2009

Is Four File Chess a Draw?

Michael Karbushev
San Jose State University

IS FOUR FILE CHESS A DRAW?

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Michael Y. Karbushev

May 2009

APPROVED FOR THE DEPARTMENT OF

COMPUTER SCIENCE

Dr. David Taylor

Dr. Richard M. Low

Dr. Teng Moh

APPROVED FOR THE UNIVERSITY

ABSTRACT

IS FOUR FILE CHESS A DRAW?

by Michael Y. Karbushev

In this work, we prove that in the game of FOUR FILE Chess, White has at least a Draw. FOUR FILE is a chess variant proposed by John Selfridge, in which only the ‘a’, ‘c’, ‘e’, and ‘g’ files are used. All chess rules are as usual, except that all moves must end on one of these files, and the game starts with the other four files vacant. Here, we prove that the White has at least a draw, by showing that White has a strategy to avoid a loss. We also show that Black can avoid a loss for ten out of eleven starting white moves and outline the steps to complete the proof that the game of FOUR FILE is a Draw.

TABLE OF CONTENTS

CHAPTER

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

Is FOUR FILE a draw? FOUR FILE is played on a chessboard with the chess pieces in their usual starting positions, but only on the 'a', 'c', 'e' and 'g'-files; i.e., a Rook, a Bishop, a King, a Knight and four pawns on each side shown in Figure 1.1. The moves are normal chess moves except that play takes place only on these four files. Because each move ends on one of the files 'a', 'c', 'e' or 'g', pawns cannot capture and there is no castling, but pawn promotion is possible. The aim is to checkmate your opponent's King [1]. The question about FOUR FILE is originally prompted by John Selfridge who specifically asks if the game is a draw.

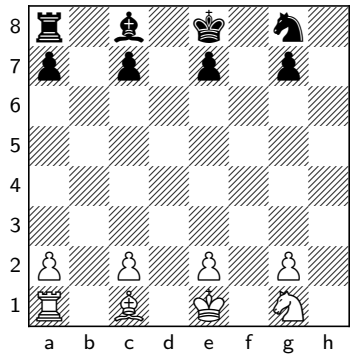

Figure 1.1: Starting Board

1.2 Results

We show that White has a strategy to avoid a loss in FOUR FILE. The idea behind the proof is that we have a list of positions that are reached in the game after White's move; in all such positions White's King is present, and from any such position, for all possible Black moves, there exists a move for White which will return to another position in the list. The list of positions is generated using a specific strategy for White, described in this paper. Once we have such a list of positions, a third party can take a list of positions and verify that no matter what move Black chooses to take; there will be a move for White to end up in one of the listed positions.

Next, we partially show that Black has a strategy to avoid a loss in FOUR FILE as well. We take the approach of reducing the problem for Black to a solved problem for White. White and Black are symmetric; hence for the purpose of having one database and rather than having a separate strategy for Black, we instead continue to play with a strategy for White but allow Black to move first. We need to consider eleven possible first moves for Black, and we have complete results for ten out of eleven.

1.3 Related Work

Although not technically a combinatorial game, the game of FOUR FILE can be qualified as a "Game of No Chance" [4] and may be analyzed using tools from combinatorial game theory [3]. A recent celebrated addition to the study of games of no chance is work done by Jonathan Schaeffer, a computer-games expert at the University of Alberta in Canada. Dr. Shaeffer proved that the game of checkers is a draw. The computer proof took 18 years to complete and is one of the longest running computations in history. [2]

We are unaware of any previous results for FOUR FILE.

1.4 Outline

We will discuss the work completed in the following order:

- Describe general approach to solving games
- How to minimize the size of FOUR FILE's game tree
- High-level description of our strategy
- Chess observations
- Implementation details
- Results
- Future work

CHAPTER 2

STRATEGY OVERVIEW

2.1 Strategy Definition

A player's strategy can be defined in multiple ways. The simplest form is to have a database of ⟨position, move⟩ tuples, so that for every position there exists a move; hence the player always knows what to do. A more sophisticated approach would be to have rules that cover all possible positions; given a position the game strategy would be to check if any of these rules apply, then make a move accordingly. We have many such rules defined for the endgame; the endgame starts after we enter one of the draw-states *(Barrier states)* described in a Section 4. Draw-states are an interemediate result that allows us to divide the proof in two stages.

It is important to note that ⟨position, move⟩ tuples are not needed for all possible positions because we control White's strategy; hence we can avoid some (most) legal positions. A simple example is if White's first move is 'a-pawn' going from 'a2' to 'a4', then we do not have to worry about any positions where 'a-pawn' is at either 'a2' or 'a3'. Another important note is that trying to show that White can avoid losing does not force us to make an absolute best move for White at all times; White does not need to force a win, even when possible, and we will sometimes choose to make sub-optimal moves for White, in order to greatly prune our game tree.

To fully prove that FOUR FILE is a draw, we consider twelve possible starting

boards:

- White goes first, all pieces are in their original positions
- We continue with using White's strategy while the original board is modified by Black taking one of the following moves:

  (1) 'a7a6'

  (2) 'a7a5'

  (3) 'c7c6'

  (4) 'c7c5'

  (5) 'e7e6'

  (6) 'e7e5'

  (7) 'g7g6'

  (8) 'g7g5'

  (9) 'c8a6'

  (10) 'c8e6'

  (11) 'c8g4'

It should be clear that Black moving first does not alter our result since White and Black are symmetric.

2.2 General Game Strategies

Let us begin by describing the usual approach one takes to analyze a game. We assume that the reader is familiar with game trees, minimax search, and the general concept of board evaluation. These are the standard techniques used by

chess, checkers abd other board games playing programs. By searching many moves ahead in minimax search, a somewhat simplistic board evaluation can lead to an effective strategy. The deeper the search, the more effective the board evaluation will be. Powerful computers are required to expertly play a game as complicated as chess. A perfect strategy would be to search the game from the beginning till the end (end being defined by a capture of the opposing King, while your King is still there). This approach would solve the problem, but is intractable.

2.3 Tractability

While the game of FOUR FILE is not nearly as complex as the game of chess, it is still not tractable in terms of a complete minimax game-tree search from the starting position. However, given a strategy for White's first moves, the search becomes more tractable. Here, White's strategy will be one which, from the start of the game, tries to force positions in which the mobility for the board in general diminishes. This not only prunes the game tree by giving White just one move to search for these positions, it will also limit the possible moves for Black.

Note that while we are looking for a strategy for White, we do not have to explore all moves for White; once we have a ⟨position, move⟩ tuple, that is the only move that White will take from the given position. As the result, it makes sense to take certain moves that are not necessarily winning moves, but they are moves that take us from one non-losing state to another non-losing state and thus decrease the number of actual positions being explored. Of course, no matter what White's strategy is, we have to explore all possible moves for Black, thus the game tree is best described by Figure 2.1.

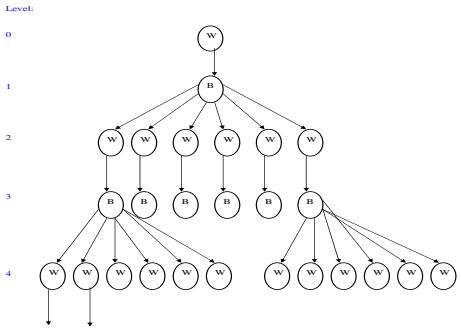

Figure 2.1: Game Tree

2.4 Pruning the Game Tree

Looking at the tree in Figure 2.1, we can see that even if we explore only one child for all W (White Nodes), the tree still gets really large. We want to minimize it further, since a smaller tree will mean a smaller set of states in our final list. To get the full White's strategy:

- Run the game simulation to depth 12 by only using *custom moves*. Basically, White's initial game strategy, and all of its initial moves, are calculated by hand. This is done to significantly reduce the size of the game-tree, both for the beginning of the game, and for later in the game. The goal here is to move the game closer to what we call a *Barrier* state, which can loosely be described as a position in which Black's pawns form a barrier which prevent Black's other pieces from attacking White.

- Enumerate all Barrier state positions (or families of positions, as many positions may be part of one Barrier state). Do this by running the program that enumerates all positions that can be part of the game tree and only store the moves for positions that can be resolved Barrier rules. Barrier states are described in Section 4.

- Run the full game simulation. Given a position in which White needs to move, calculate that move using the following sequence:

  (1) If White has a move which can take us to any position closer to the root of the game-tree such move is taken. Hence, we end up at the higher level (lower level number) position, which we already know that Black can force, and all sub-trees for this position must be explored anyway.

(2) Regular minimax pruning for win/loss positions, with depths 1, 3, 5, and 7. These searches were optimal given the time it took to run on them on the computer used. (The iterative deepening is used for efficiency.) [5]

(3) See if we are in a Barrier that does not require Black's Rook to be captured; such states are mentioned in Section 4.9. At this point White can easily force a draw. If we are, simply follow the rules of such a Barrier. *In* FOUR FILE *there are two types of barriers: one that does not allow Black to sacrifice a Rook to get out of it, and the other type that does. Here we are talking about the first type.*

(4) See if there is a move that allows White to capture Black Rook. If White has a clear rook advantage, it is easier to force a draw.

(5) See if any of the Barriers apply. Here we are looking at all Barriers (including the ones where Black can sacrifice the Rook). This is exactly why we first check if Black Rook can be captured.

(6) Try to come back to any of the known states. Do the search to depth 3 and 5. Deeper searches proved to be inefficient.

(7) Try to apply general chess rules that are described in Section 3.4.

(8) Do the move by hand. There is no clear rule that addresses this situation, and for some limited number of positions, human insight is used to decide upon the move. This usually happens when both sides are attacking, and White needs to stay ahead of the pace by one move. (These moves are then entered into the position database, so the final list will be complete.) Here, we combine automatic generation of moves for the vast majority of positions, while still relying on human skills when needed to get a

complete strategy for White.

- During the full game simulation, if Black's move takes the game into a position closer to the root, again, that branch can be pruned, as it is already explored. This is strickly an optimization, so that we do not have to make all the unnecessary checks for White.

2.5 Proof Outline

Before we jump into chess observations and implementation details let us outline the proof; so the following sections make sense. First, we need to stress the fact that the search for strategy for each starting position will be done in two steps: every leaf of the game tree is in a "Barrier" state, and then we continue expanding the game tree until every leaf is in the database for White.

At the end of first pass we have a database that is full of positions from which we have an automated way of playing described in Section 4. At the end of the second pass we have a full database which can be verified simply, without knowing how the positions were generated.

CHAPTER 3

FOUR FILE **OBSERVATIONS AND DEFINITIONS**

Here, we describe general strategies and observations used in generating our database. Observations are listed informally and without proof - the final proof is the complete database of positions.

3.1 General Game Observations

In the following observations, we assume that Black has not promoted a pawn; hence Black's Bishop and Knight are the original pieces. (As a side claim, we have also proved that White, in forcing a draw, can avoid promotion for Black.) Let us begin by giving an example in Figure 3.1 where having a three-pawn advantage is not

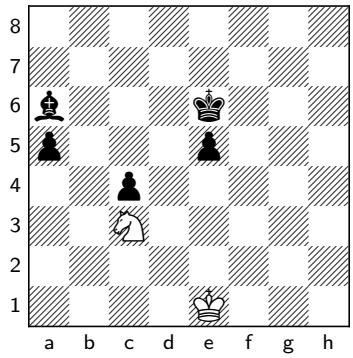

Figure 3.1: Three pawn advantage neutralized

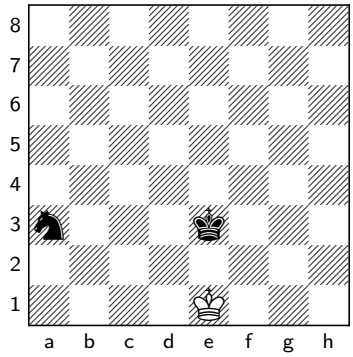

Figure 3.2: Knight Advantage neutralized (Impossible to get to **after** White's move!)

enough to promote a single pawn. White Knight controls both 'a' and 'c' pawns with the simple 'c3'→'a4'→'c3' repetition.

Here are the other basic observations which lay at the core of our definitions of the Barrier states:

- Cannot checkmate with King and Bishop (King and Bishop vs. King only); observe the fact that King's initial position is safe from opposing Bishop, and it is impossible for Kings to get by each other. Note that Bishop has access to only quarter of the board's squares unlike half in a standard chess game.

- Cannot checkmate with King and Knight (King and Knight vs. King only); again the key is that while its possible to look at the position White King on 'e1', Black King on 'e3', and Black Knight on 'c2'. However, the final move had be for the Knight from 'a3', 'a1' to 'c2'; hence the move before that had to be White King 'e2' to 'e1'. This is impossible!!! Situation is shown in Figure 3.2. Again note that Knight has access to half of the board's squares unlike standard chess where it can reach any square on the board.

- The opponent's pawn on 'e-file' can never be promoted, and it blocks the

opponent's King from playing an aggressive role in the end game; hence, capturing opponent's 'e-pawn' is not a good idea, unless checkmate is to follow.

- There are squares on each file, where the opposing pieces (excluding Rook, or King on file 'e') can not capture; pawns cannot ever capture in FOUR FILE. Example: Black's pieces can't reach squares 'a4' and 'a8', 'e4' and 'e8', and any black square on files 'c' and 'g'.

- The only way to win is to have your Rook in front of your pawns; otherwise the game is a draw due to lack of mobility. We explain the idea of Barrier States more fully in Section 4.

- The 'e-pawn' needs to move for the Knight to get involved in the game. If Black never moves its 'e-pawn', the Knight is automatically neutralized and it makes the goal for White much simpler.

- The board is divided into two halves, just like in a chess game. **Long half** 'a', 'c', and 'e' files; and **short half** 'e' and 'g' files. Note that 'e' file is included in both, since the 'e-pawn' is neutralized by the King and cannot be promoted. A single Knight or Bishop can neutralize all pawns on either half; hence a 3-pawn advantage may prove to be insufficient to win in some cases.

- The strength of pieces is: Rook, Knight, Bishop, pawn. The Rook is by far the most powerful piece; neutralizing it early makes it much easier for White to force a draw or better.

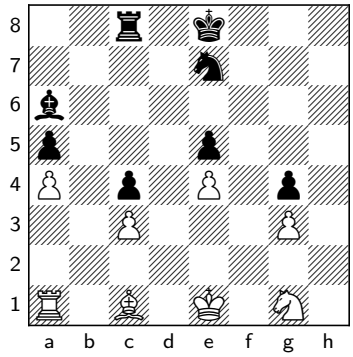

Figure 3.3: Locked Pawns (all pawns are locked)

3.2 Definitions

In discussing positions, we use the term *locked pawns*. Locked pawns cannot move, and prevent Black pieces from offensive attack. There is only one square each where we lock 'a-' and 'e-' file pawns, while two different squares each allow us to lock the 'c-' and 'g-' file pawns (This is with regards to our strategy. There are actually other ways to lock Pawns, however this does not happen here due to our strategy.) Most of the time, we lock pawns with other pawns; but there are cases where this is done with either Bishop, or Rook, as you can see in Section 4.8.

In Figure 3.3 all pawns are locked; however, in Figure 3.4 only 'c-' and 'e-' pawns are locked. In our strategy, black 'a-pawn' is locked on 'a5', black 'e-pawn' is locked on 'e5', black 'g-pawn' is locked on either 'g4', or 'g6', and black 'c-pawn' is locked on either 'c4', or 'c6'.

Another important definition is of *safe squares*. Safe squares are squares where White pieces are untouchable by Black Knight and Bishop. It's important to set all white pawns in *safe squares*; so that we can force repetition using only *heavy pieces* (Rook, Knight, and Bishop) and Kings. In Figure 3.4 all White Pawns are located on safe squares - namely: 'a4', 'c3', 'e4', and 'g3'.

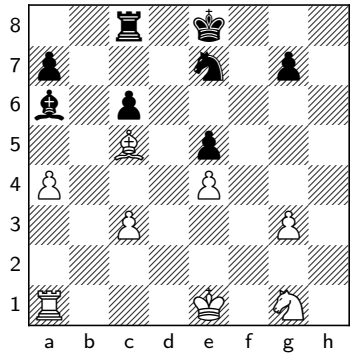

Figure 3.4: Locked Pawns ('c' and 'e' pawns are locked)

3.3 Draw Observations

Our proof involves a large number of positions. We wanted to prune the game tree whenever possible, to make it tractable, and to decrease the size of our "proof". The following situations are the ones where we initially assumed that the game could stop, before going back to complete the database:

- Black Rook, Bishop, and Knight are captured. None of black pawns may promote - the corresponding white pawn is still in play. An example is shown in Figure 3.5.

- Black Rook and Knight are captured. Black Bishop is still in play, however all white and black pawns (except for 'e-pawn') must be locked, so only the White King (*or some other heavy piece* is able to move. If extra White pieces are still in play it means that White may win, but we only care to prove that White can avoid loss. This is shown in Figure 3.6.

- Black Rook, and Bishop are captured. Black Knight is still in play, and Black does not have any extra pawns. Also all white pawns are on *safe squares* and black 'e-'pawn is still present and it is locked. If extra White pieces are still

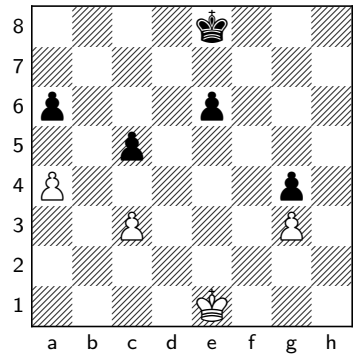

Figure 3.5: Definite Draw (option 1)

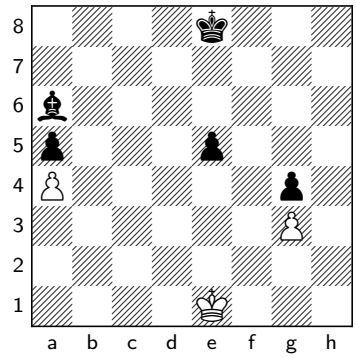

Figure 3.6: Definite Draw (option 2)

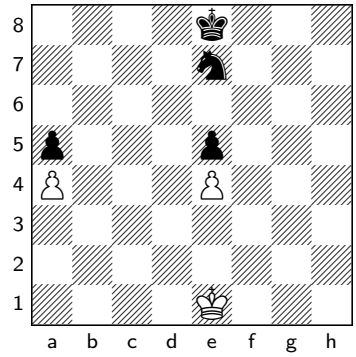

Figure 3.7: Definite Draw (option 3)

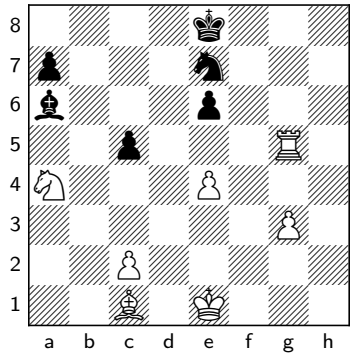

Figure 3.8: Definite Draw (option 4)

in play it means that White may win, or force repetition; but we only care to prove that White can avoid loss. This is shown in Figure 3.7.

- White has an extra Rook, and we got into this state by getting out of one of the Barrier states. Basically, Black sacrificed a Rook to take White out of the Barrier state. This is shown in Figure 3.8. Of key importance here is that each Barrier state is defined such that either Black cannot get out of the Barrier state, or if it can, it does so by sacrificing a Rook, *but does not gain any positional advantage for offense in making such a sacrifice.* (The database will be populated with moves for these positions after our main program run is complete.)

- Rooks are exchanged. Black has an extra pawn, while White has an extra piece. Example is shown in Figure 3.9. This conclusion follows from the fact that no square reachable by White Knight may be reached by Black Knight, and the same holds for Bishops. Hence, one piece may stay still blocking opponent's extra pawn; while the other piece is forcing repetition.

- White has an infinite-check capability. White may not have a win, but for

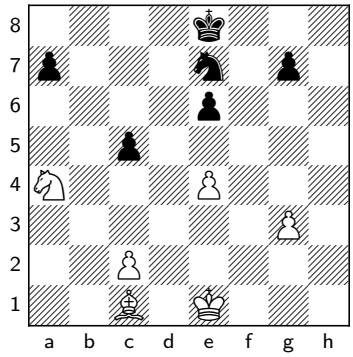

Figure 3.9: Definite Draw (option 5)

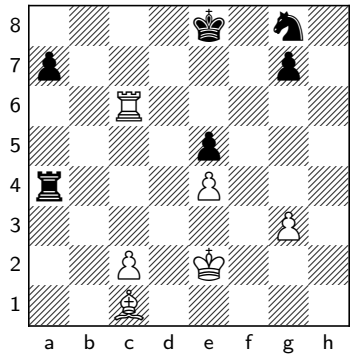

Figure 3.10: Definite Draw (option 6)

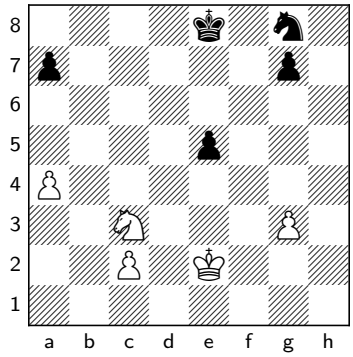

Figure 3.11: Definite Draw (option 7)

every move that Black makes White has some safe move with its Rook to check Black King. Example is shown in Figure 3.10. This usually requires Black Bishop to be captured, and Black Knight to be on opposite half of the board.

- Rooks and Bishops are exchanged; both Knights are still in play. Example is shown in Figure 3.11. Knights, just like Bishops can't capture their counterpart, therefore the repetition is easily achieved. White King has to occupy 'e2' square, so it can't be checkmated by the Black Knight – 'e2' is a safe square in regards to the opposing Knight.

- White has two piece advantage, while Rooks are still in play. Example is shown in Figure 3.12. Only positions where Black may not capture any of White's pieces with its next move, and White may get to a defensive position where White Bishop is on 'c1' and White Knight is on 'e2' are used - **not all positions with two-piece advantage**.

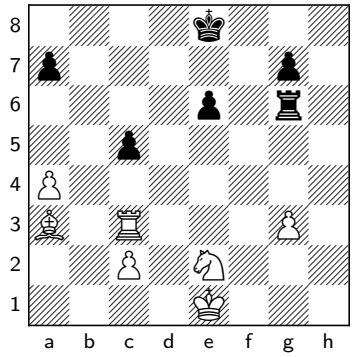

Figure 3.12: Definite Draw (option 8)

3.4 General Move Rules

We will discuss Barriers (end-game definitions) described in Section 4. In this section we want to mention general rules that our program takes to get to the end-game. Of course, there are exceptions to these rules; but those are simply done by hand. Below are listed some of the simple rules that are taken during middle-game.

- Move the King away from possible check. If White King is on 'e2' and Black Bishop is still at large, then move it to 'e1'. This usually resulted from Black sacrificing its Rook.
- Capture Black Rook. Most likely moves us to a Barrier state right away.
- Capture Black Bishop on 'e2'. Could result from a Black Bishop — White Knight exchange, or simply Black Bishop sacrifice. Therefore the capture could be made with either Knight or King.
- Lock 'e-pawn'. Described in Figure 3.13. This satisfies an important condition of many Barriers. The move is 'e2e4'.
- Lock 'g-pawn'. Usually with the move 'g2g3'.

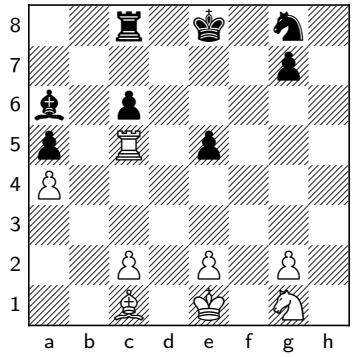

Figure 3.13: Locked e-pawn Rule

- Move White Knight to defensive position on ‘c3’. White needs to have ‘a-pawn’ on ‘a4’ and ‘e-pawn’ on ‘e4’. White Knight will take a move ‘e2c3’ in case described in Figure 3.14; but in some cases the move ‘g1e2’ needs to be made first.

- Capture ‘g-pawn’ with White Rook. White Knight has to be able to assume a defensive position from the previous rule for this rule to apply; but the goal is not to allow black ‘g-pawn’ to get to ‘g4’, unless black ‘c-pawn’ is already on ‘c4’. Both pawns” ‘c4’ and ‘g4’ will block both Black Bishop and Black Knight from any offensive play. The position we are trying to avoid in shown in Figure 3.16. On the other hand Figure 3.15 shows the time when the capture is made.

- Black Knight capture. There are multiple states when this can take place. An important state is when Black Knight gets to ‘e3’ or ‘a3’, and White Bishop is on ‘c1’; the capture **must** take place.

- Move White Bishop to ‘c1’ if it is on ‘a3’ and Black Knight gets to ‘c4’.

- Black Bishop capture with White Rook. There are multiple states when this

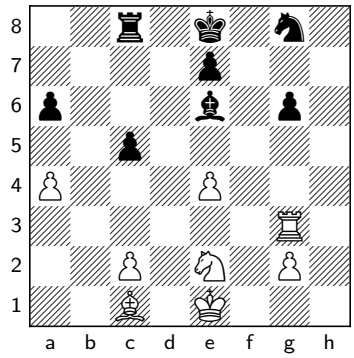

Figure 3.14: White Knight defensive

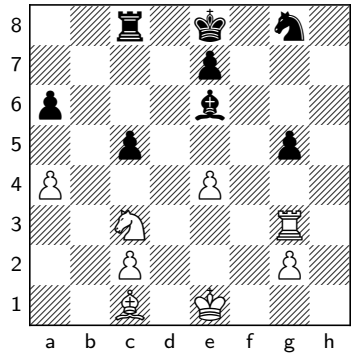

Figure 3.15: G-pawn capture

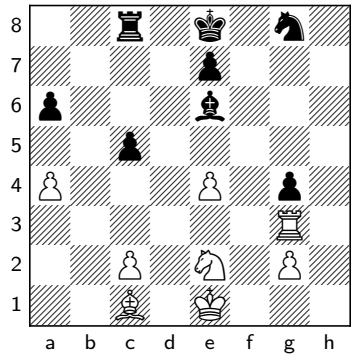

Figure 3.16: Bad state

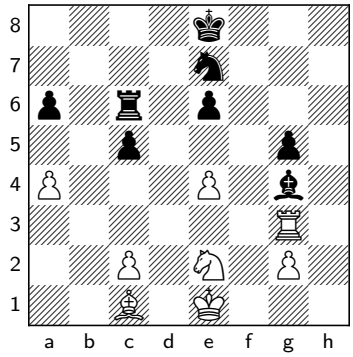

Figure 3.17: Black Bishop capture

can happen, but an important one is when black 'g-pawn' was not captured yet, and Back Bishop does not allow this to happen by occupying 'g4' square as shown in Figure 3.17.

- Lock 'c-pawn'. This happens if Black Bishop was exchanged for White Knight, and black 'c-pawn' got to 'c4'. This is done by 'c2c3' move.

CHAPTER 4

BARRIERS

Here, we outline the barrier states. In these states, Black's pawns prevent Black from mounting any effective attack. For many barrier states, there is no way out of the barrier state for Black. For others, Black's only exit is by sacrificing its Rook, without enough positional gain to mount an attack on White's King.

4.1 Barrier 1

In this Barrier the pawns are locked on 'a-file' and 'e-file'. As the result, the Black Knight is caught behind its own pawns. White Rook controls the 'c-file', while also protecting the pawn on 'g3'. The only way for Black to get out of this barrier

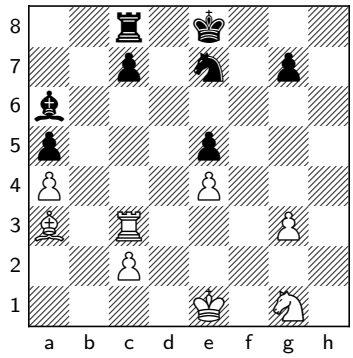

Figure 4.1: Barrier 1

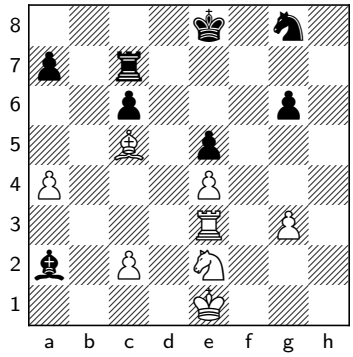

Figure 4.2: Barrier 2

is to sacrifice the Rook on either ‘g3’, ‘e4’, or somewhere on ‘c-file’. Described in Figure 4.1.

- If Black Rook capture is available - do it!
- In case Black Rook gets to ‘g4’, then White Rook goes to ‘e3’
- In all other cases, White Rook is on ‘c3’
- White has repetition by moving the Knight from ‘g1’ to ‘e2’ and back. If Black Bishop captures White Knight on ‘e2’, then White King captures Black Bishop and forces repetition with the King going from ‘e1’ to ‘e2’ and back.

4.2 Barrier 2

In this Barrier the black pawns are locked on ‘c-file’ and ‘e-file’. As the result, the Black Knight is caught behind its own pawns. The black ‘c-pawn’ is locked by White Bishop. White Rook protects the pawn on ‘g3’; White Rook can be either on ‘e3’ or ‘c3’, though it ends up on ‘e3’ once forced Black Rook move to ‘g4’. The only way for Black to get out of this Barrier is to sacrifice the Rook on ‘g3’ or ‘e4’. Described in Figure 4.2.

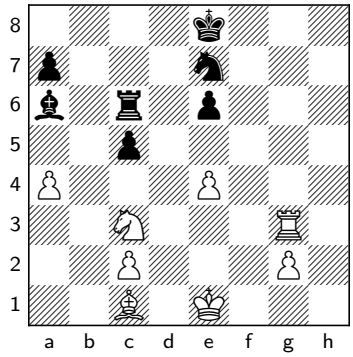

Figure 4.3: Barrier 3

- Black Rook capture on ‘g3’ or ‘e4’ is available - do it!
- In case Black Rook gets to ‘g4’, then White Rook goes to ‘e3’.
- White has repetition by moving the Knight from ‘g1’ to ‘e2’ and back. As before, if Black captures the Knight with its Bishop, then White King captures Black Bishop and forces repetition by going from ‘e1’ to ‘e2’ and back.

4.3 Barrier 3

In this Barrier the White Knight protects pawns on ‘a4’ and ‘e4’. Black pawn could be on either ‘c6’, ‘c5’, or ‘c4’ and it blocks the Black Rook from attacking the White Knight. White Rook controls the ‘g-file’. Black ‘g-pawn’ is captured, so White Rook has full maneuverability. White Bishop is on ‘c1’, so it protects ‘a3’, ‘e3’ and ‘g5’ squares. Sometimes White Bishop will end up on ‘a3’ - read below. **Reader should make sure to understand exactly why White Knight is safe on ‘c3’**. Black can get out of this Barrier by sacrificing a Rook on ‘a4’, ‘e4’ or ‘g-file’. Also Black can sacrifice the Black Knight on ‘e5’, which will lead to Black loss. Described in Figure 4.3.

- Capture Black Rook if available. Do this even if it is just an exchange!
- If Black Knight gets to ‘a3’ or ‘e3’ - capture it with the White Bishop.
- If Black Bishop or Knight is unprotected on ‘g4’ - capture it.
- If Black Bishop or Knight is unprotected on ‘g8’ - capture it.
- If Black Knight is unprotected on ‘g6’ - capture it.
- If Black Knight is on ‘c6’, White Rook is on ‘g5’ and White Bishop is on ‘c1’; then move White Bishop to ‘a3’. This is done because we need to keep the White Rook on ‘g5’ to protect the ‘e5’ square. By the same reasoning, if White Bishop is on ‘a3’; then move it to ‘c1’. If White Rook is not on ‘g5’, then move it to ‘g5’ (need to protect ‘e5’ square).
- If Black Knight gets to ‘c4’, and White Bishop is on ‘a3’; then move White Bishop to ‘c1’. *This usually means that Black already sacrificed a Bishop.*
- White has repetition by moving the Rook from ‘g3’ to either ‘g5’ or ‘g7’; since both squares can not be blocked. For these squares to be blocked either Black Knight or Black Bishop had to be sacrificed; or Black was forced into checkmate, or repetitive check. The reason is White Bishop on ‘c1’ that protects the square, so that Black has a Rook exchage at best! *As an exercise the reader can verify that it is impossible to end up with Black Knight on ‘g4’ with Black Bishop on either ‘e6’ or ‘c8’ protecting it; without Black sacrificing a Rook or worse!*

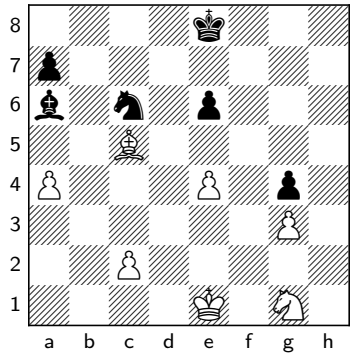

Figure 4.4: Barrier 4

4.4 Barrier 4

In this Barrier the Rooks are exchanged, Black 'e-pawn' is present, and Black has no pawns that can promote. White Bishop and Knight are still present. We assume that Black Bishop and Knight are still present too; otherwise we are in a basic draw-state. Described in Figure 4.4.

- Capture Black Knight if available.
- Make a move towards getting White Bishop on 'c5' square; from there it can control Black Knight. Take safe steps to get there. *Safe path* is from 'c1' to go to 'a3' if the opposing Knight is not on 'c4'; and then 'c5'. There are other *safe paths* from other starting locations, but the idea should be clear.
- White has repetition by moving the Knight from 'g1' to 'e2' and back. If Black captures the Knight with its Bishop, then White King captures Black Bishop and forces repetition by going from 'e1' to 'e2' and back.

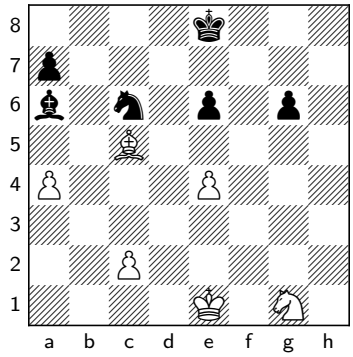

Figure 4.5: Barrier 5(a)

4.5 Barrier 5

In this Barrier the Rooks are exchanged, black 'e-pawn' is present, and Black has an extra pawn on 'g-file' that needs to be controlled. It can only start on 'g6' or 'g7'; as such Rook exchange would take place on 'g5'. Described in Figure 4.5.

- Make sure that all white pawns are on *safe squares*.
- Make a move towards getting White Bishop on 'c5' square; from there it can control Black Knight. Take safe steps to get there.
- Take White Knight into 'e2'→'g3'→'e2' repetition state. We are always one move away from this: either 'g1e2', or 'c3e2'.
- If White can capture a black pawn on 'g3' - do it! Black moved their pawn to far.
- If Black Bishop captures the White Knight on 'e2', then capture the Black Bishop with the King.
- If Black Knight is on 'a3', or 'e3' capture it with the White Bishop. It is safe

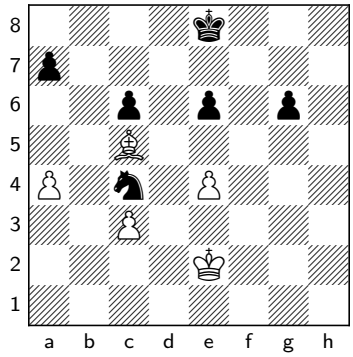

Figure 4.6: Barrier 5(b)

to do, since the furthest the black pawn can be at this point is 'g4'. White Bishop has enough time to get back to control its progress.

- After the exchange of Black Bishop and White Knight takes place, White King is safe at 'e2' and White forces repetition by moving its Bishop from 'c5' to 'g1' and back. Described in Figure 4.6.

4.6 Barrier 6

Another Barrier where Rooks are exchanged. This Barrier is different because Black 'e-pawn' is missing. This is key because if White Knight can not find a safe place to force repetion, then White may lose. In this Barrier, our strategy forces safe repetition on 'c3'→'a4'→'c3' squares. Figure 4.7 shows the Barrier.

4.7 Barrier 7

This Barrier is actually a *setup* barrier. From this barrier it is very easy to transition into many of the Barriers described earlier. The idea is simple, don't let Black Rook get out, and force some moves from Black that will eventually force the draw.

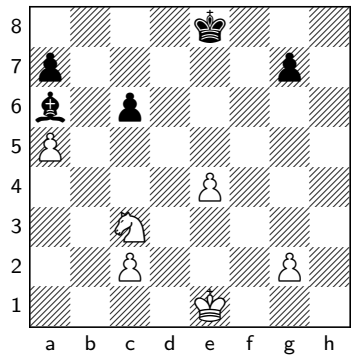

Figure 4.7: Barrier 6

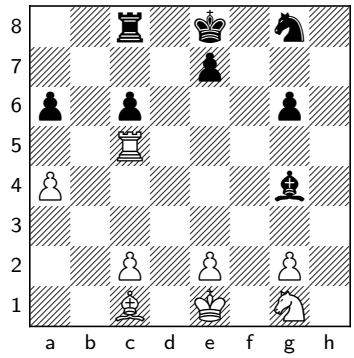

Figure 4.8: Barrier 7 (White Rook on c5)

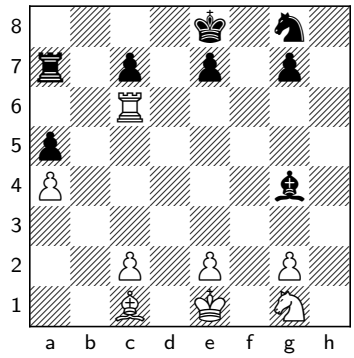

Figure 4.9: Barrier 7 (White Rook on c6)

- Moving ‘e-pawn’ before Rook can get out, hence blocking the Knight
- Moving ‘g-pawn’, hence blocking the *short half* of the board for Black
- Limited mobility results in loss of pieces

If all black pawns are in place, then there is only two ways that Black Rook can get out. The Barriers in Figure 4.8 and Figure 4.9 describe both situations.

- If White Rook is on ‘c5’ and Black moves the ‘a-pawn’ to ‘a5’, then we need to consider moving White Rook to ‘c6’, unless black ‘e-pawn’ has moved
- If White Rook is on ‘c6’ and black is moves its Knight to ‘e7’, then White has to move the Rook to ‘c5’
- Move White Bishop from ‘c1’ to ‘a3’
- Move white ‘e-pawn’ from ‘e2’ to ‘e4’
- Move white ‘g-pawn’ from ‘g2’ to ‘g3’
- Once Black has moved both: ‘e-pawn’ and ‘g-pawn’; White has repetition by moving the Knight from ‘g1’ to ‘e2’ and back. If Black Bishop captures White Knight on ‘e2’, then White King captures Black Bishop and forces repetition with the King going from ‘e1’ to ‘e2’ and back.

4.8 Other Barriers

There is also a big family of Barriers that results from Black’s passive play. Here are a couple of examples: Figure 4.10 shows the Barrier where ‘c’ and ‘g’ pawns are locked, and ‘a’ and ‘e’ pawns will get locked once Black decides to move them. Figure 4.11 shows the Barrier where ‘c’ and ‘e’ pawns are locked by White Bishop and White Rook, and Black Rook can not get ahead of the pawns blocking it.

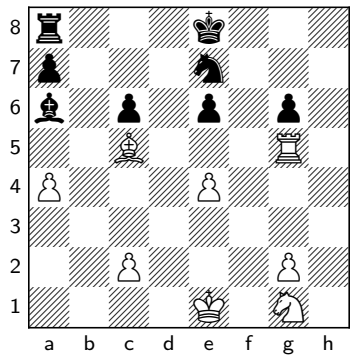

Figure 4.10: Random Barrier (a)

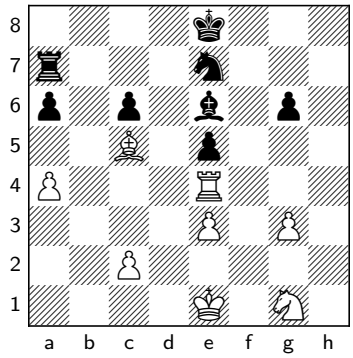

Figure 4.11: Random Barrier (b)

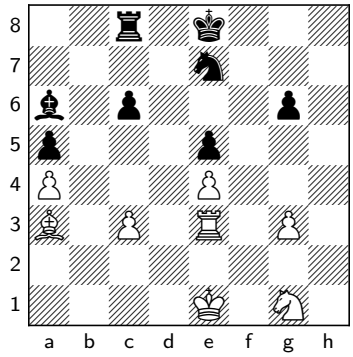

Figure 4.12: No Rook Capture in Barrier 1

- If Black Rook capture is available - do it!
- White has repetition by moving the Knight from ‘g1’ to ‘e2’ and back. If Black captures the Knight with its Bishop, then White King captures Black Bishop and forces repetition by going from ‘e1’ to ‘e2’.

4.9 No Rook-Capture Barriers

Let us mention the barrier states that will not require Black’s Rook capture even if such option is available. In these states, Black’s pawns prevent Black Rook’s mobility. They are used mainly to keep the size of the database down.

- Subset of Barrier 1; in which ‘c’, and ‘g’ pawns for Black are advanced past their starting positions. Black Rook is hidden behind black pawns. White ‘c-pawn’ also is advanced, so that it can stop black ‘c-pawn’ progress. Described in Figure 4.12.
- Subset of Barrier 2; in which ‘g-pawn’ for Black is advanced past its starting position. Black Rook is hidden behind black pawns. Described in Figure 4.13.

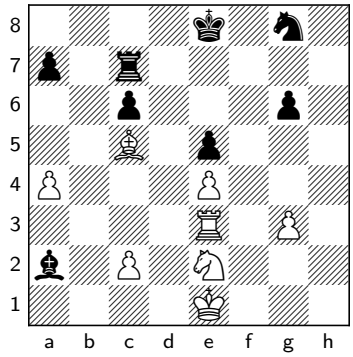

Figure 4.13: No Rook Capture in Barrier 2

4.10 Barriers Needed when Black Moves First

In this Barrier the pawns are locked on 'g-file'. White Knight protects pawns on 'a-file' and 'e-file'. Repetition is accomplished by White Rook, which may be sacrificed - exchanged for either Black Knight or Black Bishop. The barrier is shown in Figure 4.14. After the White Rook is sacrificed, then repetition is accomplished by either White King as shown in Figure 4.15, or by White Bishop as shown in Figure 4.16.

- If Black Rook capture is available - do it!
- If Black Bishop capture is available on 'e2' - do it!
- If Black Knight capture is available on either 'e3' or 'a3' - do it!
- Do general repetition with White Rook: 'e3' ➟ 'e2' ➟ 'e3'
- White Rook was exchanged for Black Knight, which means that Black Knight was captured on 'e3' square. If White Bishop is on 'e3', then bring it back to 'c1' for standard bishop repetition: 'c1' ➟ 'a3' ➟ 'c1'.

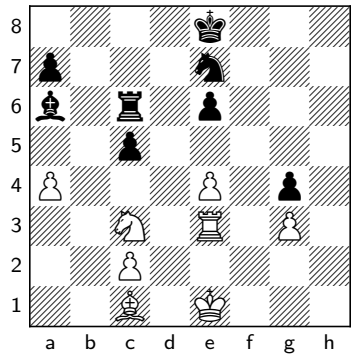

Figure 4.14: White Goes Last (Barrier 1)

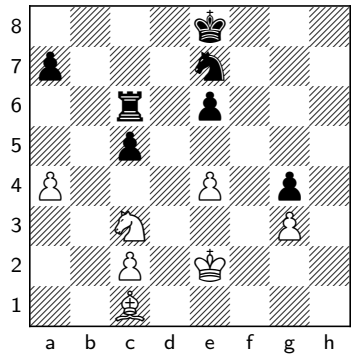

Figure 4.15: White Goes Last (Barrier 2)

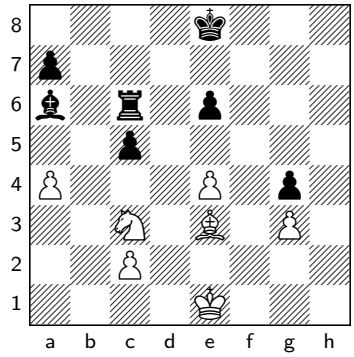

Figure 4.16: White Goes Last (Barrier 3)

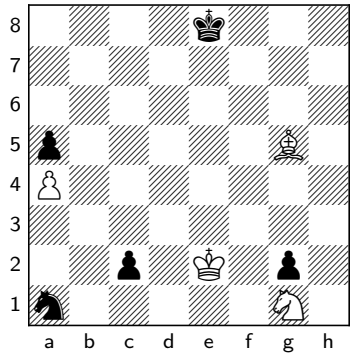

Figure 4.17: Avoiding Promotion (a)

- White Rook was exchanged for Black Bishop. Do a repetition with White King: 'e2' ➟ 'e1' ➟ 'e2'

4.11 Avoiding Promotion

There are some Barriers that result from very aggressive play by Black, where Black sacrifices a Bishop for two or three pawns. Here are two examples: Figure 4.17 shows the Barrier where in the endgame we have 'c', and 'g' pawns for Black in return for one White Bishop; Figure 4.18 shows the Barrier where in the endgame we have 'a', 'c', 'e', and 'g' pawns for Black in return for one White Bishop.

In Figure 4.17 the following rules are followed.

- Capture Black Knight if such a move is available. If Black Knight ends up on either 'e1' or 'e3', capture the Black Knight on 'e3' with the Bishop.
- White has repetition by moving the Bishop from 'g5' to 'c1' and back.

In Figure 4.18 the following rules are followed.

- Capture Black Knight if such a move is available. If Black Knight ends up on either 'e1' or 'e3', capture the Black Knight on 'e3' with the Bishop.

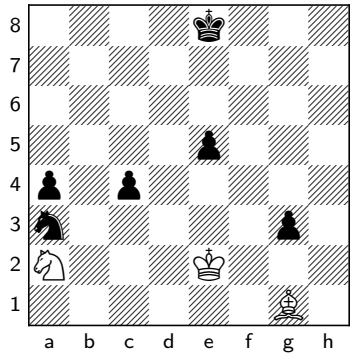

Figure 4.18: Avoiding Promotion (b)

- White has repetition by moving the Knight from ‘a2’ to ‘c1’ and back. There are other similar cases where the repetition is achieved by moving the Knight from ‘a4’ to ‘c3’ and back.

CHAPTER 5

IMPLEMENTATION OVERVIEW

5.1 Game Simulation

Currently, we have a text version of the game; as well as a GUI version that never loses for White. The game is played by simply doing a lookup in the database of known states. The database of known states was populated by doing a full game search given a known White Strategy.

5.2 Custom Moves

One of the simplest ways to optimize the game tree pruning was to allow the position enumerator ask the user for a move given a position, if all of hard-coded rules failed to produce the result. Another reason for doing this was that during the early stage of the game there were many exceptions, where Black can potentially do moves that (should) result in a loss for Black. At that point if White takes advantage, many extra positions will be added to the database, and the game tree will be much bigger. Taking an early win is not always good for game tree pruning, since the "less intelligent" moves for White may take the board to positions which need to be explored anyway. Using custom moves early allowed for more game control by replacing board evaluation methods with user evaluation, and allowed us to better

force the game into the Barrier positions. Here are the first three moves of my strategy:

(1) "a2a4" - opening for the Rook to get out

(2)
- "e2e4" - if Black 'e-pawn' is on 'e5'
- "a1a3" - in all other cases; let the Rook get out

(3)
- "e2e4" - if Black 'e-pawn' is on 'e5'
- "a3e3" - if Black 'e-pawn' is on 'e4'
- "g2g3" - if Black 'g-pawn' is on 'g4'
- "a3g3" - if there is a chance to get Black 'c-pawn' on 'c4' with Black's next move, and the 'c4' square will be protected by the following move by either Black Rook, or Black Bishop - there are four cases like that; the reader should verify for full understanding. *First case is when black 'c-pawn' is already on 'c4'; it gets protected by Black Bishop with the following move.*
- "a3c3" in all other cases

5.3 Database of Positions

We use a Hashtable to store ⟨position, move⟩ tuples for White. The number of positions can fit in memory, and there are frequent reads and writes; hence writing this information to disk is highly inefficient. Also, there is no need for permanent storage since all of positions are re-generated on every run. The key is to make storage of the board space efficient - we used 22 bytes for single board description - it could be further improved, but this wasn't needed.

CHAPTER 6

IMPLEMENTATION DETAILS

6.1 Storage Overhead

In this section let us describe the implementation details in our program that allows us to efficiently check if there is a move for a certain position, or if certain position has been visited already.

- **Hashtable** for all positions that we had seen after White has made a move; if when it is White's turn there exists a move that will take us to any of the positions in this Hashtable - we are done. This way we don't have to keep track of depth levels - if position is in the Hashtable it has been visited.
- **Hashtable** for all positions that we had seen after Black has made a move; if when it is Black's turn a move is chosen that will take us to any of the positions in this Hashtable - we are done.
- **Dictionary** of ⟨position, move⟩ tuples. It is populated at startup with all of the existing positions, and whenever our program comes up with a move for a new position the Dictionary is updated.

6.2 Chess Board

Chess Board class contains an instance of every chess piece. All chess pieces implement the same interface, so that it is easy to do the following.

- Store the state of the board, and allow the user to modify it. This includes marking pieces as captured, their locations, which side has the next move, and history of moves taken to get to this position (for debugging purposes).
- Has a method to get all possible moves for a given chess-piece, or a side. We need this functionality for a complete mini-max search of the tree.
- Checks if current position is a winning / losing position for a given side. This action can be performed for some fixed depth. The whole tree is enumerated, no pruning techniques are used here. The board evaluation is to simply check if the King for a given side is captured.
- Checks the validity of the move proposed

6.3 Defensive Rules

While we have not specified all of the rules that are followed to get the Barrier states, there are a few key evaluations that should be made every time before the move is actually chosen. These are the *defensive rules* that allow us to make sure that nothing dangerous is about to happen.

- Is White King checked?
- Can White Rook be captured?
- Can White Bishop or Knight be captured?

The evaluation is implemented in the straight-forward manner. Simply try all of Black's moves and see if any of them results in the board that has one of White's heavy pieces missing. These rules are envoked during Barrier play, as well as during checks for being in the draw state.

6.4 Game Enumeration

Because the number of positions are small enough to fit into memory, a general Breadth First Search approach was used. The implementation is simple - use two queues: one for moves to be explored by White's strategy, and the other to be fully enumerated by Black. When it is White's turn to move - the program tries to apply a strategy to come up with one new position to enumerate for Black. On the other hand when it is Black's turn - every move results in the new position being added to the queue for White (unless that position has already been explored).

6.5 Position Storage

The number of positions is relatively small; hence there is no need to write data to persistent storage during execution, a simple in-memory data structure will do. This data structure is to be populated at the startup of the program and to be updated after each new ⟨position, move⟩ tuple gets generated.

As an example here is board from Figure 4.3 represented as a string: "1g3c3c142428c6e7a6756_" [**This is ok because our strategy does not allow for pawn promotion.**]

6.6 Barrier State Play

Most of the Barrier play is simply hard-coded. Since every position is represented as a string, we use simple parsing techniques to see if this string matches rules for one of the Barrier's defined. If yes, then the rule is applied, hence resulting in a new entry in our database.

6.7 Tree Pruning Optimizations

To further prune the game tree, while not adding too many extra positions to our final list; we are using three basic techniques:

- Check for win to a certain depth. The deeper the search, the longer it takes. Depth seven proved to be a good balance between strength and speed. We avoided board evaluation, checking only if a King is captured for victory (rather than stopping at checkmate one level earlier).
- Capture opponent's Rook. Capturing Black Rook almost always takes us into one of the defined Barrier states.
- Check if we can force coming back to a known position within some small number of moves.

Figure 6.1 displays a position that results in White's victory within (at most) 7 moves.

(1) g5e5 - c4e6

(2) e5e6 - g8e7

(3) e6e7 - e8e7

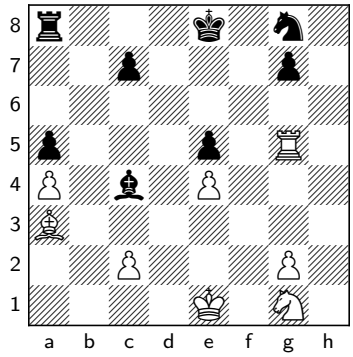

Figure 6.1: White forces win in 7 moves

(4) a3e7*

CHAPTER 7

RESULTS

7.1 Statistics

We came to the final database in two stages:

- Run the simulation with the Draw Definitions from Section 3.3.
- Using all resulting positions from step 1 - complete the database by using the rules that play the game to completion starting from any of the positions in Section 3.3.

In Table 7.1 we show how our table was growing as we explored more and more of the game of FOUR FILE.

7.2 Verification

We can represent strategy as a Graph, where every node is a position; which is either a win for White, a stalemate, or for every move that Black may take, there is a move for White that will take us to another node in the Graph. The current database has been verified for White to force a draw, and Black to force a draw for all but one starting move for white [6].

Table 7.1: Database growth description

Staring position description	Positions added to database
Standard - White moves first	2682885
Black moves first - “g7g5”	1157650
Black moves first - “g7g6”	147103
Black moves first - “a7a6”	216225
Black moves first - “c7c6”	138279
Black moves first - “c8g4”	1818041
Black moves first - “e7e6”	182372
Black moves first - “c8a6”	73114
Black moves first - “c8e6”	60095
Black moves first - “e7e5”	1067253
Black moves first - “c7c5”	148955
Black moves first - “a7a5”	**INCOMPLETE**
Total size of database (excluding duplicates)	7493909

CHAPTER 8

FUTURE WORK

8.1 Proof Completion for Black

We are very close to completing the proof for Black, but it may require one strong assumption to be broken. For Black to force a Draw - it may be necessary to allow for pawn promotion. Our research shows that there are two cases where we may have to allow for pawn promotion to complete our database

- Black promotes while Rooks are exchanged and thus promotes to a Rook *(if default promotion to Queen is not allowed - making Queen an invalid piece in this game)* or Queen and can force at least Draw even if White has extra Bishop or Knight.

- White promotes the next move after Black promotes. Because Rooks are exchanged White does not have enough material to win and Black forces a Draw.

Our board definition does not allow for Queen or multiple pieces of the same kind on the same board; so to complete the proof in this fashion one has to change the way the board is defined.

8.2 Open Problems

The current approach does not try to calculate perfect play for either side. We can consider that the problem can be solved to four different levels. First, it can be non-constructively proven that White and Black can avoid losing. Next, the proof can be constructive, as it is here, giving one such strategy to avoid loss for each side. A stronger strategy would not only avoid losing, but would also give a strategy for White which would win whenever possible, that is, if Black ever made a move which put it into a position in which White could force a win with optimal play, White would do so. Finally, these strategies could be given for arbitrary starting positions, including positions that our strategy would never enter, but which are possible. These positions would be of interest because they would allow a player to start by playing White, and if they find themselves in a tough situation, they could switch sides with the computer and continue.

BIBLIOGRAPHY

[1] http://www.msri.org/communications/books/
Book42/files/guy.pdf

[2] Jonathan Schaeffer, *Checkers Is Solved*, Science 14 September 2007, Vol. 317. no. 5844, pp. 1518 - 1522

[3] Elwyn R. Berlekamp, John Horton Conway and Richard K. Guy, *Winning Ways for Your Mathematical Plays, Vol. 1, 2, 3 and 4*, Academic Press, 1982.

[4] Richard J. Nowakowski, *Games of No Chance*, Cambridge University Press, 1996.

[5] Richard Korf, *Depth-first iterative-deepening: An optimal admissible tree search*, Artificial Intelligence, Vol. 27, No. 1, pp. 97-109, 1985. Reprinted in Expert Systems, A Software Methodology for Modern Applications, P.G. Raeth (Ed.), IEEE Computer Society Press, Washington, 1990.

[6] David Scot Taylor, Personal Communication, 2009