Dark-Hex:
An Imperfect Information Game

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

Master's in Computing Science

University of Alberta

May 10, 2021

UNIVERSITY OF ALBERTA

M Bedir Tapkan

Abstract

Dark-Hex is the imperfect information version of the game Hex. In this work, we have examined the game's interesting properties and provided the solution to some of the small board sizes. We were mainly interested in an approximate solution and a definitive player. We got Nash solution on small boards using pure strategy LP on different game modes. This allowed us to explore and analyze the properties of the different versions of the games we describe here. Next, we evaluated Sequence-LP on different board sizes, the results showed that for bigger board sizes we needed more approximate solutions. For the last part of our evaluation, we worked with the well-known algorithm for partial information environments; Counterfactual Regret Minimization. We did analyze vanilla CFR alongside evolved versions of it, RCFR, f-RCFR. We lastly present an increment to f-RCFR ¡appendName¿ ... ¡appendDifferences¿. The end of the thesis is our player. We developed a state-of-the-art player **(not sure if we can call it state of the art as there is no player exist for the domain at the moment)**. In this chapter, we have trained some classical Reinforcement Learning players, vanilla CFR, RCFR, f-RCFR, ¡appendName¿ and some other popular algorithms **(Should include some combination etc. as well, I cannot name what exactly as I need to work on them before saying anything)**. We got the benchmark by setting a league-like system and constantly competing with the players. Our player ¡appendName¿ got the top place **like this...**.

1 Introduction

Section structure is temporary

1.1 Definitions (Maybe on background instead)

Game — Game in Game Theory is defined as a set of states which has a result dependent on the actions of the players. Since we are mostly concerned on computational needs and limitations; we will accept a slightly different definition instead. A **game** is a series of actions taken by two or more players which result in a different state for every action. This definition is more complete for our purposes, since we consider information state rather then individual board states. For example;

$$\text{player2} = \left\{ \begin{matrix} . & . \\ . & \text{W} \end{matrix} \right\} \neq \left\{ \begin{matrix} \text{B} & . \\ . & \text{W} \end{matrix} \right\}$$

Here player2(W)'s information state changes depending if the move tried hits a black stone prior to a legal move or not. **??**

Strategy — An action plan for a player which includes an answer for every possible opponent move or game state.

2 Background

Here we will build up the knowledge we need to read the rest of the thesis. We start with the notations and definitions we use. Mainly we have three major topics; Combinatorial Game Theory (CGT), Reinforcement Learning (RL), Counterfactual Regret Minimization (CFR). We will follow this path in this section too, first we will explain extensive form and imperfect-information games. Then we will explain Hex and Dark Hex briefly. As Dark Hex is not well documented and very scarce in the literature, we count it as a contribution, therefor there will be a seperate chapter discussing the properties and details of the game. After the games we will introduce RL and the RL algorithms we used. Lastly we will have a CFR section which includes the explanation of CFR and all it's variants we found useful for our experiments.

2.1 Combinatorial Game Theory (CGT)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent

euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

2.2 Reinforcement Learning (RL)

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

2.3 Counterfactual Regret Minimization (CFR)

In this section, we will describe the Counterfactual Regret Minimization algorithm. We will first start with terminology and definitions, continue with the regret in a simple non-sequential game of Rock Paper Scissors. We will then proceed and explain the algorithm in its pure (vanilla) form before we go into variants such as MCCFR, RCFR, Deep CFR, and so on. We either used or tried to use the included variants in the experiments, therefore it is essential to understand the differences between them. That's why we also included a section on the differences between these variants, and which one is better to use in which situations, and why. Throughout the section we will be using the earlier definitions from the extensive form games and RL sections.

2.3.1 What is Regret and Regret Minimization

Consider yourself making an important decision; you will either study for your exam tomorrow or go out with your friends who seem to be eager to hang out with you. Here we could make a decision based on what we think is the best for us, and in the end, get a reward depending on our choice. We can intuitively think about what the regret will be. If we go out with friends, let's call it event f, we will get a reward r_f and otherwise, if we study, let's call it event s, we get a reward r_s. Regret will be the reward we got from the action chosen minus the reward we could get if we have chosen the other action. In our case it will be -if we went out- $R = r_s - r_f$. Simply how much do we **regret** not choosing the other action?

Now let's add a third option, going home and watching a movie and call it m. Now how do we calculate regret? The cumulative regret will follow from the first part, and we sum the regrets we have from all simultaneous choices we didn't make;

$$R = \sum_{i \in E} r_f - r_i$$

where E is the number of events we have and r_f is the reward of the event we chose.

The same logic goes for regret in games, every time we make a decision and take an action we prioritize it over the others, which will let us calculate the cumulative regret. We then can use this regret to optimize our actions if that situation was to occur again, hence **minimizing the regret**.

2.3.2 A Better Strategy

We can use the notion of regret and self-simulated play to make our actions better in future states of the game. We will do this by minimizing expected regret in

the game over time. Here we will consider the normal form zero-sum two-player simultaneous-move game of Rock-Paper-Scissors (RPS) because of its simplicity and being non-sequential (The round we are in, is not affecting the future rounds). This simple method will give us the Nash equilibrium if we run enough iterations.

The RPS game is a simple three-action game, each action has a counteraction, and an action it counters. Rock beats Scissors, Scissors beats Paper, and Paper beats Rock. The players choose an action simultaneously. Let the utility be the net gain/loss we make, assuming the players bet a dollar for each round. The player who wins the round gains $+1$ while the other player gains -1 or loses $+1$.

So in a simple turn that we play scissors and the opponent plays rock, our gain will be -1. We will be regretting not playing rock but regret even more for not playing paper. This is because our gain would have been even higher if we did play paper. So the definition of the **regret** is the difference between the utility of not choosing an action and the utility of choosing an action.

In more technical terms, let's call a_i action player i took, and a_{-i} for all other players actions, where $a \in \mathcal{A}$ is the action profile. Let's also call $u(a_i, a_{-i})$ the utility for player i taking action a_i and the other players taking actions a_{-i}. Let $a_i^{'}$ be an action in A player i did not take. Then the regret for picking a_i over $a_i^{'}$ for player i is $u(a_i^{'}, a_{-i}) - u(a_i)$.

We can use this regret mechanism to evolve our future decisions and make them better. For that, we could abuse our knowledge and emphasize the play on the move that brought us the least regret so far, i.e. play scissors because you played rock for the last round and received a regret of 2 (opponent plays paper) and now you don't want to regret so you should play scissors. This could work in a sense, but the opponent might exploit our selections, so we do not want to be predictable. Here we have **regret matching** to help us overcome this issue.

Regret Matching: Choosing actions randomly according to the positive regret distribution. Positive regrets shows us the relative loss for not choosing an action. Therefor we use regret matching and choose the next action proportional to the relative regret, for each action that will be the regret over their sum. For example for picking scissors when opponent picked rock our relative regrets will be $0, \frac{1}{3}, \frac{2}{3}$ for scissors, rock and paper respectively.

If we show one more round, and let's say we play paper this time, and opponent plays scissors. We will have regrets 0, 1, -1 respectively for scissors, rock

and paper. Adding the positive regrets to the previous ones we now see that we have $0, \frac{2}{4}, \frac{2}{4}$ as a strategy.

Dark Hex

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

3 Number of games for 2x2 board

How many possible games are there for Dark Hex games? What will be our search space? How can we make sure that we capture the bounds correctly? While discussing this all, we have come to realize that even for the smallest boards it is not visible really quick how big the search space or game strategies will be. The easiest way to grasp a concept is to go with an example and tackle a problem along the way. So here we will discover the number of possible strategies on a 2 by 2 board.

3.1 Without end game rules

What we are considering is not an optimal play, so that we can evaluate every possible pure strategy rather than dominant strategies. For the sake of seeing the bigger picture, we will first make an assumption; a game is not over unless the board is filled. On a 2 by 2 board this means there will be exactly 4 moves executed.

First move: b_1

The first move has no rejection possibility therefore the move will for sure go through. Since we do not care about the win at the moment (we only care about filling the board) we won't be checking the cells separately as advantageous or not. We also will ignore isomorphism. These conditions give us equally important 4 cells on the board. So Black's first move can be on any of these cells, and since all of them have the same value for us, we will examine only one, and results will multiply accordingly. We will call the first Black move b_1. Let's call Black's information state B and White's information state W.

$$B = \{b_1\}$$
$$W = \{\}$$

Second move: w_1

The second move divides into two; if white hits the b_1 or not. They both will have three possible moves (hits b_1, and moves; or moves one of the other three cells). The information state differs depending on the rejection happening or not, therefore the two moves will have different continuations, let's call the strategy with rejection S_r and without rejection S_n.

$$B_{S_r} = \{b_1\}$$
$$W_{S_r} = \{b_1, w_1\} \qquad \text{or}$$
$$W_{S_n} = \{w_1\}$$

Third move: b_2

The third move is the same as the second move, only this time for black. Black will make its move either finding w_1 or making a move directly. Gives us 2 move possibilities for each.

$$B_{S_r(1)} = \{b_1, b_2\} \quad \text{or}$$
$$B_{S_r(2)} = \{b_1, w_1, b_2\}$$
$$W_{S_r} = \{b_1, w_1\}$$

Last move: w_2

The last move is where it gets a little more complicated. Depending on what white's prior knowledge is, we have to have separate the results. There are 5 possible W's for the strategy S_n and 2 for S_r.

$$W_{S_n} = \{w_1, w_2\} \quad \text{or}$$
$$W_{S_n} = \{w_1, b_1, w_2\} \quad \text{or}$$
$$W_{S_n} = \{w_1, b_2, w_2\} \quad \text{or}$$
$$W_{S_n} = \{w_1, b_1, b_2, w_2\} \quad \text{or}$$
$$W_{S_n} = \{w_1, b_2, b_1, w_2\}$$

$$W_{S_r} = \{b_1, w_1, w_2\} \quad \text{or}$$
$$W_{S_r} = \{b_1, w_1, b_2, w_2\}$$

So we have;

$$4 \times 3 \times (2 \times 2) \times 5 +$$
$$4 \times 3 \times (2 \times 2) \times 2 = 336$$

Here we have two products; one for S_n and one for S_r, which are the same for the prior part since the only difference is on the second move where white either hits the black stone or not, and there are 3 different moves for both cases. We have resulted that 336 is the maximum number of games if we assume that the game doesn't end until the board is full. Now let's prune this tree by adding the end game rules.

3.2 With end game rules

It is a bit more manual to calculate the number of games when including the rules. We will examine the moves considering isomorphic properties, which will allow

us to calculate the games as a whole when we can benefit. For the initial move for example we have 4 moves, but 2 isomorphic positions (0 is the same as 3, and 1 is the same as 2). So for the first move, we will only consider two positions -close corner(2) and far corner(0)-.

3.2.1 First move to far corner

We have two far corners, and they are isomorphic. That's why investigating one and multiplying the result by 2 is going to suffice. Let's assume we moved to 0 as the first move.

- **Second move to 1** We have two options after this move, either black finishes the game by moving to 2 or game goes to 4 moves if black goes for cell 3.
  - B win $\binom{1}{1}\binom{2}{1}\binom{2}{1}$
  - B loss $\binom{1}{1}\binom{1}{1}\binom{2}{1}\binom{7}{1}$
- **Second move to 2** Here it's easy to see that white will win no matter what black's last move is. This means this branch will expand all the way down (4 moves).
  - B loss $\binom{1}{1}\binom{1}{1}\binom{4}{1}\binom{7}{1}$
- **Second move to 3** We have again win-loss chance for both players. If black moves 2, it's a win (3 moves game) if not it will be a full branching (4 moves).
  - B win $\binom{1}{1}\binom{2}{1}\binom{2}{1}$
  - B loss $\binom{1}{1}\binom{1}{1}\binom{2}{1}\binom{7}{1}$

In the end for the far corner first move:

$$\binom{2}{1}\binom{2}{1}+\binom{2}{1}\binom{7}{1}+\binom{4}{1}\binom{7}{1}+\binom{2}{1}\binom{2}{1}+\binom{2}{1}\binom{7}{1}=64$$

3.2.2 First move to close corner

The same goes here, we will examine only cell 2.

For the second move, white has two different options; either play to the virtual connection black has, or play next to the black move (and lose). So if the branch continues on VC the game might go on and white still has a probability to win, otherwise, the game ends on 3 moves for sure since both moves give black the connection.

- **Second move to VC** If next Black move is to VC black will win, otherwise black let's white to win in a 4 moves game.
  - B win $\binom{1}{1}\binom{4}{1}\binom{2}{1}$
  - B loss $\binom{1}{1}\binom{3}{1}\binom{2}{1}\binom{7}{1}$
- **Second move to 3** Black wins for sure since it owns the virtual connection and either remaining moves result in a win.
  - B win $\binom{1}{1}\binom{1}{1}\binom{4}{1}$

$$\binom{4}{1}\binom{2}{1}+\binom{3}{1}\binom{2}{1}\binom{7}{1}+\binom{4}{1}=54$$

By the demonstration of every possible game we now see the result as $2\cdot(64+54)=236$.

4 Abrupt Dark Hex Practices

Here is the full pay-off matrix. I think 0-1 without any context looks very much confusing. As far as I can see if the first move of the player B is not blocking the first players second move strategy it's a win otherwise it's a loss. So; Couple other observations:

1. Diagonal is always first player win, since the strategies are the same second player will get rejected both moves and lose.
2. Pattern is the same for each strategy, first players second move is the only important move. If the move is

	01	02	03	10	12	13	20	21	23	30	31	32
01	1, 0	1, 0	1, 0	0, 0	0, 0	0, 1	1, 0	1, 0	1, 0	1, 0	1, 0	1, 0
02	0, 0	0, 0	0, 0	0, 0	0, 0	0, 1	0, 0	0, 0	0, 0	0, 1	0, 0	0, 0
03	0, 0	0, 0	0, 0	0, 0	0, 1	0, 0	0, 0	0, 1	0, 0	0, 0	0, 0	0, 0
10	0, 0	0, 0	0, 0	1, 0	1, 0	1, 0	1, 0	1, 0	1, 0	1, 0	1, 0	1, 0
12	1, 0	1, 0	1, 0	1, 0	1, 0	1, 0	0, 0	0, 0	0, 0	1, 0	1, 0	1, 0
13	0, 0	0, 1	0, 0	0, 0	0, 0	0, 0	0, 1	0, 0	0, 0	0, 0	0, 0	0, 0
20	0, 0	0, 0	0, 0	0, 0	0, 0	0, 1	0, 0	0, 0	0, 0	0, 0	0, 1	0, 0
21	1, 0	1, 0	1, 0	0, 1	0, 0	0, 0	1, 0	1, 0	1, 0	1, 0	1, 0	1, 0
23	1, 0	1, 0	1, 0	1, 0	1, 0	1, 0	1, 0	1, 0	1, 0	0, 0	0, 1	0, 0
30	0, 0	0, 0	0, 0	0, 0	0, 1	0, 0	0, 0	0, 1	0, 0	0, 0	0, 0	0, 0
31	0, 0	0, 1	0, 0	0, 0	0, 0	0, 0	0, 1	0, 0	0, 0	0, 0	0, 0	0, 0
32	1, 0	1, 0	1, 0	1, 0	1, 0	1, 0	0, 0	0, 1	0, 0	1, 0	1, 0	1, 0

5 Notes

5.1 Dark Hex Versions

Dark Hex is a very poorly researched topic as far as I see. First thing to do seems to be to bring out the interesting points of the game, why is it worth researching on, what properties lays underneath?

The rules of the game is hardly described anywhere, therefor it seems like we have a quite wide definition window. We came up with multiple versions of the game, we are not sure what will be the version we investigate at the moment. I will list the current versions discussed, and look into them in detail when deciding the interesting properties of the game. We might look into more than one versions in this thesis. It's yet to be decided.

5.1.1 Classic Dark Hex (Kriegspiel Hex)

Dark Hex is the extension for Kriegspiel Chess on Hex game. It turns the perfect information game of Hex to its imperfect version. The rules of winning and losing still stands as Classic Hex. What changes is that the players are not exposed to opponents move information. So for a player the current

state is his/her own stones on the board, the number of moves made by opponent, cells where there is no known stone, and opponent stones where the player tried to make a move and got rejected.

Rejection: If an opponent stone is on the position the player is trying to play on it will result in rejection. Rejection is not a terminal for the current player, meaning the player still needs to make a move, only this time he/she has information on one more of the opponents stones.

Example Game: TODO

5.1.2 Abrupt Dark Hex

Abrupt or short Dark Hex is also an interesting version of Dark Hex. The difference is that instead of rejection Abrupt DH has collisions. Other than that the two games follow same properties.

Collision: If an opponent already has a stone on the location the player tries to make a move, collision happens, the player who tried to make a move loses his/her turn.

5.1.3 Noisy Dark Hex

Noise is an extension for the other versions of DH. The game follows exact version of Abrupt DH or Classical DH with the exception of noise.

Noise: Knowledge of the attempted move is publicly available for both players instead of only the player who tried to make the move. i.e. Player 1 makes a move on a2, there is a Player 2 stone on a2 therefor collision or rejection happens (depending on the type of the game), Player 1 knows the exact location of where one of Player 2's stones are is, Player 2 hears 'a noise' meaning that he/she will have the information that Player 1 tried to make a move on one of his/her stones, but won't know which location that is.

5.1.4 Flash Dark Hex

Same as Noisy Dark Hex except that instead of a noise, flash takes place.

Flash: Instead of noise this time opponent gets a 'flash' for the collision or rejections. Flash's are shown on exact location where collision or rejections has happened.

6 Probability One State Calculator (PONE)

```
1: function PONE(s, h)
2:     if White's turn then
3:         if Game is over then return False
4:         else
5:             h ← h + 1.
6:         end if
7:     end if
8:     if s is Black win then return True
9:     else if s is White win then return False
10:    else
11:        if h == 0 then
12:            for all x ∈ X where X is the set of empty cells do
13:                Ensure: (s + B[x], h) is legal
14:                if PONE(s + B[x], h) then return True
15:                end if
16:            end for
17:        else if h > 0 then
18:            for all x ∈ X where X is the set of empty cells do
19:                Ensure: (s + W[x], h − 1) is legal
20:                Ensure: (s + B[x], h) is legal
21:                if PONE(s + W[x], h − 1) and PONE(s + B[x], h) then
22:                    return True
23:                end if
24:            end for
25:        end if
26:    end if
27:    return False
28: end function
```

```
1: function SEARCH-PONE
2:     n ← number of cells
3:     results ← empty dictionary
4:     for all e where e < n do
5:         for all h where h < n/2 do
6:             S ← all possible states given e and h
7:             for all s ∈ S do
8:                 if s is legal then
9:                     results[(s,h)] ← game result with s,h
10:                    if PONE(s,h) then
11:                        results[(s,h)] ← Black wins
12:                    end if
13:                end if
14:            end for
15:        end for
16:    end for
17: end function
```