can do (in zero-sum two-player games), considering the other player is not changing their *strategy*. However, finding the exact Nash Equilibrium is usually unfeasible for large games because of its need to traverse every game possible. Nevertheless, a significant amount of work was put in to either approximate or calculate the Nash Equilibria for games such as Poker and Liars Dice [13], [14].

An ϵ-Nash Equilibrium is an approximate Nash Equilibrium. The measure ϵ represents how distant the given strategy is from a Nash Equilibrium strategy in terms of value. A *best response* strategy is a strategy tailored to win as much as possible against a specific strategy. We can measure the success of a strategy by finding the expected win rate against its best response strategy. This measure is called the *best response value*.

Research in games

Research in games has been a common goal and interest of the AI community for many years [15], [16]. Most of this effort has been directed to perfect information games [4], [17], [18]. Research on imperfect information games was elevated after the development of sequence-form linear programming [19] in 1994.

Poker has been the most popular family of games in this domain. Kuhn Poker [20] and similar variants [21], [22] have made it possible to test ideas on smaller games in the same family. Around the year 2000 computer Poker research became the interest of many researchers [23]–[28]. The University of Alberta has developed poker AIs using the sequence-form LP mentioned before. Many competitions and poker programs were developed after 2000. Competitions with some of the world's strongest human players in two-player poker games were held where the AI beat all the players in the end [29], [30]. A regret-based algorithm called Counterfactual Regret Minimization was the main technical achievement [31]. Many variants and descendant algorithms further improved the CFR algorithm [32]–[34]. In 2016 a CFR-based algorithm beat a group of professional Poker players in the full-size No-Limit Texas Hold'em Poker game [3]. In recent years, the popularity of *deep learning* has also carried on to CFR algorithms [34]–[36].

Abstraction and a Strong Dark Hex Player

For large games, a common approach is to use an *abstraction* of the game that holds its basic structure in only a fraction of the size. A solution to this new abstraction then gets translated back to the original version. If the abstraction inherits the strategical properties of the game, the newfound strategy performs well in the original game as well. There is an extensive research on this traditional approach [28], [37]–[40]. This method can be flawed since there are cases where abstractions that perform better against average play are more exploitable [37].

One property, in particular, makes abstraction in imperfect information games harder: The abstraction might never reach some states, and not make use of the actions that are needed in order to reach the equilibrium point. A strategy that was never encountered might be necessary to beat another player's strategy. Therefore one would need extensive game-specific knowledge to craft a well-designed abstraction of the game.

We use two different abstraction techniques in this work. The first one is called imperfect recall [41]. We ignore the history of the game and only use the current board state as the information state. This reduces the memory requirements of any algorithm by orders of magnitude. We also use a neural network embedded in Neural Fictitious Self Play algorithm, which is the second abstraction.

Contributions

The main contributions of this thesis are:

- **A precise description and analysis of the game Dark Hex and its variants.** The game has found little attention in modern research so far. We introduce the rules and different versions of the game alongside some terminology.

- **New approximate bounds for each player in 4×3 Dark Hex.** We introduce three new, successively stronger strategies for both players on the 4×3 board that improve the best known bounds in approximation. We first develop stronger strategies by hand, then introduce a better strategy generated by computer using an end-to-end learning approach. The first player bound is improved from 0.112 to 0.125 and then to 0.205. The second player's is improved from 0.732 to 0.75, then to 0.786. These new bounds get us to 0.99 of the Nash equilibrium. In a final step, We introduce a technique to smooth out the learned probabilities, which improves the first player bound to 0.793; overall we get an ϵ-Nash equilibrium strategy with $\epsilon = (1-0.205-0.793) = 0.002$ for both players in approximation.

- **A database of sure win states for both players on small boards.** Pruning the state space is extremely helpful for games with high branching factors. Following prior work on determining sure win states [12], we generate a database of all sure win states up to 4×3 board size and make it publicly available at our GitHub repository (github.com/BedirT/darkhex. We make use of this database in our Dark Hex player experiments, and show how effective they are.

- **Introduction of an Imperfect Recall Dark Hex Model.** We introduce an imperfect recall representation of Dark Hex, and evaluate it in different settings against the perfect recall, showing how effective the abstraction is, reducing the game to $\tilde{0}.001$ of its original size.

- **New players and empirical results on their performance.** We provide the first Dark Hex players in the literature which use the techniques of Neural Fictitious Self-Play and Monte Carlo Counterfactual Regret Minimization. We develop these agents under different pruning and abstraction settings and compare their strength on the 4×3 board. We show that our final player SIMCAP+ shows a huge success while again our player SIMCAP takes the second spot. Other than our methods, we show that MCCFR performs much better compared to all the NFSP methods. Also Imperfect Recall methods perform a lot better than perfect recall.

Chapter 2

Background and Related Work

In this chapter, we discuss background material in game theory and machine learning that is needed to read the rest of the thesis.

In Section 2.1 we introduce the basics of game theory, the terminology and explain the games that we use throughout the thesis. In Section 2.2 we introduce techniques for regret minimization: Counterfactual Regret Minimization (CFR) and Monte Carlo Counterfactual Regret Minimization (MCCFR). We also discuss the sampling method we use, Outcome Sampling. In Sections 2.3 and 2.4 we give short reviews of Reinforcement Learning (RL) and Neural Fictitious Self-Play (NFSP). Lastly, in Section 2.5 we explain the imperfect recall abstraction technique.

2.1 Game Theory

Here we give the foundations of game theory needed for the rest of the thesis. This thesis is about two-player finite games. Nonetheless, most of the concepts here apply to multiple players.

A **strategy** is a set of probability distributions for each possible position in a game for a player. A **normal-form game** is a game where all players decide on a strategy simultaneously, without knowing the opponent's strategy. Each player receives a payoff (reward, *utility*), stating how good or bad the chosen strategy was against the opponent's chosen strategy.

In a *bi-matrix game*, a table (matrix) states the payoffs for each player, given the strategy used by both. Traditionally, the first player is the *row* *player* and the second player is the *column player*.

In a **zero-sum game** the players' utilities sum to zero, meaning that if a player has utility u, then the opponent has utility $-u$.

Example: Table 2.1 shows an *Rock-Paper-Scissors (RPS)* game payoff table. This is a zero-sum game. It is common to show only the payoff matrix for the row player in zero-sum games as in Table 2.2.

	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0

Table 2.1: Rock-Paper-Scissors payoff table. Players pick a move simultaneously and get their corresponding payoff. The game is zero-sum.

	R	P	S
R	0	-1	1
P	1	0	-1
S	-1	1	0

Table 2.2: A simplified zero-sum payoff table shows the payoff for the row player only. The negated value is the payoff for the column player.

In a **pure strategy** the player chooses a single action with probability 1 in each state. An example in RPS, is playing Rock all the time.

In zero-sum games the minimax equilibrium is equivalent to the Nash equilibrium and was introduced by John Von Neumann in 1928 **??**, but since the majority of the researchers on this topic uses Nash equilibrium we also continue using that.

A **pure Nash equilibrium** is a pair of pure strategies such that no player benefits from diverging from it. We call strategies that form a Nash equilibrium **Nash strategies**. Define S_i as the set of possible strategies for player i. Let $s_i \in S_i$ be a pure strategy for player i. $s = (s_1, s_2)$ is a **pure strategy profile**, and $u_i(s)$ is the **utility** for player i when players follow s. s is a pure Nash equilibrium iff

$$\forall s_1^{'} \in S_1, \forall s_2^{'} \in S_2 : u_1(s) \geq u_1(s_1^{'}, s_2) \text{ and } u_2(s) \geq u_2(s_1, s_2^{'}) \tag{2.1}$$

Example 2: Prisoner's Dilemma Officials have arrested two people for a crime. There is, however, very little evidence against them. Both are taken to different isolated cells and asked to betray the other. If only one person betrays, there will be no charges against the one who betrays, and the other person will go to prison for three years. If both betray, they will go to prison for two years each, and if neither betrays, they can only keep them in for one year. The matrix is shown in Table 2.3. This is not a zero-sum game.

		Prisoner 2	
		Betray	Silent
Prisoner 1	Betray	1,1	0,3
	Silent	3,0	2,2

Table 2.3: Prisoner's dilemma.

At first glance the best overall outcome for the group is for both prisoners to keep silent and get one year each. However, from the individual's point of view, betraying is more beneficial regardless of what the other prisoner does. For example for prisoner 1, if prisoner 2 decides to keep silent, it is more beneficial to choose to betray since he will go free right away. If prisoner 2 decides to betray, it is again more beneficial for prisoner 1 to betray and get the sentence of two years instead of three. So if both prisoners think from their perspective only, they reach a sub-optimal point for the group and themselves. The Nash equilibrium in this example is for both players to betray.

There is no pure Nash equilibrium in RPS. A player should not choose the same move all the time, considering that the other player can exploit this strategy. A **mixed strategy** is a probability distribution over all possible pure strategies. Let us denote the mixed strategies for player i as Σ_i, and a combination of mixed strategies for both players as $\Sigma = \Sigma_1 \times \Sigma_2$. A **strategy profile** is $\sigma = (\sigma_1, \sigma_2)$ where $\sigma_i \in \Sigma_i$. So, σ_i is a probability distribution over strategies in S_i. A strategy profile is a Nash equilibrium if no player i benefits from changing this strategy unilaterally. A Nash equilibrium is defined using **expected payoffs**, $u_i(\sigma)$. Formally,

$$u_i(\sigma) = u_i(\sigma_1, \sigma_2) = \mathbb{E}_\sigma[u_i(s)] = \sum_{s_1 \in S_1} \sum_{s_2 \in S_2} \sigma_1(s_1)\sigma_2(s_2)u_i(s_1, s_2) \tag{2.2}$$

For denoting the strategies of all opponents of player i in σ, we use σ_{-i}.

An **ϵ-Nash equilibrium** is a generalization of the Nash equilibrium discussed above, where players cannot gain more than ϵ by changing their strategy unilaterally. Formally, σ is an ϵ-Nash equilibrium iff

$$\forall i \forall \sigma_i^{'} \in \Sigma_i : u_i(\sigma_i^{'}, \sigma_{-i}) - u_i(\sigma) \leq \epsilon \tag{2.3}$$

Equation 2.1 is the special case of Equation 2.3 with $\epsilon = 0$.

A **best response strategy** $BR(\sigma_{-i})$ is a strategy where player i achieves the greatest possible payoff against a given opponent strategy σ_{-i}. Formally $\sigma_i \in BR(\sigma_{-i})$ iff

$$\forall \sigma_i^{'} \in \Sigma_i : u_i(\sigma_i, \sigma_{-i}) \geq u_i(\sigma_i^{'}, \sigma_{-i}) \tag{2.4}$$

Any Nash strategy is interchangeable with any other Nash strategy in a zero-sum two-player game. So for Nash equilibrium profiles $\sigma = (\sigma_1, \sigma_2)$ and $\sigma^{'} = (\sigma_1^{'}, \sigma_2^{'})$, $(\sigma_1, \sigma_2^{'})$ and $(\sigma_1^{'}, \sigma_2)$ are also Nash equilibrium profiles. All the expected utilities of Nash strategies for a given player are the same, i.e. $u_i(\sigma) = u_i(\sigma^{'})$.

The expected utility of a player is also called the **value** of the game for that player. From now on, we will use the term value for the first player's expected payoff unless otherwise specified. The second player will receive the negated value.

Exploitability is the distance of a strategy from the equilibrium value when opponent is playing the best response strategy. If player i changes their strategy from a Nash equilibrium to the new strategy with value $v - \epsilon_i$ where v is the equilibrium value, it is exploitable by $\epsilon_i \geq 0$. The opponent can exploit, and get the value $-v + \epsilon_i$. Formally, if σ is a Nash equilibrium profile where $\sigma = (\sigma_1, \sigma_2)$, and $\sigma_1^{'}$ is an updated strategy for player 1 that is exploitable by ϵ_1, then

$$\exists \sigma_2^{'} \in \Sigma_2 : u_1(\sigma_1^{'}, \sigma_2^{'}) = \min_{\sigma_2^{''} \in \Sigma_2} u_1(\sigma_1^{'}, \sigma_2^{''}) = v - \epsilon_1 \tag{2.5}$$

The sum of the exploitability of the two players is a measure of the distance of a strategy profile to the Nash equilibrium; $\epsilon_\sigma = \epsilon_1 + \epsilon_2$.

An **extensive form game** is a game where the players take sequences of actions in a single game. Unlike normal-form games, each move results in a new state until reaching the terminal state. A tree structure represents all possible plays of this type of game. The root of the tree represents the starting state of the game. Each node represents a game state, and each edge represents an action taken in the parent node state, resulting in the child node state. Every leaf node represents a terminal state with a payoff for the root player. Figure 2.1 shows an example of an extensive-form game tree.

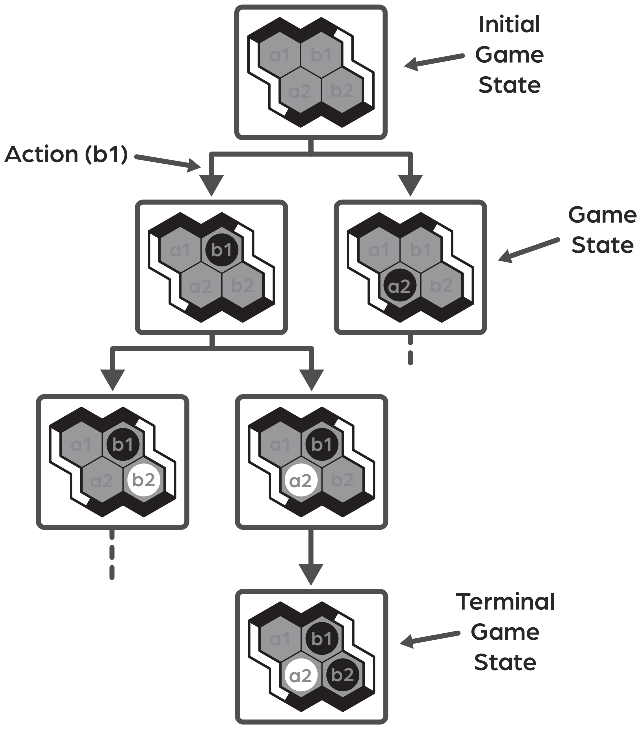


Figure 2.1: Representing an extensive form game as a tree. The root is the initial game state, an edge represents an action by one player and nodes are game states. Leaf nodes are terminal states of the game.

A **history** is a sequence formed by the actions of all the players starting from the initial game state. Let H be the set of possible game histories. A history $h \in H$ is a **terminal history** if it reaches a terminal state. A **successor** of history h is denoted as ha where the h is extended with action a. A **player function** $P(h)$ determines the player to play after h.

An **information state** is a partial game state information a player has at a given point in an imperfect information game. An information state does not necessarily represent the full game state since the players may have private information. An **information set** is the set of all histories that end up in the same information state. Figure 2.2 shows an information set $I = \{h_1, h_2\}$ where $h_1 = a$ and $h_2 = bc$ for actions a, b, c.

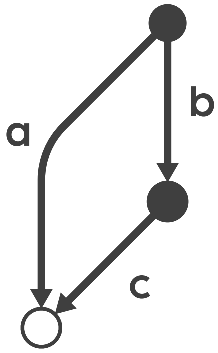


Figure 2.2: Information set tree representation with multiple histories. Both paths end up in the same information state, but the actions are different.

2.1.1 Hex

Hex is a two-player zero-sum perfect-information extensive-form game. It is newer than games such as Go and Chess. With the recent developments in Computer Games [6], [7], [17], [42] hex also drew the renewed interest of researchers [18], [43]–[46]. We describe the rules and give some information about the game.

Hex is played on a board with hexagonal cells. Each player is assigned two opposing sides of the board. Black tries to connect the top and bottom sides, while white connects the left and right sides. A connection is an uninterrupted chain of stones in a single colour. Players take turns placing their stones on the board starting with black. A player can place one stone of their own color on any empty cell when it is their turn. Figure 2.3 shows a hex board where Black has won by connecting its sides.

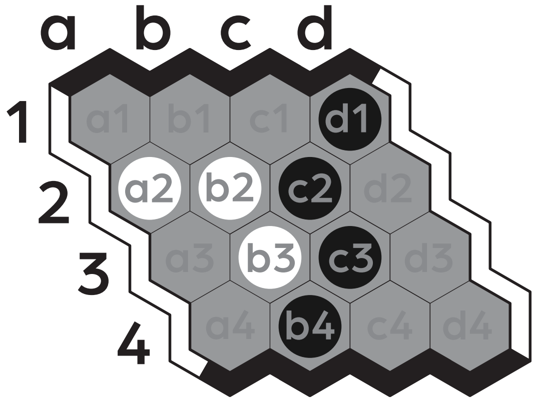


Figure 2.3: A 4x4 hex game. The cells are labeled with letters representing the (diagonal) columns and numbers representing the rows.

Hex is a no-draw game. One player will win no matter how the players proceed [10]. An extra stone is never harmful to a player [10]. The first player has a winning strategy in hex on any n×n board as proved by John Nash [10]. In tournaments and casual games, it is common to use 11×11 or 13×13 boards.

2.1.2 Dark Hex

Dark hex is an imperfect information version of the game hex. All the rules of hex apply except that the players have incomplete information about the game state.

Players have no knowledge of the location of the opponent's stones unless they have discovered them by trying to make a move on an occupied location. The referee in the game sees the full game state and informs the player if a move tried is legal or not. The detailed rules about the information leaked to the other player during these tries depend on the game version. We discuss several of these versions in Chapter 3. In this thesis, we mainly focus on the rules where the player who attempted a move tries again until the move is successful, and opponent is not informed of the failed tries. Figure 2.4 presents a sample game on a 3×3 board.

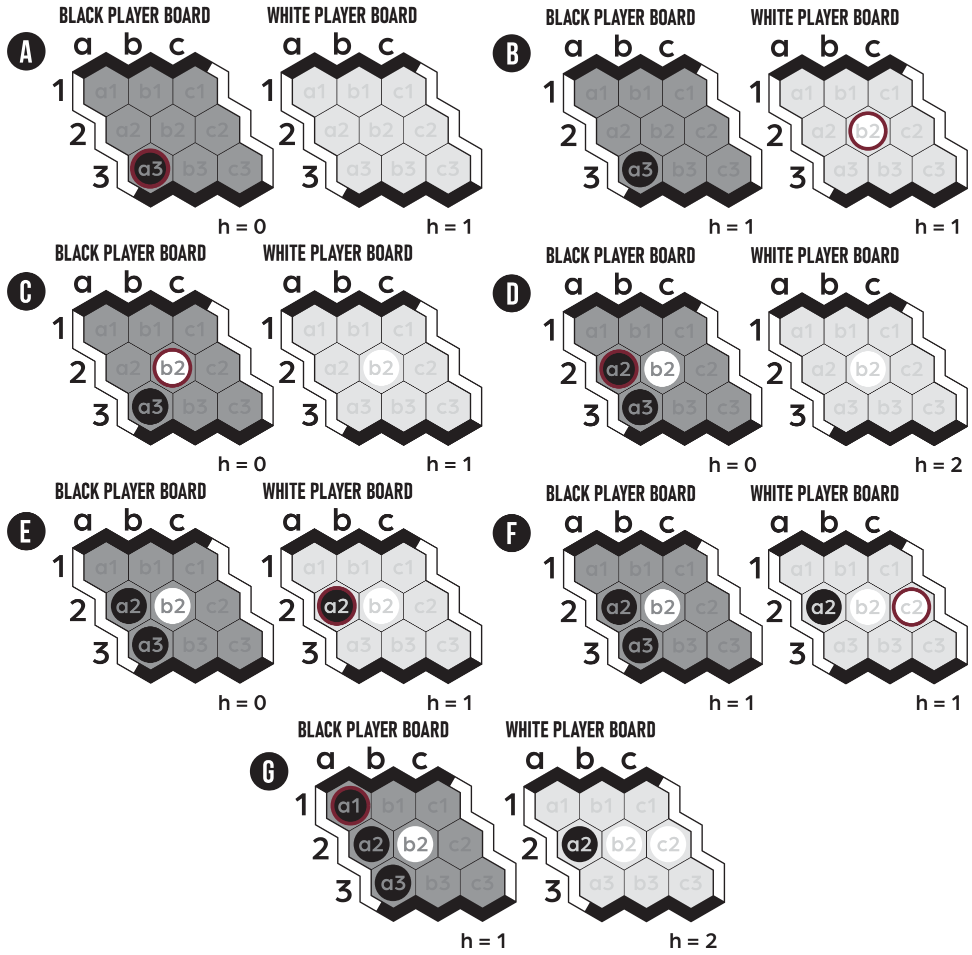


Figure 2.4: An example game of Dark Hex where black wins on a 3×3 board. **A)** Black plays on $a3$. The white player's board is not updated, but the number of hidden stones h is increased. **B)** White plays in the center updating their own board only. **C)** Black tries to play in the center as well, resulting in failure. Black now knows that there is an opponent stone on $b2$. **D)** Black makes a different move, which succeeds. **E)** White tries $a2$. **F)** White plays $b1$. **G)** Black plays on $a1$ and wins the game.

2.2 Counterfactual Regret Minimization

A way to compute Nash equlibria is to learn a Nash strategy iteratively over time, improving the strategy step by step by calculating the ϵ-Nash equilibria, and modifying to improve the ϵ, such that $\epsilon \to 0$. A method to approximate Nash equilibria is Counterfactual Regret Minimization (CFR).

CFR is the current baseline algorithm that is widely used for playing imperfect information games. CFR uses the regret minimization theory proposed by Hannan in 1957 [47] with regret matching by Heart and Mas-Collell [48]. We explain Regret Minimization in Section 2.2.1, CFR in Section 2.2.2 and MCCFR, a Monte Carlo variant of CFR, in Section 2.2.3. In Section 2.2.4 we introduce outcome sampling, the sampling method we use for MCCFR.

2.2.1 Regret and Regret Minimization

Regret is a measure of success for a given state action. At a decision point, a player takes action $a \in \mathcal{A}$ and receives reward r. In a learning environment, a players' goal is to maximize their total reward. This is an **online learning** problem since the rewards are received only after the actions are taken. The players decide between **exploration** or **exploitation**, either trying out a less explored action or choosing to exploit the action with the highest estimated reward.

Assume an agent making decisions at some state for every time step $t \in T$ and receiving a utility u^t for the corresponding time step, $\{u^1, u^2, \ldots, u^T\}$. Let u^t_a be the utility of choosing action a at time step t. We define the expected **regret** of not taking action a at all time steps as

$$R^T(a) = \mathbb{E}\left[\sum_{t \in T}(u^t_a - u^t)\right] \tag{2.6}$$

External regret is a way to measure how good an agent is doing. For player decisions $\mathcal{N}$ the accumulated external regret is

$$R^T = \max_{i \in \mathcal{N}} R^T(a) = \max_{i \in \mathcal{N}} \mathbb{E}\left[\sum_{t \in T}(u^t_a - u^t)\right] \tag{2.7}$$

An algorithm is considered a **regret minimization** algorithm if the average regret $\bar{R}^T = R^T/T$ approaches 0 over time. Using regret minimization, an algorithm will eventually approach the minimax value. If the players play T times against each other the average reward for player i will at least be $v_i - \bar{R}^T$, where v_i is the minimax value for player [49].

If $\bar{R}^T < \epsilon$ for both players, then the average strategy profile is a 2ϵ-equilibrium [49]. Regret-based algorithms use this idea to approximate the Nash equilibria.

2.2.2 Sequential Games and CFR

So far we only considered non-sequential games. In this section we focus on sequential games by extending the regret minimization framework. CFR makes use of the Regret Matching Algorithm. Since there are dependencies in sequential games (a move leads to the next state of the game) we have to consider:

- The probabilities of reaching each information set
- In a forward pass, updating the game state and probabilities of player action sequences
- In a backward pass, computing the utilities of the information sets

We first define the necessary components to examine and understand the regret matching and counterfactual regret formulations. We then summarize CFR mathematically and finish the section with the pseudo-code.

Formula components:

- $\boldsymbol{h}$**:** A history, the action sequence from the initial game state to some state s.
- $\boldsymbol{I}$**:** An information set, the set of all the histories leading to a certain state.
- $\boldsymbol{t}$**:** A time step where $0 \leq t \leq T$, where T is the time limit.
- $\boldsymbol{\sigma_i^t}$**:** A strategy, a mapping for player i from a legal action $a \in \mathcal{A}(I_i)$ in information set I_i to the probability of taking that action at time t.
- $\boldsymbol{\sigma^t}$**:** A strategy profile, a probability distribution over all possible actions for all players at time t.
- $\boldsymbol{\sigma_{-i}^t}$**:** A strategy profile that excludes player i's strategy.
- $\boldsymbol{\sigma_{I \rightarrow a}}$**:** A probability distribution equal to σ, except action a is always chosen for information set I.

- $\boldsymbol{\pi^\sigma(h)}$**:** A reach probability, the probability of reaching h when following σ.

- $\boldsymbol{\pi^\sigma(I)}$**:** A reach probability, the probability of reaching information set I through any possible history $h \in I$, $\pi^\sigma(I) = \sum_{h \in I} \pi^\sigma(h)$.

- $\boldsymbol{\pi^\sigma_{-i}(I)}$**:** The counterfactual reach probability for information set I, the probability of reaching I when player i always takes actions to reach the information set I.

- $\boldsymbol{Z}$**:** The set of all terminal game histories.

- $\boldsymbol{z}$**:** A terminal history, $h \sqsubset z$ for $\boldsymbol{z \in Z}$.

- $\boldsymbol{u_i(z)}$**:** The utility of terminal history z for player i.

The **counterfactual value** $v_i(\sigma, h)$ for player i given strategy σ at non-terminal history h is defined as

$$v_i(\sigma, h) = \sum_{z \in Z, h \sqsubset z} \pi^\sigma_{-i}(h) \pi^\sigma(h, z) u_i(z) \tag{2.8}$$

The *counterfactual regret* for not taking action a in history h is

$$r(h, a) = v_i(\sigma_{h \to a}, h) - v_i(\sigma, h) \tag{2.9}$$

The counterfactual regret of not taking action a in the information set I is

$$r(I, a) = \sum_{h \in I} r(h, a) \tag{2.10}$$

$r^t_i(I, a)$ is the regret when players other than i move according to σ^t on information set I. The cumulative counterfactual regret is

$$R^T_i(I, a) = \sum_{t=0}^{T} r^t_i(I, a) \tag{2.11}$$

We define $R^{T,+}_i(I, a)$ as $\max\Big(R^T_i(I, a), 0\Big)$. The regret is the difference between the value of always choosing action a at I and using σ for the action selection. We weigh the values by the other players' probability of reaching a node. Heart and Mas-Colell's regret matching [48] applied to cumulative regrets leads to the updated strategy

$$\sigma_i^{T+1}(I,a) = \begin{cases} \frac{R_i^{T,+}(I,a)}{\sum_{a\in\mathcal{A}(I)} R_i^{T,+}(I,a)}, & \text{if } \sum_{a\in\mathcal{A}(I)} R_i^{T,+}(I,a). \\ \frac{1}{\mathcal{A}(I)}, & \text{otherwise.} \end{cases} \tag{2.12}$$

This iterative method is computationally heavy since we need to traverse every action. This makes it impossible to use vanilla CFR on games with practical sizes. Monte Carlo CFR (MCCFR) in Section 2.2.3, is a more scalable and efficient variant of CFR.

2.2.3 Monte Carlo Counterfactual Regret Minimization

Monte Carlo CFR is a sample-based version of the CFR algorithm in Section 2.2.2. Instead of finding the true counterfactual regret values, MCCFR computes unbiased estimates of them. This gives an anytime algorithm that can use as many samples as wanted. In expectation, it still converges to the same value as the true value. The sampling method plays an important role in MCCFR. In this work, we use outcome sampling (described in Section 2.2.4).

Let us define $\mathcal{Q} = \{Q_1, Q_2, ...\}$ as the subset of all terminal histories, Z, where $\bigcup_{Q\in\mathcal{Q}} Q = Z$. Q_j is a block of terminal histories where $Q_j \subseteq Z$. The difference of MCCFR from CFR comes from the samplings of these blocks. For each time step t, the probability of sampling Q_j is $q_j > 0$. Then the probability of any terminal history z is contained in a given block for all information sets I such that $z \in Q_j, h \sqsubset z, h \in I$, is:

$$q^t(z) = \sum_{j:z\in Q_j^t} q_j^t \tag{2.13}$$

The sampled counterfactual value $\bar{v}$ is:

$$\bar{v}_i(\sigma, I|j) = \sum_{z\in \boldsymbol{Q_j}\cap Z_I} \frac{\mathbf{1}}{\boldsymbol{q^t(z)}} \pi_{-i}^{\sigma}(h)\pi^{\sigma}(h,z)u_i(z) \tag{2.14}$$

If one guarantees $q^t(z) \geq \delta > 0$ then the sampled regret is well-defined.

The bolded parts are the only difference between v and $\bar{v}$ in CFR (equation 2.8). The expected value of $\bar{v}_i(\sigma, I|j)$ is equal to $v_i(\sigma, I)$ [50].

The main question when using MCCFR is how to pick a good sampling scheme. Sampling affects both theoretical guarantees and empirical performance. As an example, if $\mathcal{Q} = \{Z\}$ the MCCFR is the same as vanilla CFR. In this thesis, we use outcome sampling (Section 2.2.4).

Algorithm 1: Monte Carlo Counterfactual Regret Minimization

```
Require: a sampling scheme $S$
Initialize:
    regret tables: $\forall I \in \mathcal{I}, \forall a \in A(I) : r_I[a] \leftarrow 0$
    cumulative strategy tables: $\forall I \in \mathcal{I}, \forall a \in A(I) : s_I[a] \leftarrow 0$
    initial profile: $\forall I \in \mathcal{I}, \forall a \in A(I) : \sigma(I, a) \leftarrow 1/|A(I)|$
for $t \in \{1, 2, 3, ...\}$ do
    for $i \in N$ do
        Sample a block $Q$ using $S$
        for $h$ where $h \sqsubset z, h \in I, z \in Q, P(h) = i$ do
            $\sigma_i(I) \leftarrow RegretMatching(\mathrm{r}_i)$ using Eq. 2.12
            for $a \in A(I)$ do
                Let $\bar{r} \leftarrow \bar{r}(I, a)$ sampled regret for $a$
                $r_I[a] \leftarrow r_I[a] + \bar{r}$
                $s_I[a] \leftarrow s_I[a]+$ AverageStrategyIncrement$(s_I, t, \sigma_i, I, a)$
```

"AverageStrategyIncrement" in the algorithm refers to updating the average strategy. This update is a simple summation for a given information state, starting from 0 (just like the strategy tables). The time complexity of the MCCFR algorithm depends on the sampling method used to define Q. The space complexity is $O(|C_1| + |C_2|)$ where $C_i = \{(I, a)|I \in \mathcal{I}_i, a \in A(I)\}$, since any sampling method needs to keep all the information sets for both players in memory.

2.2.4 Outcome Sampling

Outcome sampling [50] chooses $\mathcal{Q}$ such that each block only contains a single terminal history z. In each iteration of MCCFR, outcome sampling updates only the values for the information sets along with the history z. As long as the sampling probability distribution σ has some positive probability $p > 0$

for each pair (I, a), Equation 2.14 is well-defined.

Outcome Sampling MCCFR (OS-MCCFR) works as follows:

1. Sample a terminal history z using policy σ.

2. Compute every reach probability for every history $h \sqsubset z$ by a forward traversal.

3. Backward traverse from z and compute the players' probabilities of playing every action other than the action to reach z.

4. Compute the counterfactual regret for z and add it to the total regret.

We use on-policy ϵ-greedy exploration. With probability ϵ, the algorithm samples an action uniformly at random, and with probability $1 - \epsilon$, it follows the players policy. Since sampling is for a single terminal history, this results in *linear time complexity* in the depth of the game tree for each iteration. [50] for further information on sampling methods.

2.3 Reinforcement Learning

We use Reinforcement Learning (RL) with the algorithm Neural Fictitious Self-Play (NFSP). Here we introduce the basics of RL before explaining NFSP.

RL is a branch of machine learning where players try to maximize their expected reward by interacting with the environment. The environment is modelled as a **Markov decision process (MDP)**. The individual players are called **agents**. An **environment** surrounds an agent. A **state** is a representation of the environment that the agent receives. Agents behave according to a probability distribution over legal actions, called a **policy**, at each state. Agents try to improve their policies by maximizing their total reward over time. For time step t, the total reward is $R_t = \sum_{i=t}^{T} r_{i+1}$, where T is the time step where game terminates, and r_i is the reward received at time step i. Checkout Figure 2.5 for a visual representation of an RL system.

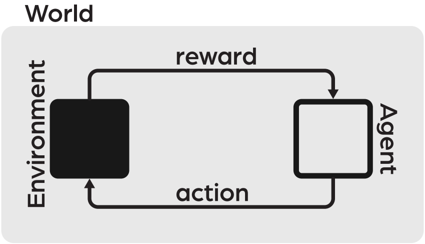


Figure 2.5: The dynamics of a reinforcement learning system.

Experience can be represented by a **transition** tuple, as $(s_t, a_t, r_{t+1}, s_{t+1})$. At state s_t, the agent takes the action a_t, receives the reward of r_{t+1} and transitions to the next state s_{t+1}.

Learning an **action-value function** $Q(s, a)$ is a common goal in reinforcement learning. It represents the value to take action a at time step t and following the policy π afterwards, $Q(s, a) = \mathbb{E}_\pi[R_t|S_t = s, A_t = a]$. In **On-policy** learning, the agent learns about the policy it follows, while in the **off-policy** case the agent learns about a different policy.

Q-learning is one of the most popular off-policy methods [51], which learns about the greedy policy. A Deep Q Network (DQN) [52] utilizes neural networks in an online fashion for Q-learning.

2.4 Neural Fictitious Self-Play

Neural Fictitious Self-Play (NFSP)[53] is a method we use to develop Dark Hex players. NFSP is a scalable end-to-end reinforcement learning approach for learning an approximate Nash Equilibrium strategy with no prior knowledge about the game.

Fictitious Self-Play(FSP) agents learn from interacting with each other. Each agent has two different memories. $\mathcal{M}_{RL}$ stores the transition tuples generated while playing against each other. $\mathcal{M}_{SL}$ stores the agent's own behaviour. The self-play sampling is set up so that the $\mathcal{M}_{RL}$ memory for an agent approximates the data of an MDP that is generated by the average behaviour of the other agents. Therefore a learned solution gives us an approximate best response against this MDP. An agent's supervised learning data ($\mathcal{M}_{SL}$), similarly, predicts the agent's own average behaviour.

NFSP incorporates neural networks as function approximators in fictitious self-play. Each agent has two datasets suitable for deep RL and SL, generated as described above. First using $\mathcal{M}_{RL}$, the agent trains a neural network and off-policy RL to predict action values, $Q(s, a|\theta_Q)$. This network defines the agents' best response strategy to the MDP, with an epsilon exploration factor, $\beta = \epsilon\text{-greedy}(Q)$.

The second network is trained to predict a best response policy $\Pi(s, a|\theta_\Pi)$ against the agent's own past using $\mathcal{M}_{SL}$. During self-play, an agent chooses its actions based on a mixture of both these strategies, π and β.

2.5 Imperfect Recall

With perfect recall, players remember every piece of information during a game. Many algorithms that are guaranteed to converge to a Nash equilibrium are based on perfect recall. Once some information is forgotten, the convergence guarantees do not hold anymore.

Some game information holds no value to a player but explodes the number of information states. Such information should be safe to forget.

Game abstraction is mostly applied to reduce the size of the game. Before finding an equilibrium, the resulting strategies are translated back to the original game. Imperfect Recall is an abstraction where players forget parts of the past information.

In [50] Lanctot shows that Imperfect Recall abstraction holds the convergence guarantees if the abstracted game has well-formed properties (section 5.2). A relaxed version *skew well-formed* is shown to allow deriving a bound on the average regret. Empirical results are shown for another condition being relaxed and still getting good results in CFR. We derived our abstraction based on the results of the relaxed conditions since we don't have well-formed or skew well-formed properties

The game-specific details of Imperfect Recall for Dark Hex (for our version), such as the convergence guarantees and information release for the players, are presented in Section 4.1.

Chapter 3

Analysis of Dark Hex Using Game Theory

In this chapter, we examine Dark Hex using traditional game theory methods. We introduce three new versions of Dark Hex and describe the algorithmic tools we built to investigate the game, collect data for early win conditions, and define a notation to formalize strategies. We give new bounds for the 4×3 board together with the associated strategies.

3.1 Dark Hex Versions

We discuss three alternate versions of Dark Hex alongside the original version, define new terminology, and illustrate the games with short examples.

Classic Dark Hex

We mentioned the rules of the classic version in the Background section. Here we introduce and compare the different versions.

In classic Dark Hex (CDH), after each move by the opponent the player only learns that the opponent has completed their move. After each move attempt, a player learns either that the attempt was successful, and the move was made and it is now the opponent's turn, or that it was unsuccessful, in which case the player tries again. Figure 3.1 shows a sample game.

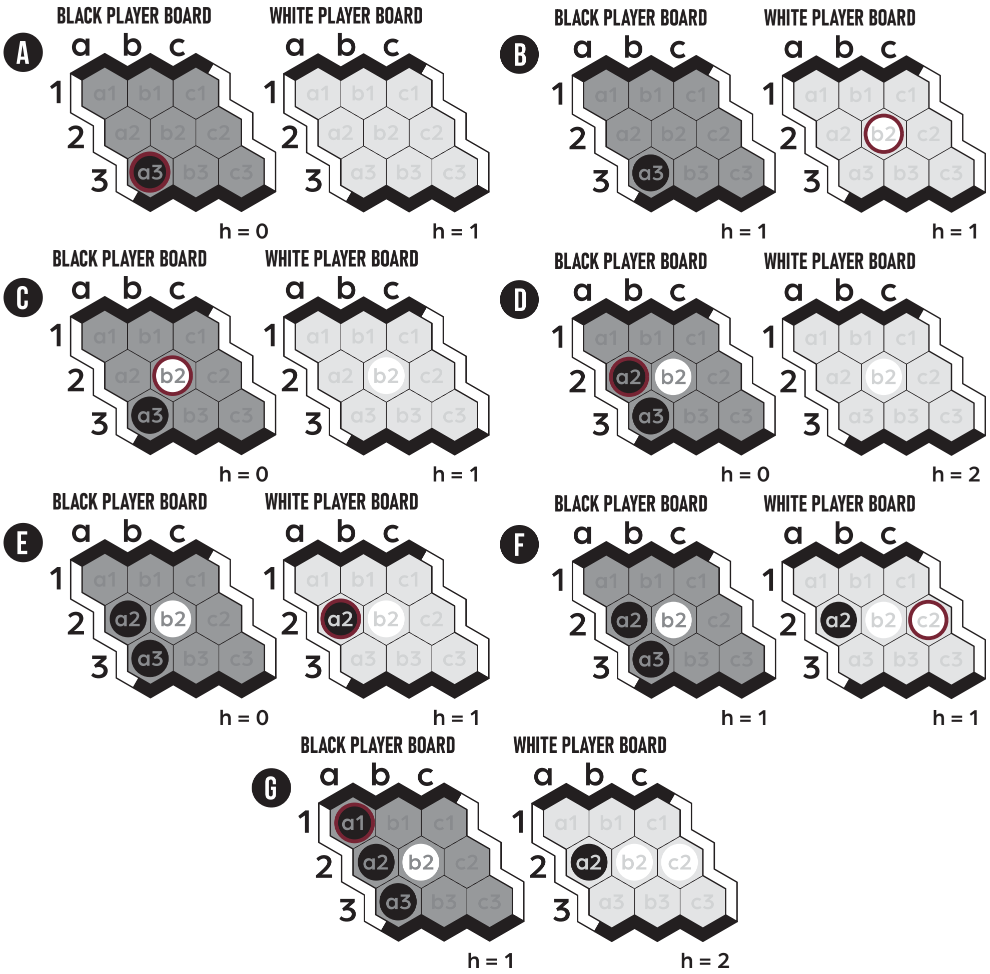


Figure 3.1: A classic Dark Hex game. Both players boards are given for each move. The most current move is shown with a red line around the stone. The number of hidden stones for each player is shown on the right bottom corner of their boards. **A)** Black starts with $a3$. **B)** White plays $b2$. **C)** Black plays $b2$ and discovers (collision) the white stone. **D)** Black makes another move, $a2$. **E)** White discovers the black stone on $a2$. **F)** White plays again, $c2$. **G)** Black wins with $a1$.

Abrupt Dark Hex

Abrupt Dark Hex (ADH) modifies the collision rule of classic Dark Hex. After an attempted move, the player loses their turn. See Figure 3.2.

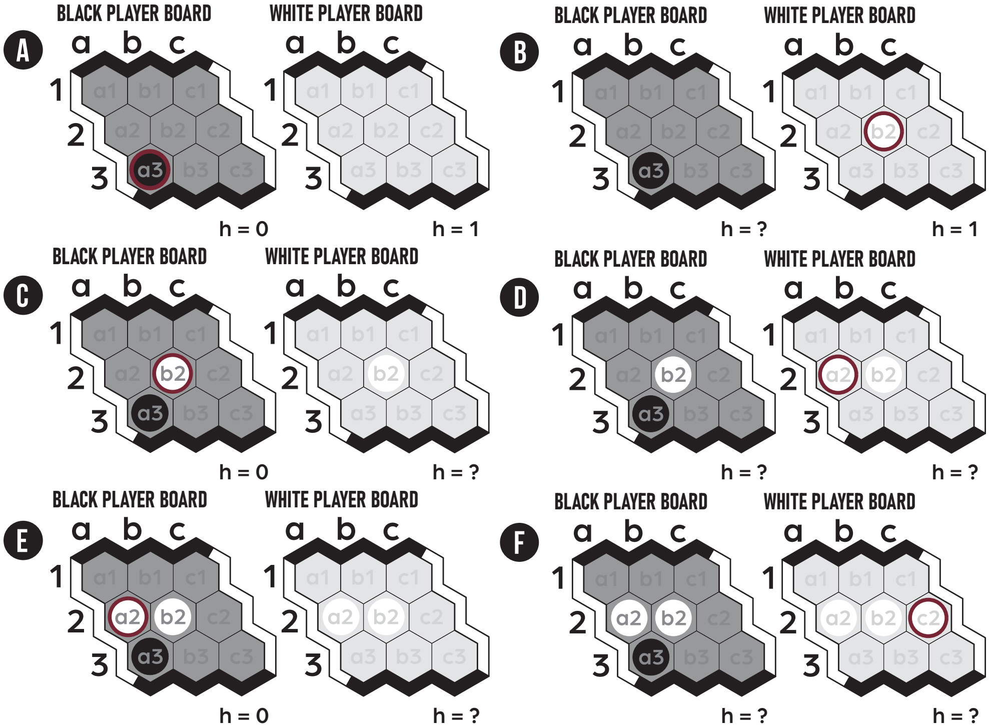


Figure 3.2: The abrupt Dark Hex version of the game shown in Figure 3.1. h differs from the classic Dark Hex. The player often has no clear knowledge of the number of opponent stones on the board. **A)** Black plays *a*3. **B)** White plays *b*2. **C)** Black tries *b*2 and fails. **D)** White plays *a*2 **E)** Black tries *a*2 and fails. **F)** White plays *c*2 and wins.

Noisy Dark Hex and Flash Dark Hex

Classic Dark Hex and abrupt Dark Hex differ in their collision rules, whereas noisy Dark Hex (NDH) and flash Dark Hex (FDH) differ in the information revealed after a collision. After each player's move attempt that is unsuccessful,

- in NDH the opponent is told that there has been a collision,
- in FDH the opponent is told that there has been a collision, and where the collision occurred.

Both CDH and ADH can be played in three ways, with no noise or flash, with noise, or with flash.

In this thesis we only consider CDH and leave further exploration of other variants to future work.

3.2 Dark Hex Strategy Generator

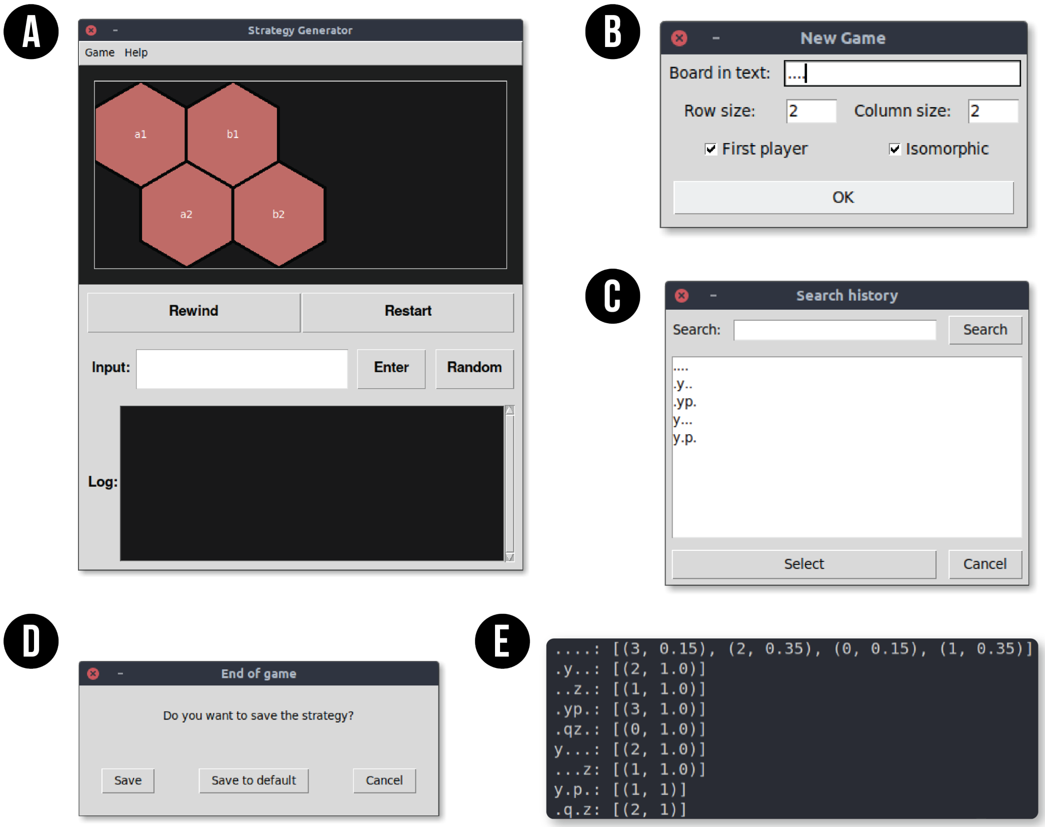
A strategy is encoded as the probability of taking each legal action at a given information state. For a strategy to be complete, it needs to include a probability distribution for every possible game state that the player can encounter when following that strategy.

When we write a strategy on paper, we do not specify the strategy in the same way as on the computer. We use a concise syntax formulation that we can understand with minimal detail. This is very convenient since the number of information states in the strategy can get large.

We developed Dark Hex Strategy Generator (DSaGe), an easy-to-use tool to help generate a strategy. DSaGe makes sure that the given strategy is complete, and that the player has an answer for any reachable information state. We use DSaGe to generate the strategies in Sections 3.4.2 and 3.4.3. Figure 3.3 shows the interface. We can either use notational representation as input or follow along with a game where every game state possible is presented and requires input from the user to define the actions and their probabilities.

For the first input method, the user must follow the notation given in Section 3.4.1. If the given strategy is not complete, DSaGe will give an error and terminate the process. The user then needs to retry after fixing the strategy. This method is only useful when the strategy is developed carefully by hand, or when using an altered version of the output from the second method.

The second method is more intuitive and understandable. The user starts with an empty board and provides the probability distribution for every information state presented. Figure 3.4 shows an example of generating a strategy by using the second method. In the end, DSaGe can output a strategy in the format of a dictionary of the given strategy.



Board in text: ....

Row size: 2 Column size: 2

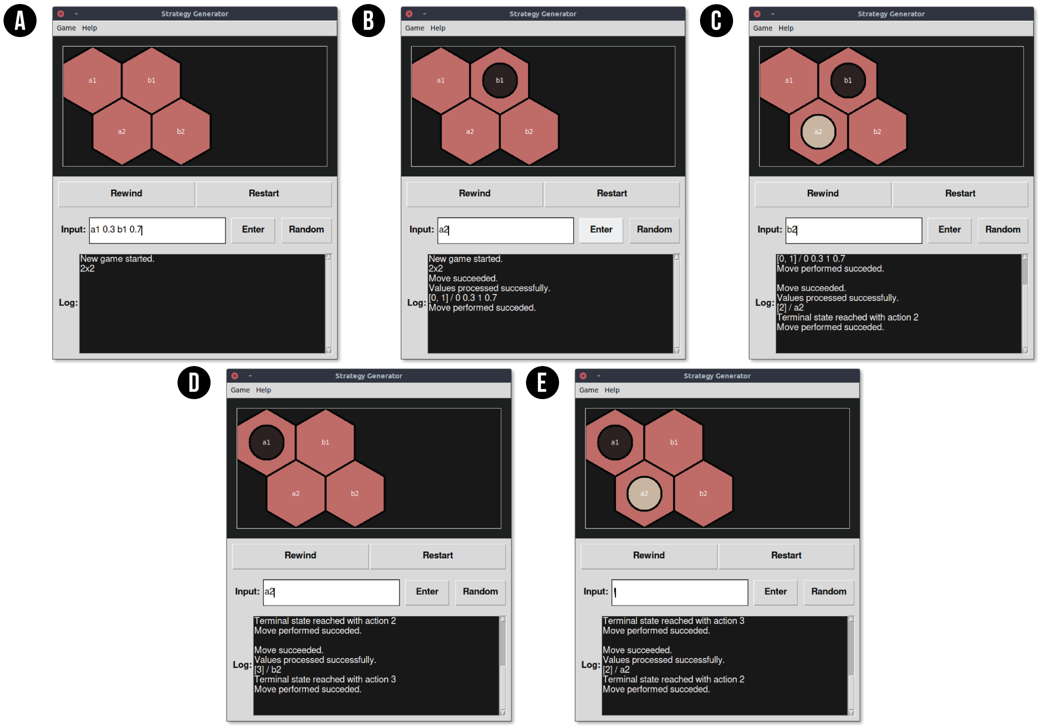
☑ First player ☑ Isomorphic

```
....
.y..
.yp.
y...
y.p.
```

Do you want to save the strategy?

```
....: [(3, 0.15), (2, 0.35), (0, 0.15), (1, 0.35)]
.y..: [(2, 1.0)]
..z.: [(1, 1.0)]
.yp.: [(3, 1.0)]
.qz.: [(0, 1.0)]
y...: [(2, 1.0)]
...z: [(1, 1.0)]
y.p.: [(1, 1)]
.q.z: [(2, 1)]
```

Figure 3.3: DSaGe user interface with a 2×2 Dark Hex example. **A)** Main page. **B)** New Game prompt. Initial board position, row and column sizes must be entered as well as which player the strategy is for, and whether to use isomorphic states. **C)** Past entries can be searched and found in the search history, and the state can be set to any selected position. **D)** After all the necessary input is given, DSaGe prompts the user to save the strategy. **E)** A printed version of the dictionary output, in the form of "information state: (action, probability)".



A) Input: a1 0.3 b1 0.7

```
Log:
New game started.
2x2
```

B) Input: a2

```
Log:
New game started.
2x2
Move succeeded.
Values processed successfully.
[0, 1] / 0 0.3 1 0.7
Move performed succeded.
```

C) Input: b2

```
Log:
[0, 1] / 0 0.3 1 0.7
Move performed succeded.

Move succeeded.
Values processed successfully.
[2] / a2
Terminal state reached with action 2
Move performed succeded.
```

D) Input: a2

```
Log:
Terminal state reached with action 2
Move performed succeded.

Move succeeded.
Values processed successfully.
[3] / b2
Terminal state reached with action 3
Move performed succeded.
```

E) Input: !

```
Log:
Terminal state reached with action 3
Move performed succeded.

Move succeeded.
Values processed successfully.
[2] / a2
Terminal state reached with action 2
Move performed succeded.
```

Figure 3.4: Generating a first player strategy on the 2×2 board using DSaGe. **A)** Play $a1$ with probability 0.3 and $b1$ with 0.7. **B)** Since there is no possibility for a collision any move succeeds. For the next move we play $a2$. There is no probability given, so DSaGe assumes a probability of 1. **C)** If $a2$ is successful the game terminates, otherwise we need to provide another move. We continue with the non-terminated branch, which is a collision, and play $b2$. **D)** After all the branches from first move $b1$ are terminated, DSaGe moves back to the $a1$ branch. We play $a2$ for this information state. **E)** If $a2$ succeeds the game terminates, otherwise we still to provide the next move. Black has no chance to win this game, so we enter ! or press random on the UI for randomly completing this branch of the strategy.

3.3 Evaluation: Best Response

The ultimate goal when producing strategies is finding the optimal strategy, meaning finding a strategy that will win against any strategy as much as it is possible. To be able to find this strategy is the goal we have throughout this thesis. However another problem arises before we can go ahead and come up with new strategies: evaluation.

We have mentioned the bare bones of best response in Chapter 2, but we did not go in detail for the practicality of it. In the case of Dark Hex, being able to calculate true best response is near impossible. In this section, we go over the reasoning as to why, and introduce the alternative approaches we use.

In Dark Hex, we have uneven game tree; a state that is black to play, might be followed by a state that is again black to play. This makes it unfeasible to make true best response calculations based on tree level. Instead we have to calculate it based on the histories. This requires us to keep the entire set of histories in the memory (or calculate them repeatedly), in which case the memory requirements or time requirements to be able to run the best response will be too large to handle with today's computers. Reduced action branching sometimes allows us to calculate such value, but this is limited to very small strategies.

We introduce Abstract Best Response (Ab-BR) as a way of evaluating the game in the abstract space. We use this heuristic for getting an idea of how well a given strategy is doing in the imperfect recall setting. Unfortunately however, the best response value found on the abstract game does not translate back to the real game. For example, Figure 3.5 shows a strategy where the move ordering is important and Ab-BR fails to capture the real best response. Here if the player started with $c1$ the next move is going to be $b3$, if player started with $a4$ the next move will be $b2$. This strategy is a part of our handcrafted strategy, and a key point to evaluation; true best response will use the history and get 0.625 as value while Ab-BR will get 0.75.

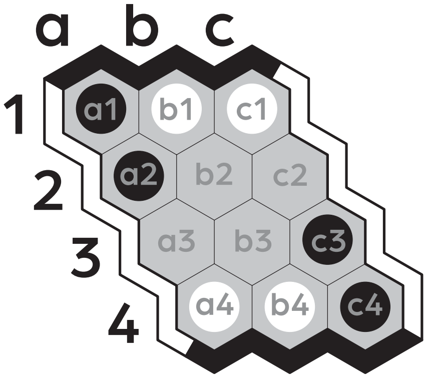


Figure 3.5: An example state where move ordering effects the outcome of the best responder and the player.

In Ab-BR, instead of using the histories to calculate the reach probabilities, we are bucketing each history to information sets and using the summation of the reach probabilities as a metric to reaching some information state without the knowledge of the history. So the Ab-BR value of strategy σ in information set I for player i is

$$v_i(\sigma, I) = \max_{a \in \mathcal{A}} q(I, a) \sum_{h \in I} \pi^{\sigma}_{-i}(h)$$

Apart from this heuristic, which does not hold much value in the full game, we are also making use of Approximate Best Response (ABR). ABR uses Reinforcement Learning to find a best response strategy against any given strategy. When we fix a strategy in an imperfect information environment such as Dark Hex, we get a single player stochastic environment. We then can use this single player environment in combination with well known RL algorithms such as Alpha Zero or DQN or any other one that does well in this setting.

For our experiments we use an implementation with a DQN due to time constraints, but using Alpha Zero would give a closer result due to stronger signal.

3.4 New Strategies and Improved Bounds for 4×3 Dark Hex

Duane Broline proved that the first player can always win Dark Hex on a 3×3 board [54]. For any board size $3 \times m$ where $m > 3$, the first player can use the same 3×3 strategy on a part of the board, ignoring the other part, to guarantee a win.

The more interesting question for us is if the first player needs to go the longer distance on a $m \times 3$ board where $m > 3$. We focus on the 4×3 board. Previous bounds for 4×3 Dark Hex found by François Bonnet were that the first player can win with at least probability 0.112 and the second player with at least probability 0.732 [12]. However Bonnet does not specify any strategy to actually match these numbers. We aim to give strategies that one can play against, and also provide the matching best response values. In this section, we provide new hand-crafted strategies discovered by Ryan Hayward with values of 0.1428 and 0.75 against the Ab-BR. We then get 0.625 for the second player against true best response. Later on, in Section **??** we present two better strategies generated by our players and discuss some of the differences.

We first provide the notation that we use to formalize the strategies and then explain strategies S_1 and S_2 for black and white players respectively.

3.4.1 Notation

We use symbols to describe the moves in a player's strategy. A complete strategy from the root must contain an answer for every possible reachable information state. This notation provides all the essential tools to describe a complete strategy.

1. $\boldsymbol{s_i}$**:** Play move/sub-strategy s_i.

2. $\boldsymbol{s_i, s_j}$**:** Play moves/sub-strategies s_i, s_j in order.

3. $\boldsymbol{(s_i, s_j)}$**:** Priority on groups of strategies. Parenthesis imply a precedence on the strategies.

4. $\boldsymbol{\{s_0 : p_0, s_1 : p_1, \cdots, s_n : p_n\}}$**:** Play moves/sub-strategies s_i with corresponding probabilities p_i, where $0 \leq i < n$. Every move must be paired with a probability and $\sum_i p_i = 1$.

5. $\boldsymbol{\{s_0, s_1, \cdots, s_n\}}$**:** Play moves/sub-strategies s_i equiprobably, $p_i = \frac{1}{n+1}$.

6. $\boldsymbol{s_i \neg (\cdots)}$**:** Try s_i and if there is a collision continue inside the parentheses.

7. $\boldsymbol{\{s_i | s_j\}}$**:** Equivalent to $\{s_i \neg (s_j), s_j \neg (s_i)\}$

8. **!:** Continue uniformly at random, select any legal action until the game is over.

As an example, we develop a first-player strategy for the 2×2 board. Assume the strategy S_b where Black first plays $a1$ with probability 0.3 and $b1$ with probability 0.7. There is no possibility for the first move to fail since

Black starts the game, therefore there is no need to specify what would happen in the case of a collision. For the second Black move, if the first move was $a1$ the player plays $a2$. If that fails, it is a definite loss no matter where the opponent plays, so Black can play anywhere. If the first move was $b1$, the second move will be $a2$ or $b2$ equiprobably, if either one fails then Black will play the other one.

We formalize this example using our notation:

$$S_b \quad = \quad \{a1(a2\neg(!)) : 0.3, b1(\{a2|b2\}) : 0.7\}$$

Figure 3.6 shows the strategy visually, with the corresponding notational representation on each move or condition.

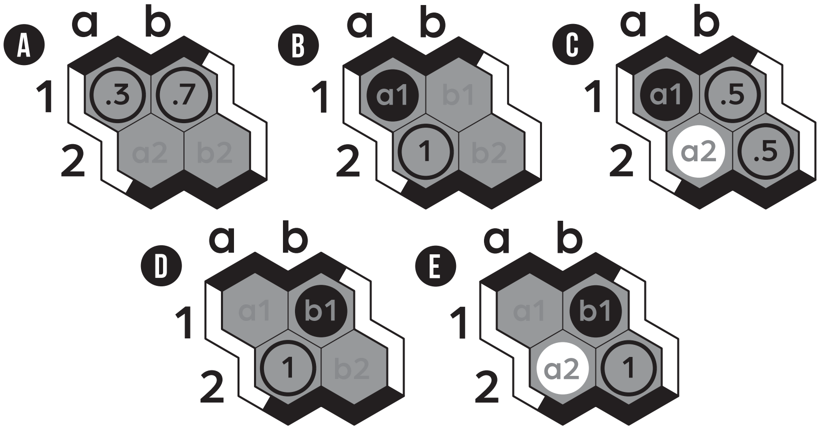


Figure 3.6: Visual representation of the strategy S_b. Circles present the next moves with their probabilities in the middle. **A)** Sub-strategy $\{a1(...) : 0.3, b1(...) : 0.7\}$. **B)** Sub-strategy $a1(a2\neg(...))$. **C)** Sub-strategy $a1(a2\neg(!))$. **D-E)** Sub-strategy $b1(\{a2|b2\})$.

3.4.2 Improved First Player Strategy for 4×3 Dark Hex

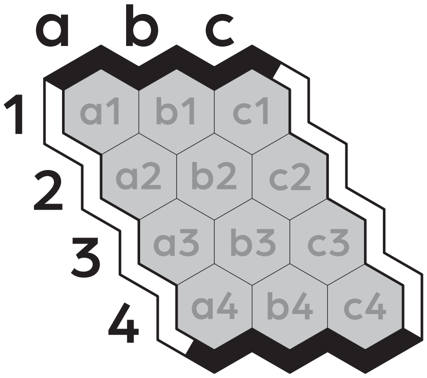


Figure 3.7: An empty 4×3 Dark Hex board.

In strategy S_1, Black plays $\{b2, b3\}$ as the opening move. The $b3$ strategy, S_1^{b3} is the isomorphic transformation of $b2$, S_1^{b2}. We show this transformation as $T(S_1^{b2}) = S_1^{b3}$. T simply converts the moves to their isomorphic equivalents. For the 4×3 board, isomorphic pairs of cells are shown in Figure 3.8.

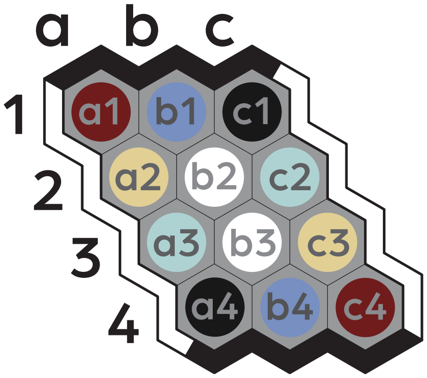


Figure 3.8: Isomorphic pairs on the 4×3 board marked with the same color. The pairs are: $(a1, c4)$, $(b1, b4)$, $(c1, a4)$, $(a2, c3)$, $(b2, b3)$, $(c2, a3)$.

In S_1^{b2}, Black has 3 connection options from $b2$: Connect to the top edge through $b1, c1$ (x), connect to $c3$ through $b3, c2$ and $b4, c4$ (z), or connect to $a4$ through $a3, b3$ (y), shown in Figure 3.9. In the end Black needs to connect x with either y or z to win. In strategy S_1^{b2}, Black selects one of x, y, z to start with, and tries to connect in randomized manner.

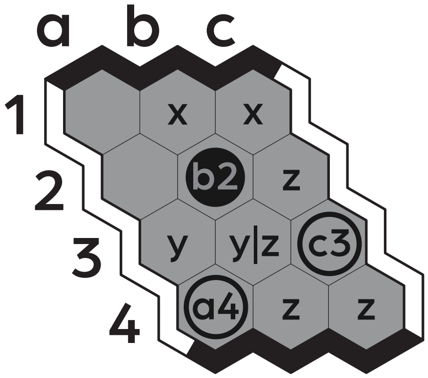


Figure 3.9: Black connections for the continuation on S_1^{b2}. The figure summarizes the idea of the strategy. After $b2$, Black moves $c3$, $a4$ to get the connections y and z respectively, or moves on a x. The filled circle represents a move that was already played, and empty circles represent the possible connection stones for Black.

We provide the strategy S_1 formally in Definition 1.

Definition 1: Formal Definition of S_1

$$\begin{aligned}
S_1 &= \{S_1^{b2}, S_1^{b3}\}\\
S_1^{b2} &= b2\ \{\{\alpha, \beta, \gamma\} : 0.858, \{\delta, \theta\} : 0.142\}\\
S_1^{b3} &= T(S_1^{b2})\\
\\
\alpha &= \{b1|c1\}, \{X, Y\}\\
X &= a4\neg(c3\neg(!), \{x_0|x_1\}), a3\neg(b3\neg(!))\\
x_0 &= b3\neg(c2, b4\neg(c4)), b4\neg(c4)\\
x_1 &= b4\neg(c4, b3\neg(c2)), b3\neg(c2)\\
Y &= c3\neg(a4\neg(!), a3\neg(b3)), \{y_0, y_1\}\\
y_0 &= b3\neg(c2\neg(!), b4\neg(c4\neg(a4))), b4\neg(c4\neg(a4)), !\\
y_1 &= b4\neg(c4\neg(a4, b3\neg(a3))), b3\neg(c2\neg(!))\\
\\
\beta &= c3\neg(a4, a3\neg(b3), b1\neg(c1), !), b3\neg(c2\neg(!)), \{\beta_1, \beta_2\}, !\\
\beta_1 &= c1\neg(b1\neg(!)), b4\neg(c4\neg(!))\\
\beta_2 &= b4\neg(c4\neg(a4)), c1\neg(b1\neg(!))\\
\\
\delta &= c3\neg(a4, c1\neg(b1), a3\neg(b3), !), c1\neg(b1\neg(!)), \{\delta_0|\delta_1\}, !\\
\delta_0 &= b3\neg(c2\neg(!)), b4\neg(c4\neg(a4))\\
\delta_1 &= b4\neg(c4\neg(a4, b3\neg(a3))), b3\neg(c2\neg(!))\\
\\
\gamma &= a4\neg(\gamma_{neg}), b3\neg(a3\neg(!)), b1\neg(c1), !\\
\gamma_{neg} &= c3, b3\neg(c2), \{c1\neg(b1), b3\neg(c2)\}, !\\
\\
\theta &= a4\neg(\theta_{neg}), \{c1|b1\}\neg(!), a3\neg(b3), !\\
\theta_{neg} &= c3, \{c1|b1\}, b3\neg(c2), b4\neg(c4), !
\end{aligned}$$

Using strategy S_1, Black wins with at least probability 0.1428 in abstract game. Figures 3.10 and 3.11 show a Ab-BR strategy for S_1^{b2}, playing against S_1^{b2}.

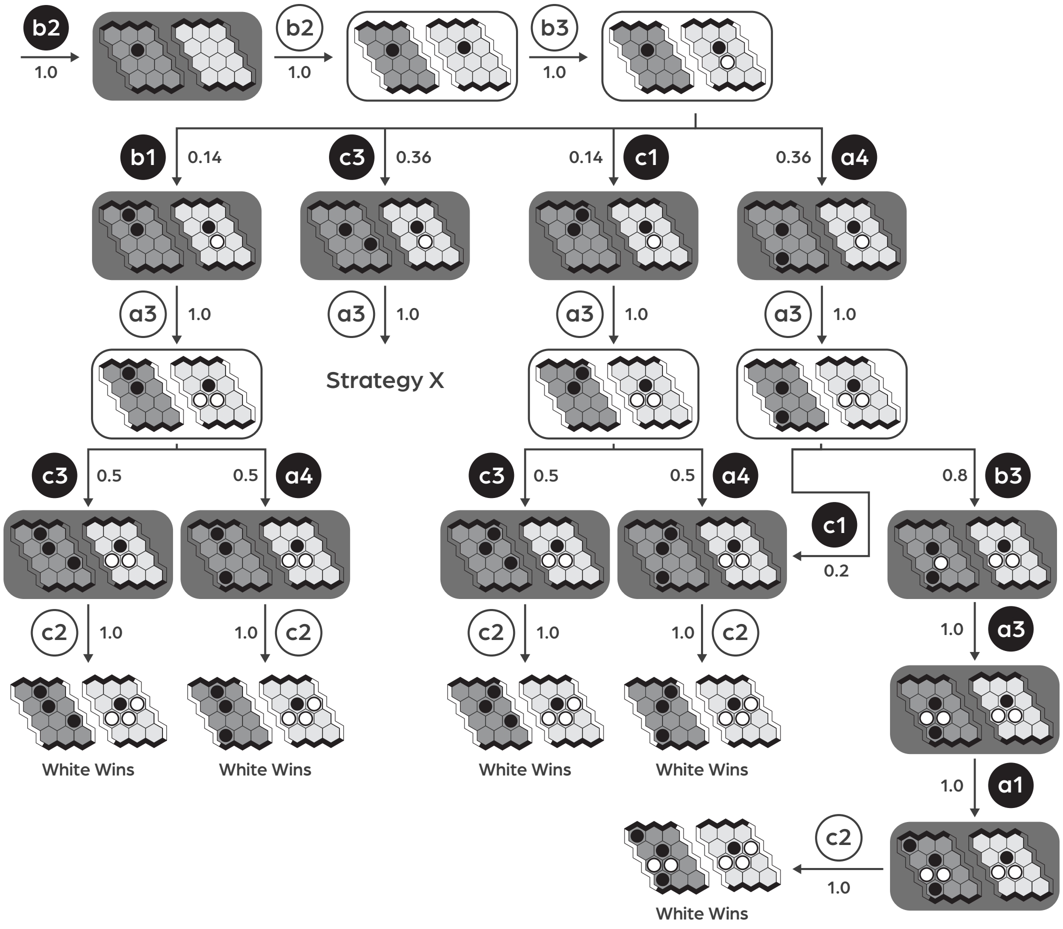


Figure 3.10: The game between S_1^{b2} and a Ab-BR strategy against S_1. Each arrow represents a move. The circles represent the action taken by the player of that color and its location. Fractions next to the arrows show the probability of the given moves. Boards in the same box represent the information state for black and white players (black left, white right). The color of the boxes shows which player board is updated last. If no box surrounds the boards, the state is terminal. Figure 3.11 shows the rest of the strategy continuing from the label **Strategy X**.

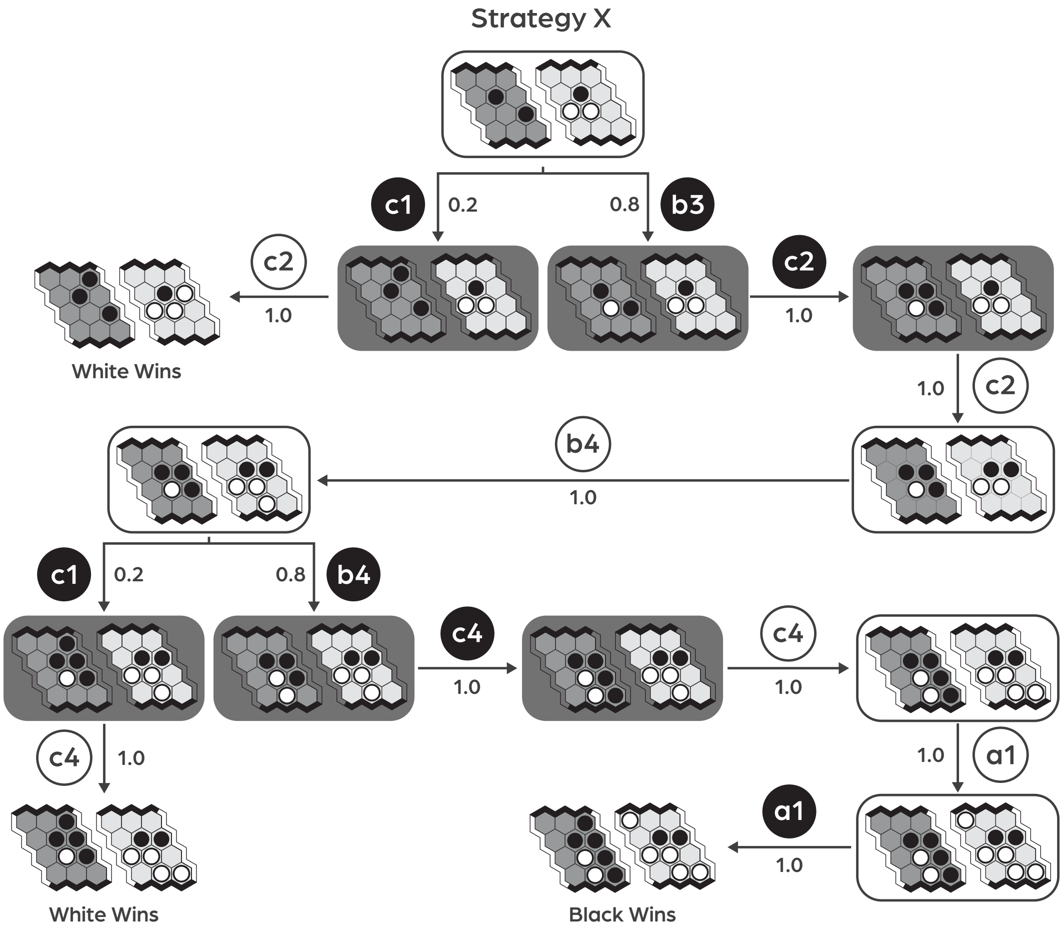


Figure 3.11: Strategy X, continuation of Figure 3.10. This branch contains the only win for the black player.

3.4.3 Improved Second Player Strategy for Dark Hex

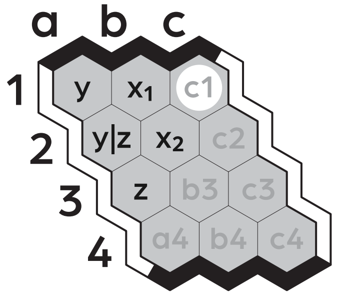


Figure 3.12: Visual representation for the beginning move sequence (following the edge). After $c1$, White tries to x_1 and connects with y. If x_1 fails, White moves to x_2 and connects with z.

In the second player strategy S_2, the first move is on either of the obtuse corners, $\{c1|a4\}$. If there is a collision, the player switches to the other corner, and plays according to the 3×3 strategy, which is a sure win. If the first move succeeds, the first player continues on the same side of the board and tries to connect by simply following the edge. For example if $c1$ succeeds, the next moves will be on $b1$ and $a1$. Figure 3.12 shows this connection idea. Unless Black blocked White using one of the patterns given in Figure 3.13, White will win following this strategy.

As an example, assume Black plays $b1, b3$ as the first two moves while White played $c1$. White can now follow the strategy $S_w = b1\neg(b2, a2\neg(a3)), a1\neg(a2\neg(\ldots))$, plays $b1$ and if succeeds continues with $a1$ and $a2$, since Black did not block $a1, a2$ White will win. If $b1$ fails, White plays $b2$, and connects with $a2, a3$. Figure 3.14 shows the end game.

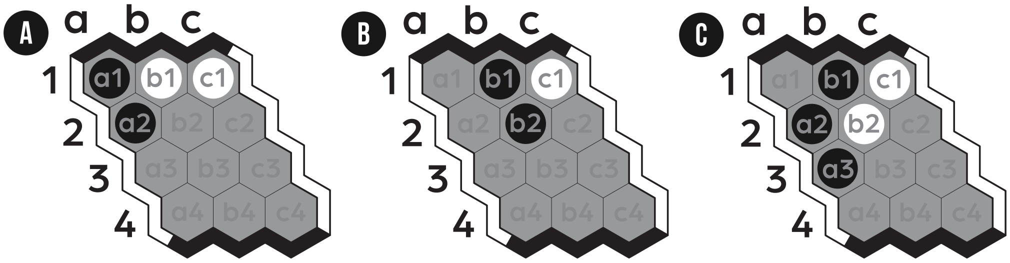


Figure 3.13: **A)** $a1, a2$ block. If considering White moves of $c1, b1$, Black leaving one of these cells empty will result in a White win. **B)** $b1, b2$ block. Without letting White move to $b1$, Black blocks White forcing him to play on the other edge. **C)** $b1, a2, a3$ block. White will discover all the black stones, and will be forced to play $a4$. This is not a good blocking for Black. Even though the win for White is delayed, it still is guaranteed. After White $a4$, Black must play $b3$ to interrupt the immediate win. White then follows the bottom edge and wins. These are the only move sequences for Black to stop White from winning with the first three successful moves.

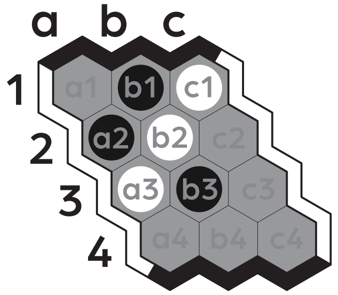


Figure 3.14: A sample game when Black does not use the blocks from Figure 3.13. White wins by just following a shortest path to the left edge.

If the direct path is blocked as in Figure 3.13, White moves to the opposite corner or its adjacent cell with equal probability, so ($\{a3, a4\}$). From here, the probability of White winning is 0.5 since White picks a side (either $b3, c2$ or $b4, c4$), and if Black picks the same pair, Black wins and otherwise White wins. See Figure 3.15 for an example of this case.

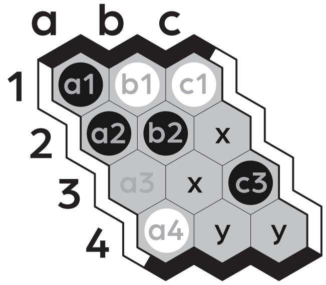


Figure 3.15: A sample information state. White players view, we assume that White has discovered all the black stones. White must select connecting x's or y's as his next move. If Black selects the same pair Black wins, otherwise White wins. The winning probability for either player from this position is 0.5.

Figures 3.16, 3.17 and 3.18 show the strategy S_2 playing against a Ab-BR strategy.

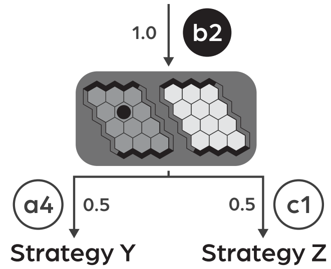


Figure 3.16: Visual representation of the game between S_2 and a Ab-BR strategy against S_2. Each arrow represents a move. The circles represent the action taken by the player of that color and its location. Fractions next to the arrows show the probability of the given moves. Boards in the same box represent the information state for black and white players (Black on the left, White on the right). The color of the boxes shows which player board was updated last. No box surrounds the terminal state. Figures 3.17 and 3.18 show the rest of the strategy continuing from the labels **Strategy Y** and **Strategy Z**.

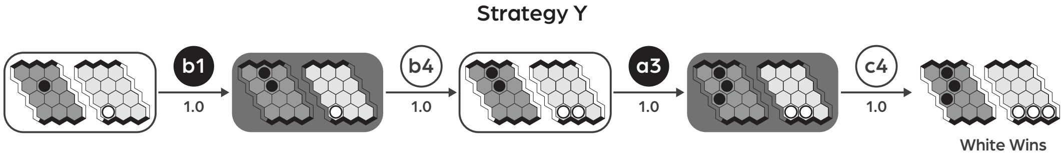


Figure 3.17: Strategy Y, continuation of Figure 3.16. Ab-BR player (black) gives up this branch, and focuses on Strategy Z instead. Black does not try to stop the connection on the bottom edge.

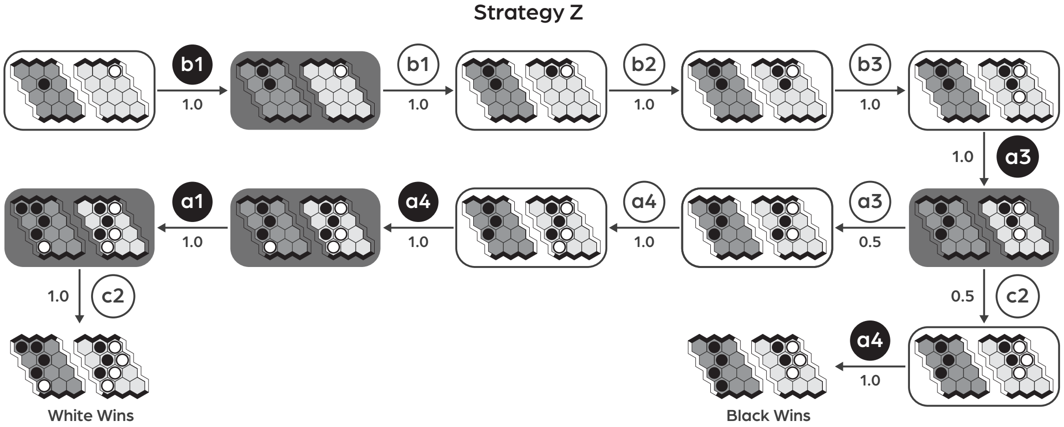


Figure 3.18: Strategy Z, continuation of Figure 3.16. Black wins this branch with 0.5 probability. This gives the Ab-BR player a 0.25 probability to win overall.

The $a4$ strategy, S_2^{a4} is the isomorphic transformation of $c1$ strategy, S_2^{c1},

$T(S_2^{c1}) = S_2^{a4}$. So we fully describe details of S_2^{c1}, then define S_2^{a4} as a transformation.

Definition 2: Formal Definition of S_2

$$\begin{aligned}
S_2^{c1} &= c1\neg(\alpha), b1\neg(\beta), a1\neg(a2\neg(\theta)) \\
S_2^{a4} &= T(S_2^{c1}) \\
S_2 &= \{S_2^{c1}, S_2^{a4}\} \\
\\
\alpha &= a4, \{\alpha_0, \alpha_1\} \\
\alpha_0 &= b3\neg(b4, c3\neg(d3)), c2\neg(c3\neg(!)) \\
\alpha_1 &= b4\neg(b3, c2\neg(c3)), c3\neg(c4\neg(!)) \\
\\
\beta &= b2\neg(\beta_{neg}), a2\neg(a3\neg(a4, b3\neg(b4, c4\neg(c3)))) \\
\beta_{neg} &= b3, \{b_0, b_1\} \\
b_0 &= a3\neg(a4, c2\neg(c3)), c2\neg(c3\neg(!)) \\
b_1 &= c2\neg(c3, a3\neg(a4)), a3\neg(a4)\} \\
\\
\theta &= \{\delta, \gamma\} \\
\delta &= a3\neg(\delta_{neg}), b2\neg(b3\neg(!), c2\neg(c3\neg(!))) \\
\delta_{neg} &= a4, \{n_1, n_2\} \\
n_1 &= b3\neg(b4, c4\neg(c3)), c2\neg(c3\neg(b2)) \\
n_2 &= b4\neg(b3, c2\neg(c3)), c4\neg(c3\neg(b3, b2\neg(c2))) \\
\gamma &= a4\neg(a3, b2\neg(b3, c2\neg(c3))), \{\gamma_1, \gamma_2\} \\
\gamma_1 &= b3\neg(b4\neg(c4\neg(c3))), c2\neg(c3\neg(b2\neg(!))) \\
\gamma_2 &= b4\neg(b3\neg(!), c2\neg(c3\neg(b2))), c4\neg(c3\neg(!))
\end{aligned}$$

Chapter 4

Self-Learning Players: Reinforcement Learning and Regret Based Methods

In Chapter 3 we introduced our handcrafted player. In this chapter we develop new learning agents which use combinations of:

- Early terminal states pruning
- Imperfect Recall abstraction
- Neural Fictitious Self-Play
- Monte Carlo Regret Minimization with Outcome Sampling

After explaining the details of each method mentioned in the list above, in Section 4.5, we experimentally compare the MCCFR and NFSP algorithms in different settings on the 4×3 board. In Section **??** We learn new strategies for both Black and White that improve the best response values provided in Chapter 3 and are generated completely by computer using an end-to-end learning approach. We then conclude this chapter by introducing two incrementally better post processing algorithms SIP and SIP+, and the improved best response values we get using these new methods.

4.1 Imperfect Recall Dark Hex

We use the imperfect recall (IR) abstraction [50] to reduce the size of the game to a level where we can apply the MCCFR and NFSP algorithms. The state-space of Dark Hex grows drastically with the board size, see Table 4.1. We need to reduce the game size to allow good approximations with little information loss.

Board Size	**Abstraction**	**Game Size**	**Relative Size**
2×2	Perfect Recall	441	1
	Imperfect Recall	42	0.9
3×2	Perfect Recall	196357	1
	Imperfect Recall	410	0.02
3×3	Perfect Recall	19119486978	1
	Imperfect Recall	12556	6.5×10^{-8}
4×3	Perfect Recall	$\approx 10^{17}$	1
	Imperfect Recall	367919	$\approx 3.6 \times 10^{-13}$

Table 4.1: Comparison of Perfect Recall and Imperfect Recall Dark Hex on small board sizes. The game size refers to the number of information states for every player.

In this section, we discuss why an imperfect recall abstraction is a good first step to abstracting Dark Hex. We then evaluate the information loss of IR on 2×2 boards by using the vanilla CFR algorithm [31].

With perfect recall, a player remembers every piece of information seen during a game. In Dark Hex, this means that the player knows their board and the order of the actions. In Chapter 3, when evaluating and generating strategies by hand, we do not take the order of the previous moves into account: the only thing that matters in the information state is the board itself. We use the same idea for imperfect recall abstraction. We ignore action history and keep only the current board information.

In NFSP and MCCFR we make use of the IR abstraction. In Section **??** we show that the computer-generated strategy that uses the IR abstraction gives better minimax bounds than the handcrafted strategies in Chapter 3, showing the success of this abstraction method.

To compare perfect recall Dark Hex with IR Dark Hex we use vanilla CFR.

For the sake of keeping the computation small, we use a 3×2 board and report the best response value. We use the same settings of vanilla CFR for both algorithms except the abstraction to isolate the effect.

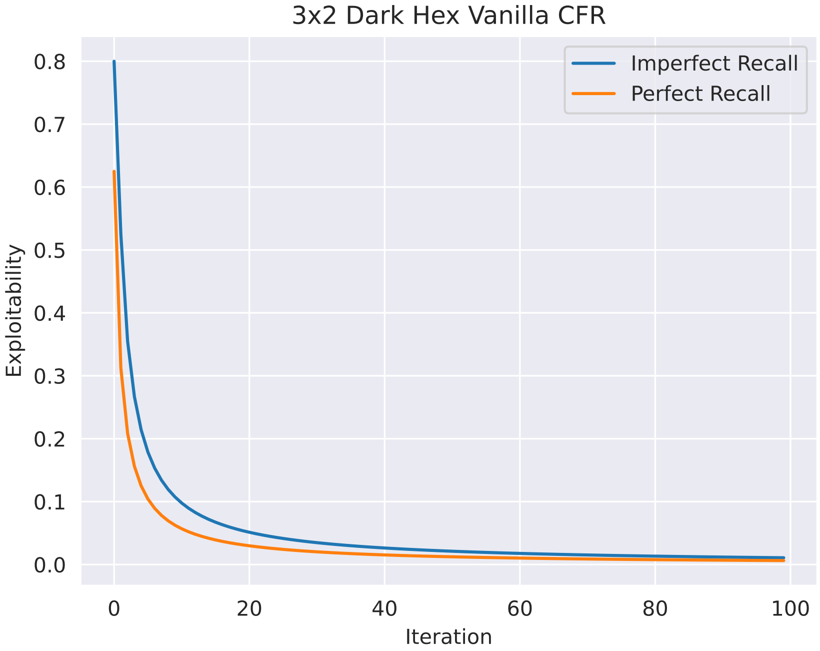


Figure 4.1: Comparison of perfect and imperfect recall on a 3×2 Dark Hex board.

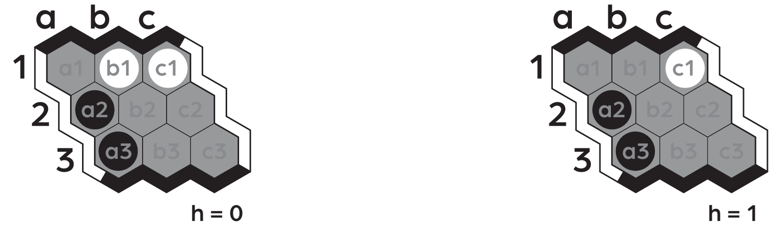
As we see in Figure 4.1 the best response value in imperfect-recall abstraction converges almost as well as perfect recall with a minor difference. In the meantime, the reduction of the size of the game is $\approx 10^{13}$ for boards bigger than 3×3 (Table 4.1). This is a huge improvement as our main concern is the game size.

4.2 Probability One Win States

In Dark Hex, if players play optimally, the set of terminal states can be extended by probability-one win states (pONE), as explained by Bonnet [12]. In this section, we explain how we handle pONE and give two different examples.

A Dark Hex game state is terminal if and only if one player has joined their two sides of the board. However, sometimes a player knows they can win with probability 1. When we generate a strategy by hand, we use this idea. We stop looking for a response and resign if we see that the opponent has a pONE win, or we follow our own pONE strategy.

In Figures 4.2a and 4.2b show two examples. In Figure 4.2a, Black knows all the white stone locations. There is no hidden stone and $a1$ is a winning move for Black. We can pre-compute this information state, and stop here, there is no need to make another move and actually play $a1$. In Figure 4.2b, Black can win with either $a1$ or $b1$, and number of hidden stones is 1. Black tries both cells and wins with probability 1. This is another pONE state where Black does not need to continue playing. Figure 4.3 shows a deeper example.



(a)

(b)

Figure 4.2: Two pONE examples. Both figures show the black player's board. **h** is the number of hidden stones.

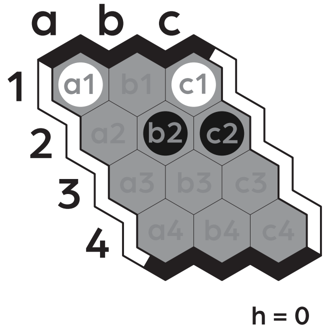


Figure 4.3: A pONE example on the 4×3 board. There are no hidden stones and black is to play. Black plays $b1$ to block white. After White move, Black tries $c3$. If $c3$ is occupied, plays $b3$ and wins with probability one: if $c3$ is not occupied, Black tries $b4$ and $c4$. If either are free black wins; if not, Black plays $a4$ and wins by connecting $a3$ or $b3$.

Incorporating pONE with our algorithms has two benefits: It reduces the game tree size and strengthens end-game play. We present two experiments to quantify these benefits.

Figure 4.4 shows the difference in the number of states encountered and the time it takes to have a single traverse on the search tree on a 4×3 Dark Hex board when using and not using pONE. We see a significant difference in the number of states, where the time shows only a minor benefit. This gap is due to the table lookup, even though the operation for item search is O(1), having to search for every state adds up to the total time. The main benefit here is for memory since we keep the information states stored. For 4×3 board, we have over %20 savings on memory.

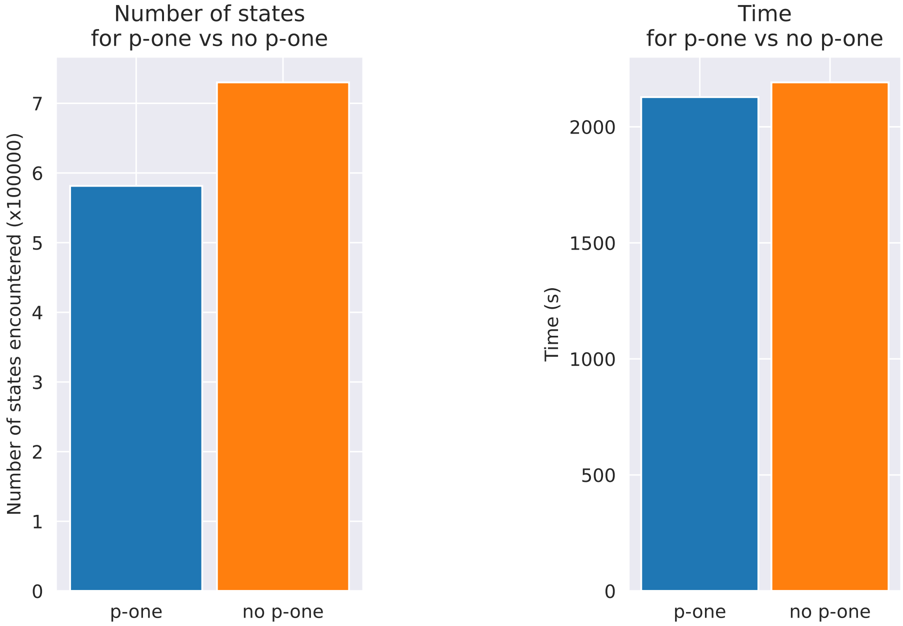


Figure 4.4: The effect of pONE on traversing the 4×3 board.

The second experiment shows that the use of pONE improves the end game results of the players. We train an MCCFR agent for 10^6 iterations on 4×3 imperfect recall Dark Hex. We then make this agent play 10^4 games against itself. Every time the agent loses a game where the pONE value was a win for itself at some state, we count that as a “missed win”. The result shows 828 missed wins, equaling to 8.2% better performance using pONE.

We used NFSP and MCCFR to train agents with the imperfect recall abstracted game and pONE pruning. In Sections 4.3 and 4.4 we first discuss the details of these algorithms applied to Dark Hex, and why are they feasible and good choices in this case. In section 4.5 we pit the agents using the handcrafted player, NFSP, and MCCFR against each other in games with perfect and imperfect recall. Lastly, in Sections 4.6 and 4.7 we introduce new Dark Hex strategies using SIP and SIP+, and present improved best response values as well as new results of the tournament with the improved players.

4.3 NFSP for Dark Hex

Neural Fictitious Self-Play is an anytime algorithm which can run with limited computational power and time. As explained in Section 2.4, NFSP iteratively improves players based on their average behaviour and an approximate best response against their opponent's average strategy. Given the right parameter adjustment, NFSP converges to the Nash equilibrium in the limit [53].

The NFSP agents are trained using a simple Residual Network with 2 convolutional layers. The first layer uses 512 filters with a kernel size of 3. Relu activation is applied after the layer. The second layer uses 256 filters with kernel size 3 and Relu afterwards again. After the second layer, max-pooling of size 2 is applied before flattening the outputs. Lastly, 2 dense hidden layers were added with 512 and 256 units and applied relu after each.

We use a replay buffer capacity of 2×10^5 and reservoir buffer capacity of 2×10^6. The learning starts at the buffer size of 1000. The rest of the parameters are as in the implementation of NFSP in version 1.1.1 of Open Spiel [55].

4.4 MCCFR for Dark Hex

Counterfactual regret minimization is a widely used algorithm in practice when developing players for imperfect information games. Vanilla CFR (Section 2.2.2) traverses the full game tree, therefore is not a good candidate for large games. Monte Carlo Counterfactual Regret Minimization [32] is a sample-based version of CFR. It is an anytime algorithm that can run as long as time and computation power allows.

We use Outcome Sampling (Section 2.2.4) as the sampling method in all the experiments with MCCFR. The training speed of MCCFR is one of the reasons why we prefer it over other methods. Compared to NFSP, MCCFR gets much better results much faster. All our MCCFR players are trained in less than a day while NFSP players were trained in half a week each. The results in Section 4.5 shows the performance even with this time difference.

4.5 Comparing Player Strengths: Round-Robin Arena

We evaluate the success of our different implemented methods on a 4×3 board.

We play $n = 10^5$ games for each pair of players. Players play as Black half the time. We describe each agent with their specific settings and provide the results and discussion.

- **Handcrafted Player (HP):** Implements the handcrafted strategy described in Chapter 3. We know that this player has 0.75 win probability in abstract game as White and 0.124 as Black.

- **MCCFR with IR and pONE (MCCFR-IR-p):** An MCCFR agent using Outcome Sampling. This agent uses both pONE and the IR abstraction. It was trained for 10^9 iterations.

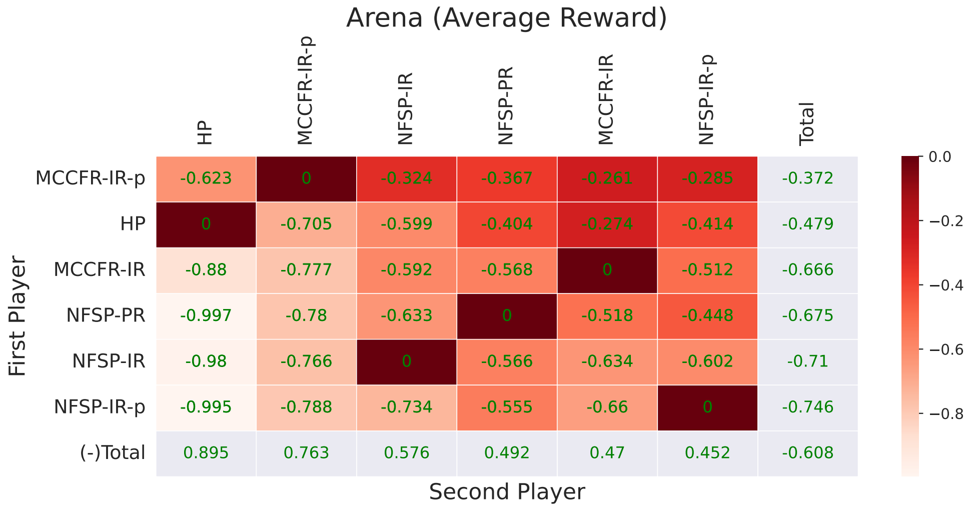
- **MCCFR with IR and without pONE (MCCFR-IR):** Same as MCCFR-IR-p, except without pONE.

- **MCCFR with perfect recall (MCCFR-PR):** Same settings as MCCFR-IR, except without Imperfect Recall.

- **NFSP with IR and pONE (NFSP-IR-p):** An NFSP agent using both pONE, and Imperfect Recall.

  NFSP with IR and without pONE (NFSP-IR): Same as NFSP-IR-p but without pONE.

  NFSP with perfect recall (NFSP-PR): Same settings as NFSP-IR, except without Imperfect Recall and having a different neural network setting. Since perfect recall is not suitable for convolutional networks as we have history data that does have any spacial value to it, we use a linear neural network instead. The details are the same as the *mlp* network in version 1.1.1 of Open Spiel [55].



	HP	MCCFR-IR-p	NFSP-IR	NFSP-PR	MCCFR-IR	NFSP-IR-p	Total
MCCFR-IR-p	-0.623	0	-0.324	-0.367	-0.261	-0.285	-0.372
HP	0	-0.705	-0.599	-0.404	-0.274	-0.414	-0.479
MCCFR-IR	-0.88	-0.777	-0.592	-0.568	0	-0.512	-0.666
NFSP-PR	-0.997	-0.78	-0.633	0	-0.518	-0.448	-0.675
NFSP-IR	-0.98	-0.766	0	-0.566	-0.634	-0.602	-0.71
NFSP-IR-p	-0.995	-0.788	-0.734	-0.555	-0.66	0	-0.746
(-)Total	0.895	0.763	0.576	0.492	0.47	0.452	-0.608

Figure 4.5: Ranking for players playing as black and white on the 4×3 board. Each player played 10^5 games against all the other opponents. Players receive +1 for a win and -1 for a loss.

Figure 4.5 shows that as Black the MCCFR-IR-p performs the best followed by MCCFR-IR. HP is ranked third, showing that the Black strategy we generated by hand was weaker than the MCCFR players. On the other hand, HP takes the first place as the White, at first glance this might seem counter intuitive, but if we look closely, we can see that MCCFR-IR-p is still outperforming HP, but HP performs better against the other players therefore in summation gets the best player spot for white.

4.6 Better Strategies for 4×3 Dark Hex using Monte Carlo CFR

In this section, we introduce Simplified Policy (SIP) algorithm to get better strategies than HP on the 4×3 board. The strategies are completely computer-generated. We get a 0.786 win rate against the abstract best response for the second player, compared to 0.75 from the previous handcrafted strategy, and 0.205 for the first player, greatly improving the previous 0.1428. This gives us a $(1 - 0.786 - 0.205) = 0.009$ epsilon on abstract best response. We also get 0.153 and 0.757 win rate against approximate best response in full game, which gives us an epsilon of $(1 - 0.757 - 0.153) = 0.09$. In Section 4.7 we improve even further by adding smoothing on top of SIP, reaching a pair of strategies within $\epsilon = 0.041$ of the Nash equilibrium (ABR).

We first explain the process of generating the strategy, then present figures showing the new strategies playing against abstract best response. Lastly, we discuss some of the differences between our handcrafted strategy and SIP strategy.

SIP is a post-processing algorithm. We use MCCFR-IR-p as a base since we got the best performance from it in Section 4.5. SIP introduces two new parameters:

1. **Branching factor:** We limit the number of actions to b. The best-response calculation is heavily effected by the branching factor. Keeping it minimal allows us to generate a best-response strategy if b was small enough.

2. ϵ **threshold:** To reduce the irrelevant actions we use a threshold of ϵ, and ignore actions with probability below the threshold. For example, if one move has probability 0.99 and the second move has probability 0.01, we can ignore the second move, which halves the time and space required. This technique also helps get rid of noise, non-zero valued actions that are not optimal.

After choosing at most b actions with at least ϵ probability in each information state, we apply softmax on the reduced probability set to normalize the values and use this simple probability distribution as the player's policy.

We tested different parameters to get the best results. For the first player to get 0.153 win rate against the approximate best response we use $(b = 8, \epsilon = 0.1)$. For the second player; to get 0.719 win-rate against true best response, we use ($b = 2, \epsilon = 0.1$). We then get 0.757 win rate against the approximate best response using ($b = 8, \epsilon = 0.1$).

Figures 4.6, 4.7 and 4.8 shows SIP as Black playing against the abstract best response strategy. We provide the figures and after each one the discussion of the differences between SIP and the Handcrafted Player (HP).

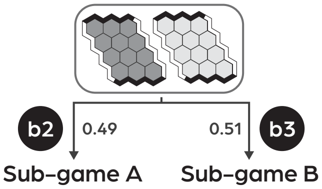


Figure 4.6: The SIP strategy for Black. Figures 4.7 and 4.8 show the continuations of the strategy.

In Figure 4.6 SIP is almost identical to HP, which selects $b2, b3$ with 0.5 probability each. Since SIP gives a slightly higher probability to $b3$, the abstract best response player goes against Sub-game B by starting the game with $b2$.

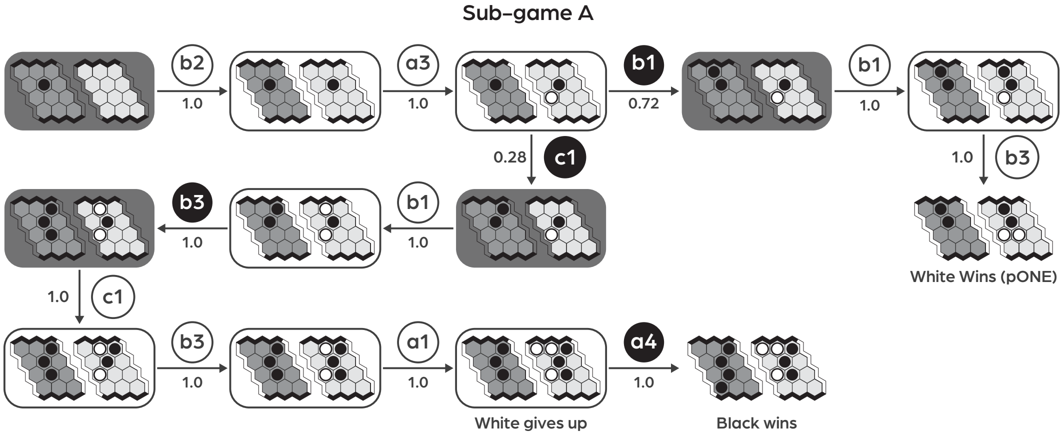


Figure 4.7: Continuation of Figure 4.6.

In Figure 4.7, Black prioritizes the short distance over the long one and connects with the top border using $b1$, $c1$. This gives Black a chance to win on either side of the board. If White plays on top Black will win, otherwise Black still has a chance to compete. Here we see the biggest difference with HP. In HP, we prioritize $c3$, $a4$ over the $b1$, $c1$ connection. SIP however completely ignores these moves at the start.

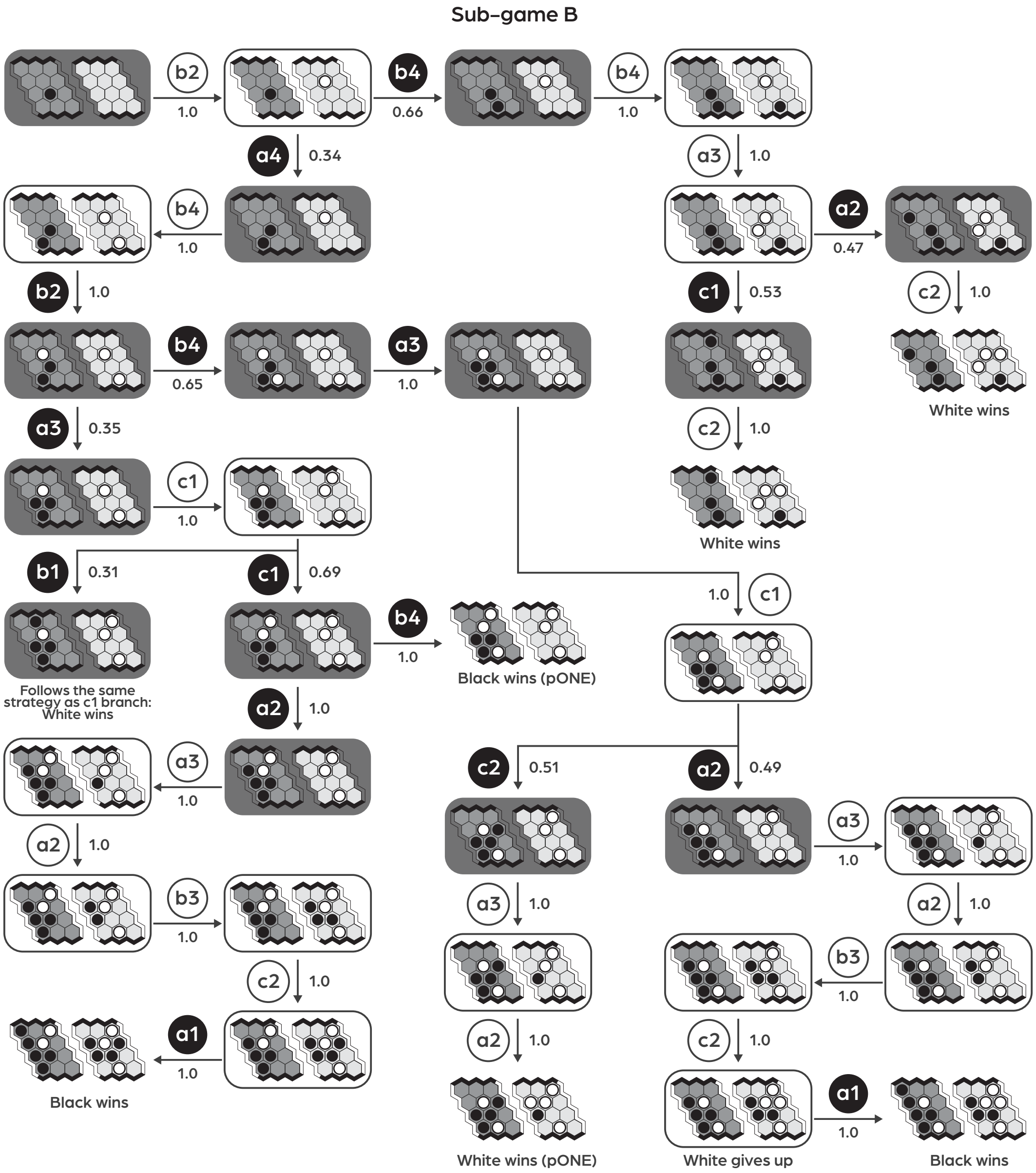


Figure 4.8: Continuation of Figure 4.6.

In Figure 4.8 SIP again focuses on the shorter distance connection with $a4$ and $b4$. The $b4$ branch leads to a loss for SIP, but the followed strategy is clear. The focus is on $a2$, $c1$, as in HP. The $a4$ branch however follows a different strategy after the discovery of white stone on $b2$. SIP plays $a3$ or $b4$. Its choice of $a3$ looks wise considering that it puts White in a fork. White needs to play either the right or the left side, which leaves the other side open. For HP, we did not consider this connection, instead, we randomly selected a side for White before making a move on $a3$. This could be one of the major differences in playing strength.

$b4$ is a less intuitive move. Black places a random stone which leads to discovering a white stone. This move leads to a pONE state. Black assumes that there is a white stone on $b4$. Otherwise, the game is already over since White has a connection on one side and two cells to connect on the other side. Here Black determines if the game will still go on and if so Black gains knowledge of a white stone, which later leads to a pONE win for Black. Other than this move, the $b4$ branch follows the same strategy as the $a3$ branch.

Figures 4.9, 4.10 and 4.11 shows SIP as White playing against abstract best response strategy.

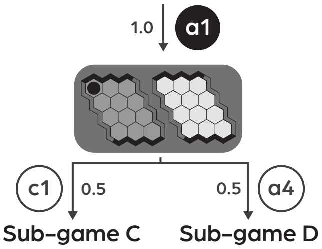


Figure 4.9: The SIP strategy for White. Figure 4.10 and 4.11 show the continuations C and D.

In Figure 4.9 we see the same opening as in HP, equiprobably playing $c1$ and $a4$. The abstract best response player opens the game with $a1$ to block White on the top row just as we explained in blocking in HP in Section 3.4.2.

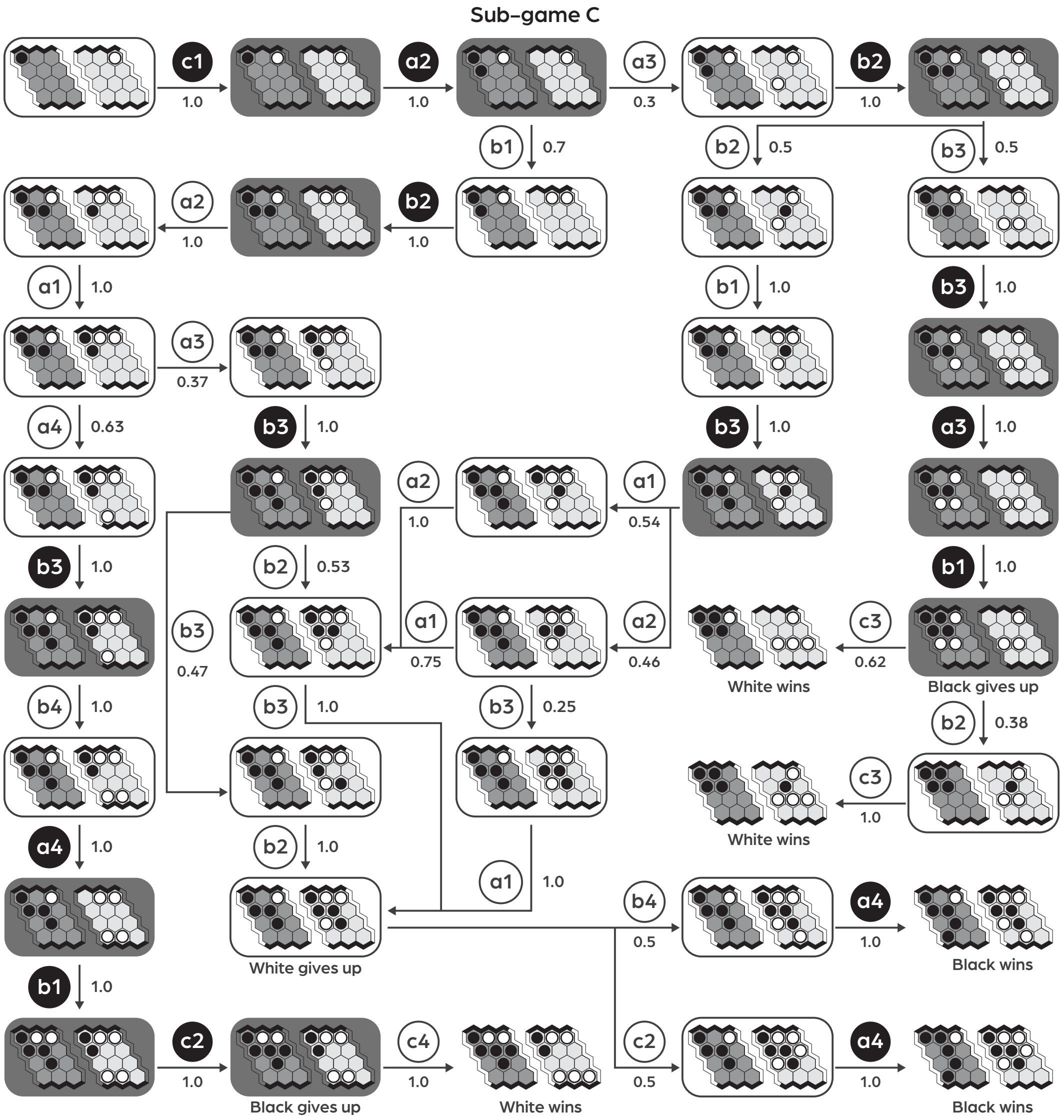


Figure 4.10: Sub-game C. Note the two Black wins in the bottom right corner.

In Figure 4.10 White plays $b1$ or $a3$ as the second move. $b1$ follows the same strategy as HP: Following the edge to join its sides until it gets blocked. After the block, as in HP it plays $a3$, $a4$ and follows the bottom edge. If the second move is $a3$, SIP tries an early connection on $c1$, $b2$, $a3$. We do not have this strategy incorporated in HP.

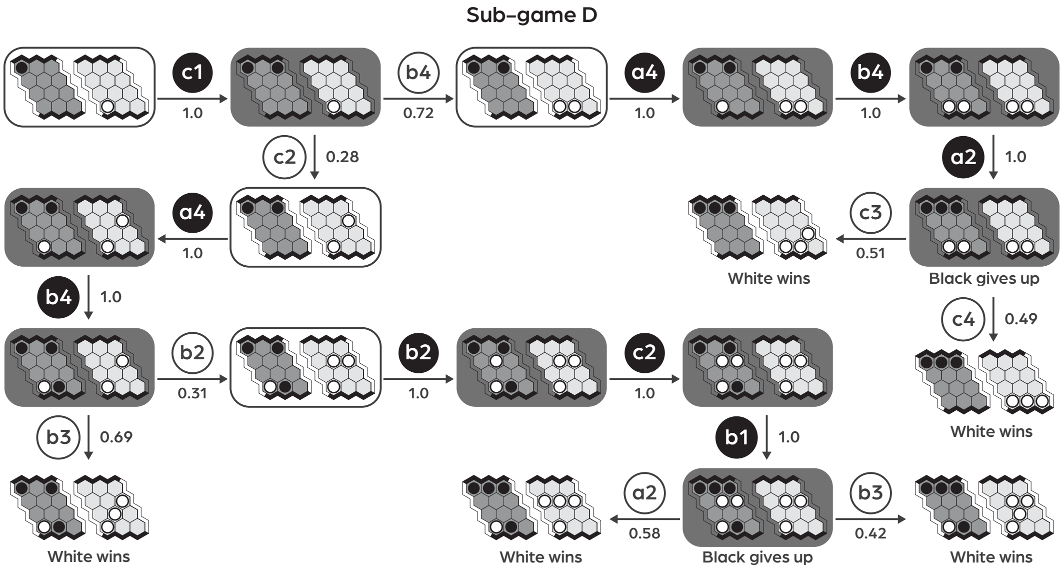


Figure 4.11: Sub-game D. Black cannot win. Black wasted two moves at the beginning on the top when White plays $a4$ so White wins on the bottom.

In Figure 4.11 SIP follows the same strategy as sub-game C but this time without getting blocked.

To evaluate its strength, we add SIP in the arena from Section 4.5.

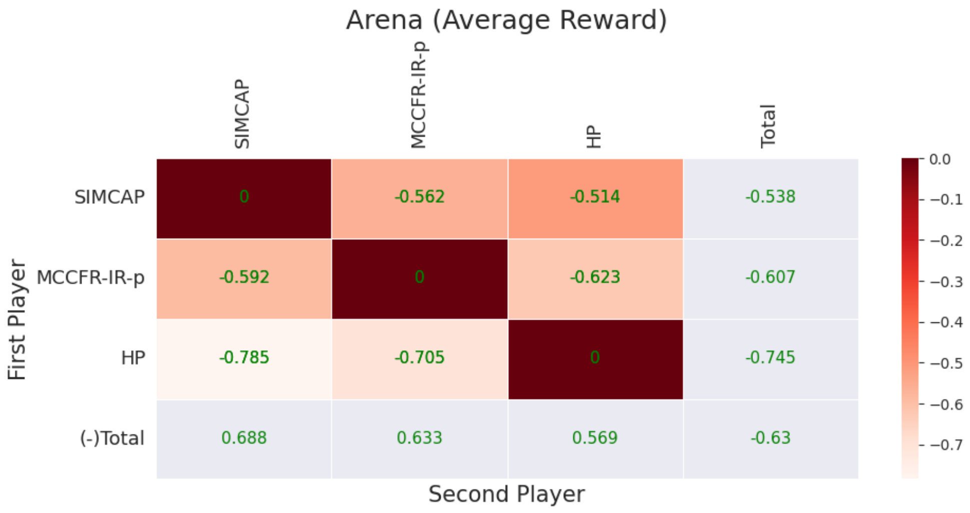


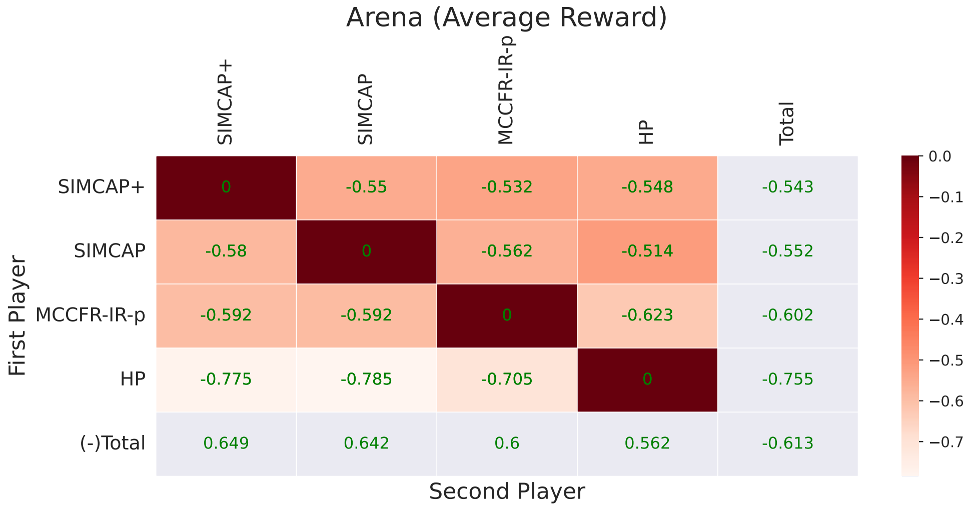
Figure 4.12: Ranking for players including SIP.

4.7 SIP+: Smooth SIP

On the first look, we realize that some of the probabilities in SIP are close to simple fractions such as $\frac{1}{2}$. We hypothesize that these probabilities should be rounded systematically to a near simple fraction.

We call this "*smoother*" version, SIP+. We add two new parameters that control the smoothing operation. The first parameter is N, the maximum fraction denominator we look for. The second parameter we introduce is η, the maximum distance from the probability to a fraction where we smooth out. In the case of multiple fractions satisfying the smoothing constraints, we choose the one closest to the input probability.

Using SIP+ we get improved results for the first player, increasing the winning probability from 0.786 to 0.793 against the abstract best response, and bringing our ϵ for the Nash equilibrium to $(1 - 0.793 - 0.205) = 0.002$ as our final result in abstract space. For these results we use $(N = 20, \eta = 0.005)$. Same strategy gives us 0.202 and 0.757 win rate against approximate best response, which gives us $(1 - 0.757 - 0.202) = 0.049$ as the final epsilon in the full game. SIP+ is the strongest player in our findings.



	SIMCAP+	SIMCAP	MCCFR-IR-p	HP	Total
SIMCAP+	0	-0.55	-0.532	-0.548	-0.543
SIMCAP	-0.58	0	-0.562	-0.514	-0.552
MCCFR-IR-p	-0.592	-0.592	0	-0.623	-0.602
HP	-0.775	-0.785	-0.705	0	-0.755
(-)Total	0.649	0.642	0.6	0.562	-0.613

Figure 4.13: Ranking for players including SIP+.

Chapter 5

Conclusion

In this thesis, we have evaluated the imperfect information game of Dark Hex. We focused on four main questions about 4×3 Dark Hex:

1. What is the game, what are some interesting variations and why is it difficult?
2. What are some techniques that we can use to analyze the game?
3. What is the best handcrafted strategy we can generate using deduction and basic evaluation?
4. How can we improve this strategy using AI methods?

We briefly introduced some versions of Dark Hex and described the rules of the game. We then gave an explanation on an improved mixed strategy generated by hand and with the help of tools such as DSaGe. We introduced a new mathematical notation which can describe a strategy concisely. We discussed the reasoning behind every move in the strategy in a game against the abstract best response player.

After creating a good handcrafted strategy we focused on using a computer based approach. We first used sure win states to prune the game tree to a more manageable level. We introduced Imperfect Recall abstracted Dark Hex which reduced the game size by a large margin. After having a workable sized game, we used the algorithms, NFSP and MCCFR to generate players. We compared these players and chose MCCFR to further work with. We introduced a new simplification method, SIP, to get rid of the actions we think would be irrelevant. The new player generated by SIP, improved upon the handcrafted strategy. We then realized some of the probabilities are close to simple fractions, and introduced two new parameters to smooth the probabilities where necessary. The improved player generated by SIP+, achieved an ϵ-Nash equilibrium with $\epsilon = 0.002$ in abstract game and got $\epsilon = 0.049$ against approximate best response. For each step of improvements we also discussed the differences between the new strategy and the previous one.

We incrementally improved the known strategies for 4×3 Dark Hex ending up with an end-to-end approach to be able to generate new strategies for large games such as Dark Hex.

5.1 Future Work

Based on our work, we describe potential follow-ups.

1. **Other Dark Hex versions.** We introduced four different versions of Dark Hex but focused only on classical Dark Hex. Investigating the same methods on different versions is one natural next step in our work. Since for the versions introduced in this thesis there are no baseline strategies, any strategy would be an improvement.

2. **Simplified MCCFR.** We simplified the MCCFR trained strategies for our best results. This idea should be generalized into a useful algorithm to implement and test on large and small games that uses fewer actions, and simpler probabilities. The embedding of simplification ideas in MCCFR is not straight-forward, since MCCFR needs to explore further in order to strengthen the player over time. We think that it is a very interesting future research topic.

3. **Larger boards.** The given methods are anytime and could be applied to larger boards, such as 4×4. With a more efficient implementation of pONE, it is possible to implement exactly the same approach on a 4×4 board and investigate the strategies.

4. **Parameter study.** The performance of NFSP in our experiments as surprisingly bad. Even though we did some preliminary parameter search and network structure tests, it might be worthwhile to do an in depth parameter search for better results with NFSP.

References

[1] J. McCarthy, "From here to human-level AI," *Artificial Intelligence*, vol. 171, no. 18, pp. 1174–1182, 2007. 1

[2] R. Sutton, *The Definition of Intelligence*, 2016. [Online]. Available: `http://incompleteideas.net/IncIdeas/DefinitionOfIntelligence.html`. 1

[3] N. Brown and T. Sandholm, "Superhuman AI for multiplayer poker," *Science*, vol. 365, no. 6456, pp. 885–890, 2019. 1, 4

[4] M. Campbell, A. J. Hoane Jr, and F.-h. Hsu, "Deep Blue," *Artificial Intelligence*, vol. 134, no. 1-2, pp. 57–83, 2002. 1, 4

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015. 1

[6] D. Silver, T. Hubert, J. Schrittwieser, *et al.*, "A general reinforcement learning algorithm that masters Chess, Shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018. 1, 13

[7] D. Silver, J. Schrittwieser, K. Simonyan, *et al.*, "Mastering the game of Go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017. 1, 13

[8] O. Vinyals, I. Babuschkin, W. M. Czarnecki, *et al.*, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019. 1

[9] G.-W. Wei, "Protein structure prediction beyond AlphaFold," *Nature Machine Intelligence*, vol. 1, no. 8, pp. 336–337, 2019. 1

[10] R. B. Hayward and B. Toft, *Hex, inside and out: the full story*. CRC Press, 2019. 3, 14

[11] D. Gale, "The game of Hex and the Brouwer fixed-point theorem," *The American Mathematical Monthly*, vol. 86, no. 10, pp. 818–827, 1979. 3

[12] F. Bonnet, "Winning strategies in Dark Hex: Hex with hidden stones," *ICGA Journal*, vol. 40, no. 3, pp. 234–245, 2018. 3, 6, 31, 45

[13] M. Johanson, N. Bard, M. Lanctot, R. G. Gibson, and M. Bowling, "Efficient Nash equilibrium approximation through Monte Carlo counterfactual regret minimization.," in *AAMAS*, Citeseer, 2012, pp. 837–846. 4

[14] F. Timbers, E. Lockhart, M. Lanctot, *et al.*, "Approximate exploitability: Learning a best response in large games," *arXiv preprint arXiv:2004.09677*, 2020. 4

[15] C. E. Shannon, "A chess-playing machine," *Scientific American*, vol. 182, no. 2, pp. 48–51, 1950. 4

[16] G. Tesauro, "TD-Gammon, a self-teaching backgammon program, achieves master-level play," *Neural computation*, vol. 6, no. 2, pp. 215–219, 1994. 4

[17] D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of Go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016. 4, 13

[18] S.-C. Huang, B. Arneson, R. B. Hayward, M. Müller, and J. Pawlewicz, "MoHex 2.0: a pattern-based MCTS Hex player," in *International Conference on Computers and Games*, Springer, 2013, pp. 60–71. 4, 13

[19] D. Koller, N. Megiddo, and B. Von Stengel, "Fast algorithms for finding randomized strategies in game trees," in *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, 1994, pp. 750–759. 4

[20] H. Kuhn and A. Tucker, "Extensive games and the problem op information," *Contributions to the Theory of Games*, pp. 193–216, 2016. 4

[21] O. Morgenstern and J. Von Neumann, *Theory of games and economic behavior*. Princeton University Press, 1953. 4

[22] G. J. Gordon, "No-regret algorithms for structured prediction problems," Carnegie Mellon University, Tech. Rep., 2005. 4

[23] D. Billings, "Computer poker," Ph.D. dissertation, University of Alberta, 1995. 4

[24] J. Schaeffer, D. Billings, L. Peña, and D. Szafron, "Learning to play strong poker," in *The International Conference on Machine Learning Workshop on Game Playing*, 1999. 4

[25] L. Barone and L. While, "Evolving adaptive play for simplified poker," in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence*, IEEE, 1998, pp. 108–113. 4

[26] G. Kendall and M. Willdig, "An investigation of an adaptive poker player," in *Australian Joint Conference on Artificial Intelligence*, Springer, 2001, pp. 189–200. 4

[27] D. Koller and A. Pfeffer, "Representations and solutions for game-theoretic problems," *Artificial intelligence*, vol. 94, no. 1-2, pp. 167–215, 1997. 4

[28] J. Shi and M. L. Littman, "Abstraction methods for game theoretic poker," in *International Conference on Computers and Games*, Springer, 2000, pp. 333–345. 4, 5

[29] N. Brown and T. Sandholm, “Superhuman AI for heads-up no-limit poker: Libratus beats top professionals,” *Science*, vol. 359, no. 6374, pp. 418–424, 2018. 4

[30] *The second man-machine poker competition*, 2008. [Online]. Available: `http://webdocs.cs.ualberta.ca/~games/poker/man-machine/`. 4

[31] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, “Regret minimization in games with incomplete information,” *Advances in Neural Information Processing Systems*, vol. 20, 2007. 4, 44

[32] M. Lanctot, K. Waugh, M. Zinkevich, and M. H. Bowling, “Monte Carlo Sampling for Regret Minimization in Extensive Games.,” in *NIPS*, 2009, pp. 1078–1086. 4, 48

[33] O. Tammelin, “Solving large imperfect information games using CFR+,” *arXiv preprint arXiv:1407.5042*, 2014. 4

[34] K. Waugh, D. Morrill, J. A. Bagnell, and M. Bowling, “Solving games with functional regret estimation,” in *Twenty-ninth AAAI conference on artificial intelligence*, 2015. 4

[35] D. R. Morrill, “Using regret estimation to solve games compactly,” M.S. thesis, University of Alberta, 2016. 4

[36] N. Brown, A. Lerer, S. Gross, and T. Sandholm, “Deep counterfactual regret minimization,” in *International conference on machine learning*, PMLR, 2019, pp. 793–802. 4

[37] K. Waugh, D. Schnizlein, M. H. Bowling, and D. Szafron, “Abstraction pathologies in extensive games.,” *AAMAS (2)*, vol. 2009, pp. 781–8, 2009. 5

[38] D. Billings, N. Burch, A. Davidson, *et al.*, “Approximating game-theoretic optimal strategies for full-scale poker,” in *IJCAI*, vol. 3, 2003, p. 661. 5

[39] M. B. Johanson, “Robust strategies and counter-strategies: Building a champion level computer poker player,” M.S. thesis, University of Alberta, 2007. 5

[40] A. Gilpin, T. Sandholm, and T. B. Sørensen, “Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of Texas Hold’em poker,” in *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 1*, ser. AAAI’07, Vancouver, British Columbia, Canada: AAAI Press, 2007, pp. 50–57, ISBN: 9781577353232. 5

[41] M. Lanctot, R. Gibson, N. Burch, M. Zinkevich, and M. Bowling, “No-Regret Learning in Extensive-Form Games with Imperfect Recall,” *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, vol. 1, May 2012. 5

[42] N. Brown and T. Sandholm, “Libratus: The Superhuman AI for No-Limit Poker,” in *IJCAI*, 2017, pp. 5226–5228. 13

[43] B. Arneson, R. B. Hayward, and P. Henderson, “Monte Carlo tree search in Hex,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 4, pp. 251–258, 2010. 13

[44] ——, “Solving Hex: beyond humans,” in *International Conference on Computers and Games*, Springer, 2010, pp. 1–10. 13

[45] P. Henderson, B. Arneson, and R. B. Hayward, “Solving 8x8 Hex,” in *Twenty-First International Joint Conference on Artificial Intelligence*, 2009. 13

[46] R. Hayward, Y. Björnsson, M. Johanson, M. Kan, N. Po, and J. van Rijswijck, “Solving 7× 7 Hex: Virtual connections and game-state reduction,” in *Advances in Computer Games*, Springer, 2004, pp. 261–278. 13

[47] J. Hannan, “Approximation to Bayes risk in repeated play,” *Contributions to the Theory of Games*, vol. 3, no. 2, pp. 97–139, 1957. 16

[48] S. Hart and A. Mas-Colell, “A simple adaptive procedure leading to correlated equilibrium,” *Econometrica*, vol. 68, no. 5, pp. 1127–1150, 2000. 16, 19

[49] A. Blum and Y. Monsour, “Learning, regret minimization, and equilibria,” 2007. 16

[50] M. Lanctot, “Monte carlo sampling and regret minimization for equilibrium computation and decision-making in large extensive form games,” Ph.D. dissertation, University of Alberta (Canada), 2013. 20, 21, 23, 44

[51] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992. 22

[52] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, “Human-level control through deep reinforcement learning,” *nature*, vol. 518, no. 7540, pp. 529–533, 2015. 22

[53] J. Heinrich and D. Silver, “Deep reinforcement learning from self-play in imperfect-information games,” *arXiv preprint arXiv:1603.01121*, 2016. 22, 48

[54] G. A. Heuer, T. Stratton, D. A. Smith, *et al.*, “Problems,” *Mathematics Magazine*, vol. 54, no. 2, pp. 84–87, 1981, ISSN: 0025570X, 19300980. [Online]. Available: `http://www.jstor.org/stable/2690443` (visited on 04/27/2022). 31

[55] M. Lanctot, E. Lockhart, J.-B. Lespiau, *et al.*, “OpenSpiel: A Framework for Reinforcement Learning in Games,” *CoRR*, vol. abs/1908.09453, 2019. arXiv: `1908.09453 [cs.LG]`. [Online]. Available: `http://arxiv.org/abs/1908.09453`. 48, 50

Appendix A

Appendix

A.1 Analysis on p-one

In this section we provide statistics on p-one for board sizes of 2×2, 3×3, 4×3. We report the number of p-one states for both players based on the number of hidden stones in an information state.

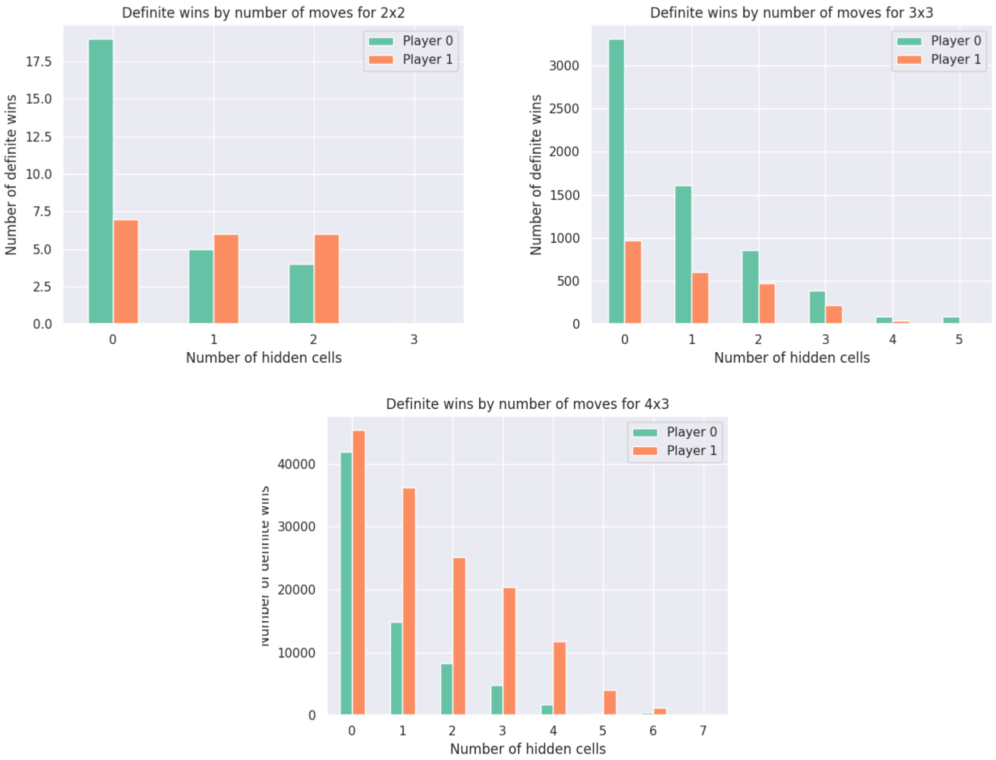


Figure A.1: Number of p-one for each player given the number of hidden stones.

We can see that the the boards of equal size have the first player in advantage. In 4×3 board however, the second player has the clear advantage.