The Effects of Rule Variations on Perfect Play Databases for Nine Men's Morris

Research · July 2016

DOI: 10.13140/RG.2.1.4972.4407

Wesley Loewer
Rosslyn Academy

The Effects of Rule Variations on Perfect Play Databases for Nine Men's Morris

Wesley B. Loewer
WESL@ROSSLYNACADEMY.COM
Rosslyn Academy, Nairobi, Kenya

Abstract

When the game of Nine Men's Morris was originally shown to be a draw, it was solved using a set of rules which differed slightly from the rules used in official tournaments sanctioned by the Weltmühlespiel Dachverband (World Association of Nine Men's Morris). Since that time, other Morris variants which have significantly different rules, such as Morabaraba and Lasker Morris, have been analyzed and solved. However, the effects of the more subtle rule variations in the standard game have not yet been investigated. This paper examines the impact these rule variations have on the standard game of Nine Men's Morris.

Retrograde analysis was used to create complete perfect play databases for the four combinations of rules commonly used in the standard game. While these databases show that the theoretical game values are draws for all four combinations, they also show that these rule variations can have an unexpected impact on the perfect play of the game.

1. Introduction

Nine Men's Morris (also known as Mill or Merrills) is a zero-sum perfect information board game played between two players. Each player has nine playing pieces, or stones, which are distinguished by color, typically white and black. The board consists of three concentric squares with line segments connecting the midpoints of each side as shown in Figure 1. The 24 intersections and corners make up the locations at which a piece can be played. By convention, White plays first.

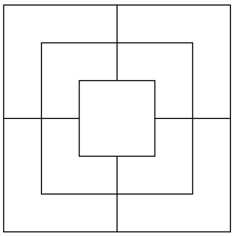

Figure 1: The Nine Men's Morris game board

2. The Rules of the Game

Since this paper deals specifically with the rules of Nine Men's Morris, it is imperative that they are explained in detail. The game has distinct opening, middle, and endgame phases in which different rules are in effect. However, throughout all phases of the game, the following ground rules apply.

1. Whenever a player manages to get three pieces lined up in a row, it is referred to as a "mill" and that player can then capture one of the opponent's pieces.

2. Any of the opposing pieces which are themselves in a mill are protected from being captured.

3. When a player has only two pieces left, or is blocked in and cannot move, then that player has lost.

2.1 The Opening Game: Setting

In the opening phase of the game, each player takes turns placing a piece on the board at one of the 24 points located at an intersection or corner. To identify the positions, a notation similar to chess's algebraic notation is commonly used. For example, suppose White started with **d6** and Black followed with **c5** then White responded with **d2**. The board resulting from these three plies is shown in Figure 2. (The term "ply" refers to the action done by one player, while the term "move" refers to the action done by both players. So if ten plies have been played, then each player has made five moves. A ply can be thought of as a "half-move.")

	White	Black
1.	d6	c5
2.	d2	

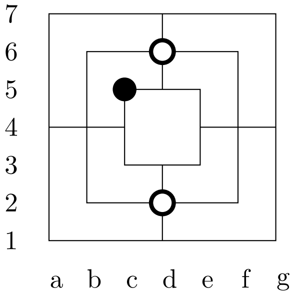

Figure 2: After 3 plies

Play continues with Black playing **c4** and White playing **f4**.

	White	Black
1.	d6	c5
2.	d2	c4
3.	f4	

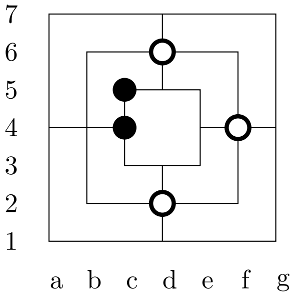

Figure 3: Black to move and form a mill

On Black's next move, she can set a piece on **c3** to form a mill (three in a row) and can therefore capture one of White's pieces, such as **f4**, and permanently remove it from play. (For the purpose of clarity in this paper, White will be referred to as "he" and Black as "she.") The notation for indicating a capture is the × symbol, such as **c3×f4** as shown in Figure 4. This opening phase continues until both players have placed all 9 of their pieces (18 plies).

	White	Black
1.	d6	c5
2.	d2	c4
3.	f4	c3×f4

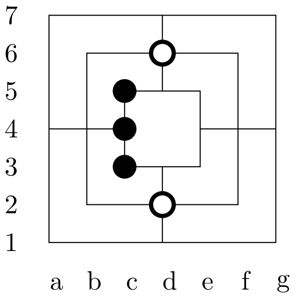

Figure 4: Black just formed a mill and captured White's piece that was on **f4**

2.2 The Middle Game: Moving

Once all 18 pieces have been placed, the game transitions into the moving phase in which players slide their pieces along the lines from one location to an adjacent empty location. The above game in Figure 4 could continue until it reaches the following state in Figure 5.

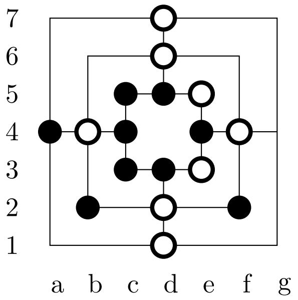

Figure 5: Ply 19, start of the moving phase

At this point, White can move from **d1** to **a1**, indicated by **d1–a1**. Now Black's only legal move is **a4–a7**. It is clear that Black has not played wisely. White blocks her in with **a1–a4** and Black loses since she cannot move, as shown in Figure 6. White wins even though he has not captured any pieces.

	White	Black
19.	d1–a1	a4–a7
20.	a1–a4	

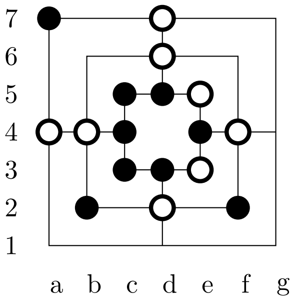

Figure 6: Black cannot move, White wins

2.3 The Endgame: Jumping

If the game continues without either side being blocked in, it is possible for pieces to be captured until the game has reached the state shown in Figure 7. Black has just closed a mill, capturing one of White's pieces, leaving White with only three pieces. Once a player is down to three pieces, he is no longer restricted to moving along the lines to adjacent positions, but can now jump to any location on the board.

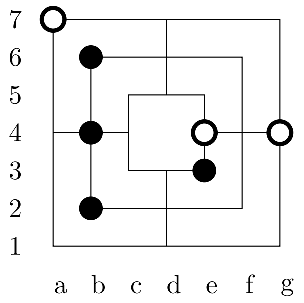

Figure 7: Start of the jumping phase

White can jump from **a7** to **f4** to close his mill. If he does this, he will be forced to capture **e3** since the other pieces are in a mill and are therefore protected. However, now Black also has only three pieces and therefore she can jump as well, eventually winning the game, as seen in Figure 8.

	White	Black
40.	a7–f4×e3	b2–a4
41.	g4–b2	b6–c4×f4

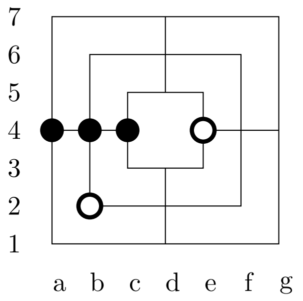

Figure 8: Black wins

3. Rule Variations

As indicated earlier, the rules for capturing dictate that when a player forms a mill, he can capture an opponent's piece; but pieces that are currently in a mill are protected from being captured. These two rules can come into conflict with each other in certain situations that require special attention.

3.1 Always Protected or Always Capture

The middle game position in Figure 9 shows how the two rules can conflict with each other. It is White's turn to move and he could close a mill by moving **d5–e5**. One rule says that White can capture a piece since he closed a mill, but the other rule says that Black's pieces are all protected from capture since they are all in mills.

Some people give preference to the rule that Black's pieces in mills are always protected, so White cannot capture anything. Others give preference to the rule that since White has formed a

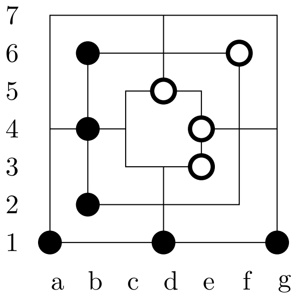

Figure 9: White to move. Can White capture a piece by closing his mill?

mill he has the right to capture something and therefore he can capture one of Black's pieces even though they are all in mills. These two rule variations will be referred to as the "Always Protected" rule and the "Always Capture" rule respectively.

3.2 Take 1 or Take 2

In the opening phase, it is possible for a player to close two mills simultaneously. In Figure 10, Black can accomplish this by playing **f4**.

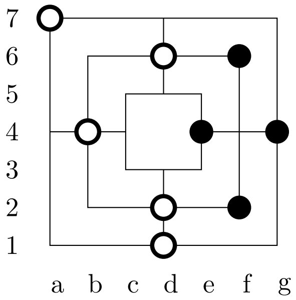

Figure 10: Black can play **f4** to close 2 mills simultaneously

Some play by the rule that if Black plays **f4**, she can capture one of White's pieces as previously described. Others prescribe that since Black has closed two mills simultaneously, she is entitled to capture two pieces. These two rule variations will be referred to as the "Take 1" rule and the "Take 2" rule respectively.

3.3 Which Rule Variations are Used by Whom?

When Ralph Gasser originally solved Nine Men's Morris in his pioneering work in 1993, he used the Always Capture, Take 1 combination of rules (Gasser, 1993). This set a precedent for others whose later research involved Nine Men's Morris (Lincke, 1994; Edelkamp, Sulewski, & Yücel, 2010; Gévay & Danner, 2014). In identifying which rules he used, Gasser states, "It seems unlikely that either of these rule variations affect the value of the game" (Gasser, 1996). It was this statement that provided the impetus for this analysis. Was this assumption, in fact, correct? It raised the question, "Do the rule variations actually affect the initial theoretical game value, and if not, to what degree do they affect the game?"

In contrast, at the time of Gasser's work the Weltmühlespiel Dachverband (WMD) rules which governed official tournaments used the Always Protected, Take 2 rules (Weltmühlespiel Dachverband, 2008). The game using these official WMD rules was first solved and shown to be a draw by Peter Stahlhacke in 2000 (Stahlhacke, 2016) prior to his solving of Lasker Morris (Stahlhacke, 2003).

In order to reduce the number of draws in tournament play (Lehner, 2016), the WMD changed their official rules in 2009 to the Always Capture, Take 2 rules (Weltmühlespiel Dachverband, 2010). This game using the new WMD rules was first solved and proven to be a draw in 2014 when the complete perfect play database was generated by the "Brillant Mill Team" of programmers and players consisting of Alexander Szabari, Jozef Mičko, Ferenc Volman, and György Bándy (Brillant Mill Team, 2016). As part of the research for this paper, the perfect play databases for all four combinations were generated by the author in 2014-15.

To illustrate how these variations are all commonly used, the table below shows which rules have been used by various researchers, software developers, and online game sites.

	Always Protected	**Always Capture**
Take 1	Muehle 2.4 (Richard Fischer) 3D Morris (Lobstersoft)	Gasser Lincke Edelkamp, Sulewski & Yücel Malom (Gévay & Danner) Nine Men's Morris (Carl von Blixen) FlyOrDie Smart Little Games Kongregate
Take 2	Mühle Datenbank (Stahlhacke) WMD (prior to 2009)	Brillant Mill (Brillant Mill Team) PlayOK/Kurnik WMD (current rules)

Table 1: Distribution of rules used by various researchers, software, and game sites

4. How Strongly were the Four Game Variations Solved?

When describing a game as being solved, Victor Allis lists three levels of solving that have been widely adopted (Allis, 1994).

- **ultra-weakly solved:** The initial theoretic game value is known.
- **weakly solved:** From the initial position, there is a known strategy to attain the theoretic game value.
- **strongly solved:** From any legal position, there is a known strategy to attain the position's theoretic game value.

Schaeffer and Lake define an additional level (Schaeffer & Lake, 1996).

- **ultra-strongly solved:** From any legal position, there is a known strategy to increase the probability of a fallible (non-perfect) opponent making an error, resulting in an outcome that exceeds the position's theoretical game value.

Finally, Gévay and Danner add yet one more level (Gévay & Danner, 2014).

- **extended strong solution:** The game is strongly solved, even for positions that are not legally attainable.

For this research, the databases for each of the four combinations of rules were solved with an extended strong solution. So even impossible positions such as Figure 11 were still evaluated.

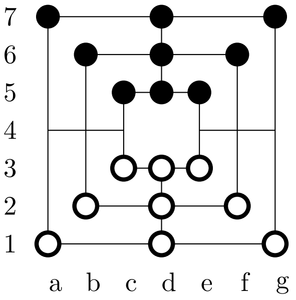

Figure 11: Impossible position, either side to move and draw

The extended solution included positions with pieces on the board that should have been captured (as in Figure 11 above) and opening positions with fewer pieces on the board than possible, such as 2 vs. 2 on ply 6. The purpose of including these impossible positions was to make the databases easier to generate since the program no longer had to determine whether or not a given position was legal. It also has the advantage of making it easier to compare the different rules' databases since they will contain all the same positions. For example, Figure 12 shows a position for ply 18 that is attainable with the Always Capture rule, but not with the Always Protected rule. In spite of this, the calculations were made for this position using both rules.

	White	Black
1.	a7	c5
2.	d7	d5
3.	f6	e5×f6
4.	g7×e5	e5×g7
5.	g7×e5	e5×g7
6.	g7×e5	e5×g7
7.	g7×e5	e5×g7
8.	g7×e5	e5×g7
9.	g7×e5	

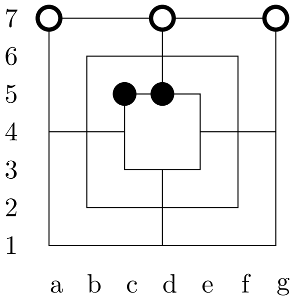

Figure 12: A contrived but legal ply 18 position for the Always Capture rule

The only restriction adhered to was that the number of pieces a player could have on the board could not exceed the number of pieces that the player had placed. For example, in Figure 12 above, the same arrangement could not be used for ply 5 as White is shown as having 3 pieces on the board even though he has already played only twice.

5. Generating the Databases with Retrograde Analysis

The methods for generating the databases for the middle and endgame (moving and jumping phases) are very similar. Therefore, the two databases are often grouped together and are sometimes referred to collectively as the "middle/endgame database."

Retrograde analysis was performed using techniques similar to Gasser's methods (Gasser, 1990) in which the game is analyzed in reverse, starting with the end of the game and working backwards towards the start. Gasser used retrograde analysis for the middle/endgame, then used an 18 ply forward alpha-beta search from the start (Gasser, 1996), thus solving the game. The approach used in the research for this paper was to solve the variations by applying retrograde analysis from the end of the game all the way back to the very beginning, producing complete perfect play databases.

The general approach that was taken in the programming of the database generator was to favor simplicity and accuracy. Since ample memory and computing power were available (see Appendix A), clever short cuts and memory reducing techniques were avoided in favor of reliability.

5.1 Symmetries

Before the positions could be evaluated, it was first necessary to make a list of all possible positions. The number of positions, legally attainable or not, that can be formed by placing up to 9 pieces of each color on the 24 board locations can be found by the following formula:

$$n = \sum_{w=0}^{9} \sum_{b=0}^{9} \frac{24!}{w!\, b!\, (24 - w - b)!} = 143{,}122{,}694{,}691$$

The number of positions needing to be evaluated can be greatly reduced by taking advantage of the high degree of symmetry of the board.

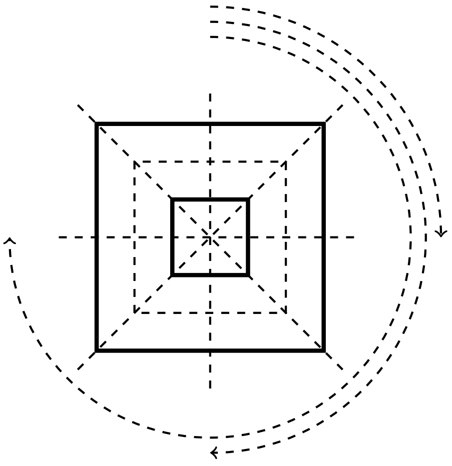

Figure 13: Board symmetries

The symmetries shown in Figure 13 include four reflections (vertical axis, horizontal axis, and two diagonal axes), three rotations (90°, 180°, 270°), and an inversion symmetry in which the outer and inner squares are exchanged. Of these symmetries, only four are actually needed with the rest being redundant. The four symmetries that were chosen were three reflections (vertical axis, horizontal axis, and the positively sloped diagonal axis) and the inversion. The other symmetries can be expressed as a combination of these four. For example, a rotation of 90° is the same as a reflection over the vertical axis followed by reflection over the diagonal axis.

Another symmetry that could be used in the middle and endgame is color symmetry. If it is White's turn to play a certain position, this is the same as if it were Black's turn to play with all the colors reversed.

There is one more symmetry that could have been used. For the special case of 3 vs. 3, since both White and Black can jump, all three squares can be exchanged with each other. This would further reduce the number of unique positions needing to be evaluated, but only by a minuscule amount compared to the entire database. Therefore, this extra symmetry was not utilized.

To take advantage of these symmetries, a perfect (reversible) hash function was created which reduced the state of the board to a single numeric value. To determine if a position was redundant, all 16 combinations of the four transformation symmetries were applied to the position, creating up to 16 unique hash values. If the original hash value matched the smallest of these 16 hash values, then that position was used in the database. Otherwise, the position was considered redundant. Using these symmetries, the total number of unique positions needing to be evaluated drops to 8,904,593,601 in the endgame database, and 17,874,891,168 in the opening database. (See Section 5.5 for reasons for the difference.)

5.2 Data Representation

Two important decisions about how to represent the data had to be made:

1. How to represent the hash values?

2. How to represent the theoretical game value?

5.2.1 The Hash Values

For determining the hash values, each of the three concentric squares on the board was described as an 8-digit, base-3 number. Each digit represented which piece was located at each of the 8 positions on the square using the values 0=empty, 1=White, 2=Black. These three 8-digit numbers (one for each square) could be manipulated separately, or be combined into a single 24-digit, base-3 number. (For the sake of speed and ease of bit manipulation, these base-3 values were temporarily converted to base 4 when applying the transformations.) The least significant digit (LSD) was chosen to be the top side of the square, **d7**, proceeding clockwise around the square to the most significant digit (MSD) of the outer square, **a7**, as shown in Figure 14. The outer square was considered the least significant 8-digit number while the inner square was the most significant.

Figure 15 shows an example board state in which the inner square is represented by 00000012_3, the middle square by 00020010_3, and the outer square by 02001100_3, expressed as a single 24-digit base-3 hash value of $00000012\ 00020010\ 02001100_3$.

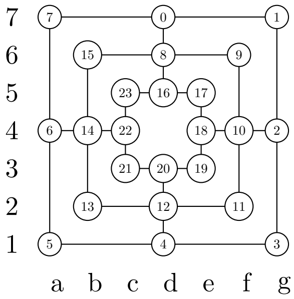

Figure 14: LSD (0) to MSD (23)

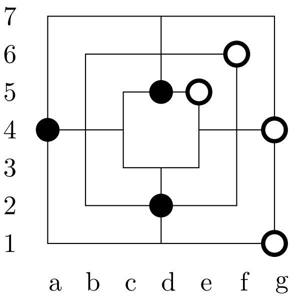

Figure 15: hash=$00000012\ 00020010\ 02001100_3$

The maximum possible 24-digit, base-3 number, $3^{24} - 1$, requires 39 bits, so these values were stored as a 40 bit (5 byte) number, wasting only 1 bit.

5.2.2 The Theoretical Game Values

In order to accommodate an unknown maximum number of possible plies, a 16-bit (2 byte) number was used to store the theoretical game value for each position. This proved to be a wise choice as the maximum number of plies turned out to be 331 which would not have fit into 1 byte.

Since the winner in Nine Men's Morris is always the player to make the final move, it is possible to represent a position's theoretical game value in one of two simple ways:

- The unsigned number of plies (half moves) to the end of the game (odd plies are wins, evens are losses)
- The signed number of full moves to the end of the game (positives are wins, negatives are losses)

The latter option of using full moves was chosen as this makes it easier to compare game values, i.e., positives are better than negatives. Zero was chosen to represent a draw so that winning values > draw value > losing values. If one were to plot these values on a number line, they would be in the order shown in Figure 16.

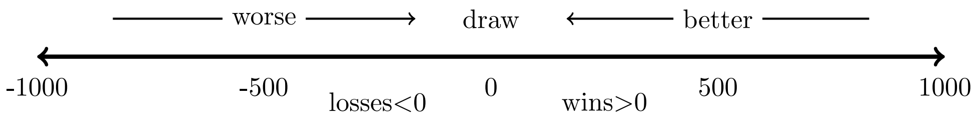

Figure 16: Moves to win or lose plotted on number line, immediate win = 1

While it is easy to store values this way, it has the undesirable characteristic that smaller positive values are better than larger positive values since a win in 3 moves is better than a win in 10 moves. Losses have the same issue but with negatives. In order to avoid this situation, the values were "inverted" by subtracting the values from a "bias" constant. For example, using a bias of 1000, a win in 1 move was stored as 999, a loss in 1 move became -999, a win in 2 became 998, etc. Any two game values, be they win, lose, or draw, could now be compared with simple signed integer comparison, as shown in Figure 17.

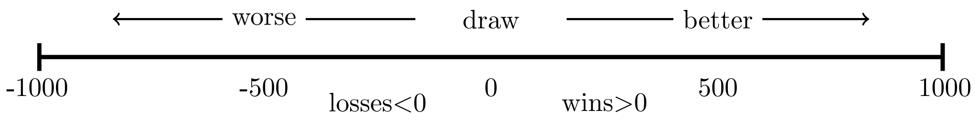

Figure 17: Inverted moves to win or lose, immediate win = 999

5.3 Differences from Previous Implementations

The retrograde analysis implementation used in this research contained a few differences from that used by previous researchers.

5.3.1 Returning the Board to its Pre-Move State

In analyzing a position in the game of checkers, Lake, Schaeffer, and Lu had their program call a function `MakeMove()` for each possible move, then get the game value for this new position with `GetValue()`, then return the board to its original state with `UnMakeMove()` (Lake, Schaeffer, & Lu, 1994) as shown in Figure 18.

```
for( i = 0; i < numb; i++ ) {
   MakeMove( pos, moves[ i ] );
   childvalue = GetValue( ... ,Index( pos ) );
   UnMakeMove( pos, moves[ i ] );
   ...
```

Figure 18: Code snippet using `UnMakeMove()`

The approach used for this research was to instead make a copy of the original position before the move was made. After finding the value of the new position, the copy was simply discarded, eliminating the need to "unmake" a move. The actual code looked different, but the equivalent concept can be expressed as indicated in Figure 19. Positions were stored as a 24 element array of enumerated values, requiring 24-96 bytes per position (depending on the compiler). Due to the speed at which small block memory copies can be performed, this approach proved to be faster.

```
for( i = 0; i < numb; i++ ) {
   CopyPosition( pos_copy, pos );
   MakeMove( pos_copy, moves[ i ] );
   childvalue = GetValue( ... ,Index( pos_copy ) );
   ...
```

Figure 19: Code snippet using `CopyPosition()`

5.3.2 Retrograde Steps

Another difference from some implementations has to do with how the retrograde step is actually accomplished. One method is to scan the entire subspace looking for positions that can lead to opposing positions for which the game values have already been determined. The other approach is to take each position whose game value has already been determined and then, by applying the rules in reverse, ascertain all the possible previous positions that could have led to this current position. With either method, any position leading to "lose in 6 plies" for the opponent becomes a "win in no more than 7 plies" for the current player.

While it is a straightforward matter to determine what legal moves a player can make from a given position, it can be a rather complex process to determine all the possible ways to arrive at a given position in Nine Men's Morris. Even though this latter method is generally faster (Cazenave & Nowakowski, 2015), the former method of scanning the entire subspace was chosen for its simplicity and reliability.

5.3.3 Hash Functions

The most significant difference from earlier researchers' methods is the manner in which the game values were looked up. Typically, a hash function converts a game state to an index which is then used to look up a game value in a table. Ideally, the hash function would produce indices that range from 0 to $(N-1)$ where N is the number of possible game states. However, creating an ideal hash function such as this can be difficult. Gasser started with a mathematical function having a range that exceeded the ideal by 18.26%, which he then showed could be improved to only 5.59% above the ideal (Gasser, 1995).

For this research, a different approach was tried. Instead of using a mathematical function to approximate an ideal index, the actual ideal index was determined, resulting in databases containing no extraneous entries. To accomplish this, two parallel arrays were used for each subspace. One was a hash array containing a sorted list of every possible subspace hash value, while the other array contained the corresponding game values. To lookup a game value, the game state's hash value was first looked up in the hash array using a binary search. The resulting index was then used to directly retrieve the game value from the value array. In essence, the hash array search acted as an ideal hash function.

Of course, this approach required more memory for the hash array, but the additional memory requirement (average = 0.4 GB, maximum = 2.75 GB) was well within the limits of the hardware. Saving these hash arrays similarly took up more disk space. This was somewhat offset by the smaller database files as they had no extraneous entries. The file size per subspace was more significantly reduced by the fact that the same hash file could be used with both the middle/endgame databases and the opening game databases for all the rule variations. For example, the 5-4 hash file can be used with 5-4 subspace for both of the middle game rules as well as the nine 5-4 subspaces for all four combinations of the opening game rules for a total of 38 subspaces which all share the same 5-4 hash file. In the end, the hash files increased the total file space by 14%.

5.4 Generating the Middle & Endgame Databases

Once the unique positions were determined, the process of generating the endgame database could begin. As the different White vs. Black (W-B) subspaces were calculated, it was always assumed that the number of White pieces was as least as many as the number of Black pieces. For cases in which there were more Black pieces on the board, the colors were simply reversed.

Starting with the smallest possible number of pieces on the board, 3 vs. 3, each position was initially marked as a draw. The entire 3-3 subspace was scanned for positions that could be won immediately. These positions were then marked as "win in 1 ply." Then, the subspace was scanned again for unresolved entries in which all possible moves led to a "win in 1 ply" for the opponent. These positions were then marked as "lose in 2 plies." The process continued: "win in 3 plies," "lose in 4 plies," etc. If a pass through the subspace made no additional changes, then the calculation for that subspace was complete. Any positions that had not yet been identified as either a win or a loss were left as draws. The process for the middle game subspaces, where both players have more than 3 pieces, was similar with the exception that the process started by marking positions that were already lost from being blocked in. (It is impossible to block in an opponent in the endgame.)

After the 3-3 subspace was complete, the 4-3 subspace could be calculated. The 3-3 and the 4-3 could not be calculated in parallel as the 4-3 depends on the 3-3. However, once the 4-3 was finished, both the 5-3 and the 4-4 subspaces could be calculated in parallel since neither depends on the other. The 5-4 subspace would have to wait until both the 5-3 and 4-4 calculations were finished since it depends on both. Once the 9-9 subspace was finished, the entire middle/endgame database was complete. The diagram in Figure 20 shows the dependencies of the middle/endgame subspaces.

The pseudo-code in Algorithm 1 shows the process used calculating a middle game database subspace for **W** white pieces and **B** black pieces on the board with Black to move.

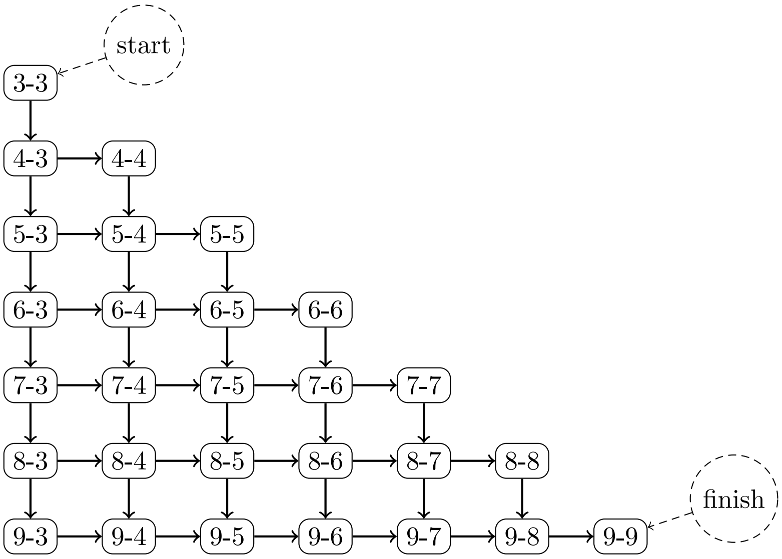

Figure 20: Evaluation sequence for middle/endgame databases

Algorithm 1 Middle game database calculation for the W×B subspace, Black to move

```
for each position, $pos_0$, in the W×B subspace do
    $pos_0 GameValue$ ← DRAW
end for
$targetGameValue$ ← lose in zero ply (i.e., game is already lost)
repeat
    for each position, $pos_0$, in the W×B subspace still marked as a DRAW do
        $worst$ ← impossibly high game value
        for each possible legal move from $pos_0$ do
            $pos_1 \leftarrow pos_0$
            apply move to $pos_1$
            if the move does not result in a mill that can capture then
                $worst$ ← min($worst$, game value of $pos_1$ in W×B )
            else
                for each possible capture from $pos_1$ do
                    $pos_2 \leftarrow pos_1$
                    apply capture to $pos_2$
                    $worst$ ← min($worst$, game value of $pos_2$ in (W−1)×B )
                end for
            end if
        end for
        if $worst = targetGameValue$ then
            $pos_0 GameValue$ ← OnePlyLonger($worst$)
        end if
    end for
    $targetGameValue$ ← OnePlyLonger($targetGameValue$)
until no changes took place in last pass through W×B subspace
```

5.5 Generating the Opening Game Databases

While the Always Protected vs. Always Capture rule choice affects all phases of the game, the Take 1 vs. Take 2 rule choice only affects the opening phase. (Closing two mills simultaneously is only possible when placing a piece, not moving a piece.) Therefore, all four combinations of the rules apply to the opening:

- Always Protected, Take 1
- Always Protected, Take 2
- Always Capture, Take 1
- Always Capture, Take 2

The opening database calculations involved some significant differences from that of the middle/ endgame. With the endgame, the calculations start with the small 3-3 subspace and work backwards to the large 9-9 middle game subspace. As shown in Figure 21, the opening database calculations start with the last opening move, ply 18, which could be the large 9-8 subspace, and work backwards to the beginning of the game, ply 1, which is always the tiny 0-0 subspace containing only a single entry—a completely empty board. While the middle/endgame database contains a total of 49 subspaces, in the opening database, ply 18 alone has 90 subspaces. For each of the four opening databases, there are a total of 615 subspaces.

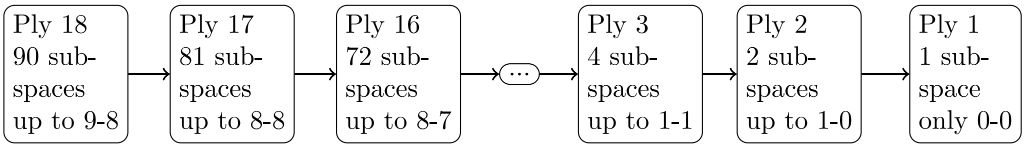

Figure 21: Evaluation sequence of the opening game database

With the middle/endgame database, it was possible to reverse the colors so that a 5-7 position could be treated as a 7-5. This is not possible with the opening database as the ply dictates which color's turn it is to play. Reversed colors would be inconsistent with the ply.

In the middle/endgame it is possible to make dozens of moves before a capture occurs, taking the play out of one subspace into another. However, the entire opening database is just 18 plies long with each subspace being only 1 ply long. This means that each opening subspace can be calculated in a single pass. For example, if the game is in a ply 12, 6-5 position, even if no capture occurs, the next play will be from a ply 13, 6-6 position—a completely different subspace. Furthermore, if the next two plies each result in a capture, the play will continue from a ply 15, 6-6 position which is yet another subspace. The increased number of subspaces makes the opening database about twice the size of the middle/end game database, but the shallower depth makes the opening database much faster to calculate.

One implication of the Take 2 rule is that a subspace is dependent on more previously calculated subspaces. In the middle/endgame, a subspace could be dependent on as many as two previously calculated subspaces, such as the 7-6 being dependent on the 7-5 and the 6-6. In the Take 2 opening database, the ply 14, 7-6 subspace is dependent upon the ply 15, 7-7 (no captures), the ply 15, 6-7 (one piece captured), and the ply 15, 5-7 (two pieces captured). This means that even more memory was required as more subspaces were loaded into RAM.

In keeping with the extended strong solution, positions were considered even if there were fewer pieces on the board than physically possible. For example, for ply 6, White will have already played 3 pieces and Black will have played 2 pieces. In an actual game, the board could have either 3-2 or 3-1 if White had already captured a piece. For the purposes of database calculations, all 12 subspaces were calculated from 0-0 to 3-2 (0-0, 1-0, 2-0, 3-0, 0-1, 1-1, 2-1, 3-1, 0-2, 1-2, 2-2, 3-2). This was easier to program and the extra database files were insignificant in size.

The pseudo-code in Algorithm 2 shows how each opening game database subspace was generated.

Algorithm 2 Opening database calculation for the ply P, W×B subspace, Black to move

```
for each position, $pos_0$, in the ply P, W×B subspace do
    $worst$ ← impossibly high game value
    for each possible move from $pos_0$ do
        $pos_1$ ← $pos_0$
        apply move to $pos_1$
        if the move does not result in a mill that can capture then
            $worst$ ← min($worst$, game value of $pos_1$ in P+1,W×(B+1) )
        else
            for each possible capture from $pos_1$ do
                $pos_2$ ← $pos_1$
                apply capture to $pos_2$
                if rules or position dictate that only 1 capture is possible then
                    $worst$ ← min($worst$, game value of $pos_2$ in P+1,(W−1)×(B+1) )
                else
                    for each possible 2nd capture from $pos_2$ do
                        $pos_3$ ← $pos_2$
                        apply 2nd capture to $pos_3$
                        $worst$ ← min($worst$, game value of $pos_3$ in P+1,(W−2)×(B+1) )
                    end for
                end if
            end for
        end if
    end for
    $pos_0GameValue$ ← OnePlyLonger($worst$)
end for
```

6. Verification of Databases

Each database was generated twice using different compilers to avoid compiler errors and hardware issues such as memory or hard drive errors. A checksum for each file was created so that the integrity could be verified upon transmission to another location. This double checking essentially eliminates the possibility of hardware and compiler errors, but of course does not detect programmer errors.

A verification program was successfully used, but since the same person wrote the original code, this could not be considered an independent verification. To truly verify the databases, confirmation would need to come from an external source. Fortunately, Gábor Gévay and Gábor Danner, as well as Peter Stahlhacke, agreed to help verify the specific databases that corresponded with their own. Rather than trying to directly compare each of the billions of entries, it was decided instead to follow the example of Jonathan Schaeffer and Ed Gilbert who verified each other's checkers 10-10 endgame databases by comparing the win/lose/draw frequencies for each subspace (Schaeffer, 2009, p. 478) (Gilbert, 2005). It would be possible, though quite unlikely, for a program to make offsetting errors such that the win/lose/draw record was the same.

To prevent even this unlikely possibility of an accidental match, the verification went one step further by comparing the frequency distribution of plies to the end of the game. Using the 4-4 Always

Capture middle game database as an example, instead of merely saying that the win/lose/draw frequencies were 159/29/3,225,409 the frequencies of all the "plies to end" were compared as shown in Table 2. Notice that the frequencies of wins (odd plies) add to 159 while the losses add to 29. With this additional information, the chance of an accidental match is essentially zero. Two independently generated databases whose ply frequencies match for all the subspaces should, for all practical purposes, be regarded as verified.

plies to end:	draws	already lost	1	2	3	4	5	6	7	8	9
frequency:	3,225,409	6	29	3	24	10	46	6	48	4	12

Table 2: Frequency distribution of plies for 4-4 Always Capture endgame

The ply frequencies from Gévay and Danner's Always Capture, Take 1 opening database and Always Capture middle/endgame database matched perfectly, as did Stahlhacke's Always Protected middle/endgame database. The format of Stahlhacke's Always Protected, Take 2 opening database was such that it did not allow easy extraction of the ply frequencies. For that opening database, a rudimentary verification was done by taking a number of randomly selected positions, as well as some hand selected "error prone" positions, and comparing the game values to the values reported by Stahlhacke's commercial software, *Mühle Datenbank Advanced* (`http://muehle-24.de/`). Each entry tested was found to be an exact match. Selected positions from the Always Capture, Take 2 opening database were similarly compared with results from the commercial software, *Brillant Mill* (`http://www.brillant-mill.eu/`) and were found to match as well.

The only portion of the databases that was not verified externally was the Always Protected, Take 1 opening database as there are, at this time, no other known instances of this database. The generator for this database shares nearly all of its code with either the Always Protected, Take 2 and Always Capture, Take 1 code, so similar results would be likely.

7. Results: The Effects of the Rule Variations

The opening databases prove definitively that the game is a draw for all four combinations of the rule variations. A number of corollary questions naturally follow:

- How much of an effect do these variations have on each phase of the game?
- How early in the game can the differences affect the play?
- How early in the game can the differences change the ultimate outcome?

7.1 Effects on the Middle and Endgame

In actual play, the differences in game play between the Always Protected and Always Capture rules do not typically become evident until the endgame when one or both players is down to three pieces (Schürmann & Nüscheler, 1980, p. 44). However, in the perfect play databases, the effects of the rules differences are seen throughout the game. (Simultaneous mills can only be formed in the opening game, so the Take 1 and Take 2 rules have no bearing on the middle/endgame.)

In the endgame, the difference in rules does not seem to make a large difference in the outcome of the more even games (3 vs. 3 to 6), but it does make a noticeable difference in the more lopsided positions (3 vs. 7, 8, or 9). (See Appendix B for details.) Figure 22 shows a rare 7-3 example where the rule does make a difference.

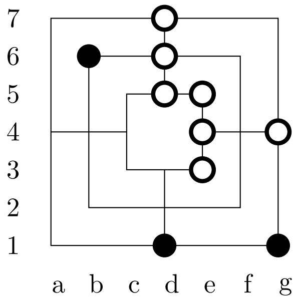

Figure 22: White to move and draw (Always Protected), or win in 17 (Always Capture)

In the 3-3 subspace, there is not a single case in which the win/lose/draw outcome would have been different, but there are cases in which the outcome is easier to achieve, as shown in Figure 23.

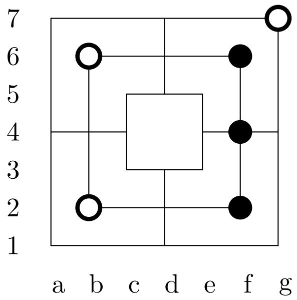

Figure 23: White to move and win in 23 (Always Protected), or win in 1 (Always Capture)

In the 4-4 subspace with either side to play, and in the 4-3 with Black to play, there are no differences at all between the two databases. In the 4-3 subspace with White to play, there are only 748 out of 760,398 possible positions that would have been immediate wins with Always Capture that are draws with Always Protected.

There are numerous cases in which changing from Always Protected to Always Capture will change the outcome from a draw to either a win or loss. However, there are very few cases in the middle/endgame in which the outcome is reversed, where a loss is changed to a win or vice versa. In fact, there are no such cases when both players have fewer than 6 pieces on the board. There were a total of only 3533 cases out of nearly 9 billion in the extended solution middle/endgame databases in which the outcome was reversed, and the vast majority of these were positions that would be impossible to reach in actual play. Of these, only 433 such positions were even remotely possible. Figure 24 shows an example from the extended solution database in which White wins with Always Protected but loses with Always Capture.

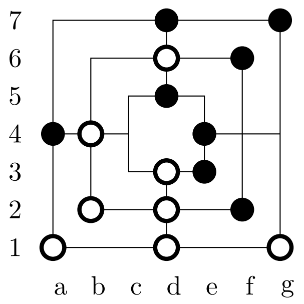

Figure 24: White to move and win in 23 (Always Protected), or lose in 30 (Always Capture)

Table 3 shows a summary of how changing from the Always Protected rule to the Always Capture rule affects the outcomes of all the positions in the middle/endgame database.

total positions	8,904,593,601
draws to losses	29,145,537
draws to wins	34,254,690
losses to draws	350,087
wins to draws	450,042
losses to wins	2,717
wins to losses	816

Table 3: Endgame outcome differences changing from Always Protected to Always Capture. Indicated outcomes are relative to current player.

One statistic that is a curiosity to the game theorist is the maximum number of plies to the end of the game. The maximum number of plies for the middle/endgame is 329 for the Always Protected rule and 204 for the Always Capture rule. (See Appendix C for more detailed information about the maximum number of plies for each subspace.)

These large ply counts go well beyond the capability of human players. Of greater interest to the player is how often a game becomes significantly easier to win by using one rule or the other. Approximately 0.3% of the middle/endgame positions have game values that do not change in outcome but differ by more than 30 plies. Of these, about 1/3 were easier to win using Always Protected and 2/3 with Always Capture.

7.2 Effects on the Opening Game

The opening game is affected by the Always Protected vs. Always Capture rule choice as well as the Take 1 vs. Take 2 rule choice. One of the goals of this investigation was to determine how early in the game would these rules make a difference.

7.2.1 Always Protected vs. Always Capture Rules

The Always Protected vs. Always Capture rule choice is usually thought to mostly affect the endgame, so it is somewhat surprising that this rule can impact the theoretical game-value as early as Black's second move. By ply 5, there are 13 positions in which changing from Always Protected, Take 2 to Always Capture, Take 2 changes the outcome to or from a draw. Examples of these are shown in Figure 25. In the later plies of the opening, this occurs more often, but remains relatively infrequent. It is not until ply 14 that an outcome is completely reversed from a loss to a win or vice versa in the extended solution.

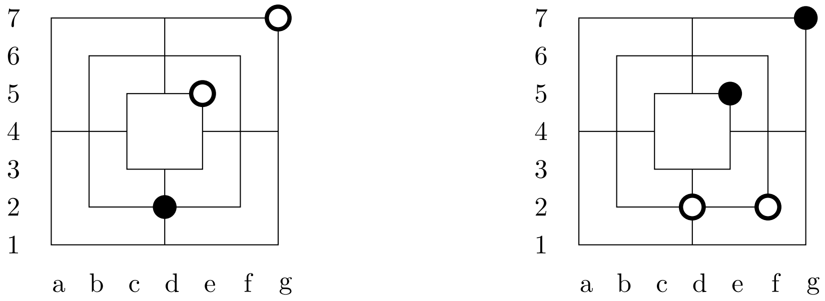

Figure 25: Left: Ply 4, Black to move and win in 79 (Always Protected) or 55 (Always Capture). Right: Ply 5, White to move and draw (Always Protected), or win in 67 (Always Capture).

7.2.2 Take 1 vs. Take 2 Rules

The Take 1 vs. Take 2 rule choice determines how many to capture if two mills are closed simultaneously. The earliest that simultaneous mills can physically occur is on ply 9 (White's 5th move). Surprisingly, the effects from this rule can impact the game as early as ply 5 where it can already change the outcome. The position shown in Figure 26 is a draw with the Take 1 rule, but is a win with the Take 2 rule.

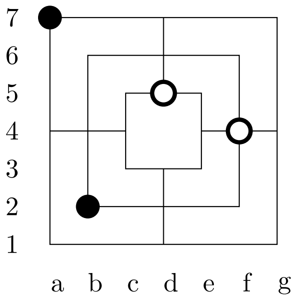

Figure 26: Ply 5, White to move and draw (Take 1), or win (Take 2)

What makes Figure 26 particularly interesting is that if both players were to play perfectly from this point on, the simultaneous mills would never actually be formed, or even need to be directly blocked by Black. Merely having the threat of simultaneous mills down the road is sufficient to cause Black to avoid the situation altogether. If Black had not taken the Take 2 rule into account, White would have eventually formed simultaneous mills on ply 17.

This is consistent with what experienced players have found. When asked about the occurrence of simultaneous mills in tournament play, Daniel Lehner, President of the WMD, stated, “I personally never saw it happen in a tournament that a player was able to close two mills at once. This would be such a tremendous advantage that the opponent will always try to prevent it” (Lehner, 2016).

7.2.3 Combinations of Rules Taken Together

One way to look at the overall effect that the various rules have on the opening game is to examine the statistics for ply 18, the last move of the opening, to see if the rules change the number of draws. Table 4 shows that the number of draws is slightly reduced by the Always Capture, Take 2 rules.

	Wins for White	Wins for Black	Draws
Always Protected, Take 1	17.34	62.53	20.13
Always Protected, Take 2	17.28	62.67	20.05
Always Capture, Take 1	17.57	62.88	19.56
Always Capture, Take 2	17.50	63.02	19.48

Table 4: Win percentages at ply 18

Note that the percentages shown in Table 4 are the percentage of entries in the ply 18 database subspace that are wins for White or Black. This does not mean that in a game between two human players that Black has a 63% probability of winning (unless players played randomly in the opening and then perfectly in the middle/endgame). At the same time, it does give some credence to the belief among experts that Black has a slight advantage. While White can lead the attack by going first, Black has the advantage of being able to place the last piece in the opening phase (Schürmann & Nüscheler, 1980, p. 74).

Figure 27 shows a summary of how changing from one of the four combinations of rules to another affects the outcomes of all the positions in the opening database. As can be seen, the Always Protected vs. Always Capture rule choice has a greater impact than the Take 1 vs. Take 2 rule choice on reducing the number of draws and has a lesser impact on reversing win/loss outcomes.

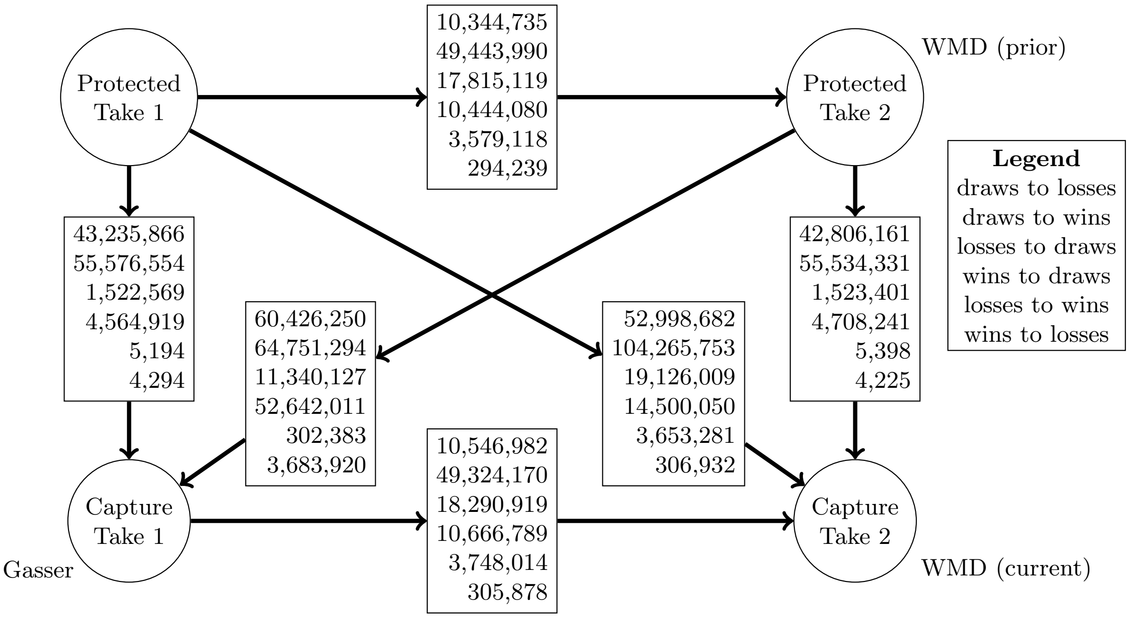

Figure 27: Outcome differences resulting from changing from one rule to another, out of 17,874,891,168 opening database entries. Indicated outcomes are relative to the current player.

7.3 Conclusions

The overall conclusion is that none of the four combinations of rules affect the initial theoretical game value—the game is always a draw.

The overall impact of changing from the Always Protected rule to the Always Capture rule has the desired effect of slightly reducing the number of draws without having the undesirable effect of significantly reversing the outcome of games.

Using the Take 2 rule rather than the Take 1 rule has a much smaller impact on the game, but very slightly further reduces the number of draws.

Acknowledgments

I wish to thank Alan Davis of the Rosslyn Academy IT Dept. for the use of a spare server whose specifications made this project much more manageable.

I would also like to thank Gábor Gévay, Gábor Danner, and Peter Stahlhacke for their generous cooperation in the process of independently verifying the databases.

Additional thanks goes to Gábor Gévay and Gábor Danner for their valuable suggestions concerning this paper.

Appendix A. Hardware and Software

The endgame databases were generated in 2014-15 on a server with dual Intel Xeon E5620 quad-core 2.40 GHz processors (8 total physical cores) with 96 GB of RAM running Windows Server 2012. These specifications allowed the entire subspace along with the subspace dependencies to be loaded entirely into RAM, with no need for data compression. For instance, when calculating the 9-8 subspace, the 9-7 and 8-8 subspaces and hash tables were also loaded into memory, requiring about 15 GB for this instance of the program. The 8 cores and 96 GB RAM still allowed several subspaces to be calculated simultaneously.

For confirmation purposes, both the Always Protected and the Always Capture endgame databases were calculated twice using two different C compilers, Microsoft C 18.0 and GCC (MinGW) 4.8.2. The entire run time for both endgame databases and their confirmations took about two and a half months.

The opening game databases were generated in 2015 on an Intel i7-3770 quad-core 3.40GHz processor (4 physical cores) and 24 GB RAM running Windows 10 using the Microsoft C 18.0 compiler. Since the opening database generator could require as much as 18 GB of RAM, it was not possible to run multiple instances in parallel, but even so, the databases for all four rule combinations were generated in about 10 days.

For confirmation, all four opening databases were recalculated a second time on the same physical machine, but running on Linux Mint 17 under Virtualbox, using the GCC 4.8.2 C compiler.

Appendix B. Win/Lose/Draw Data

Tables 5 and 6 show a summary of the win/lose/draw percentages for both the Always Protected and the Always Capture rules. The table assumes that it is White's turn to move. The number of White pieces on the board is shown down the left column and the number of Black pieces is shown across the top row. The symbol "0+" is used to indicate a small non-zero value that rounds to 0.0%.

W\B	**3**	**4**	**5**	**6**	**7**	**8**	**9**
3	82.9/16.9/0.2	13.5/0/86.5	0.2/0.4/99.4	0/2.4/97.6	0/39.2/60.8	0/76.6/23.4	0/99.0/1.0
4	9.8/0.4/89.8	0+/0+/100.0	0+/0+/100.0	0+/4.4/95.6	0+/71.1/28.9	0+/91.1/8.9	0+/99.5/0.5
5	22.3/0/77.7	0.1/0+/99.9	0.1/0+/99.9	0+/4.4/95.5	0+/59.7/40.3	0+/86.8/13.2	0+/98.1/1.9
6	39.8/0/60.2	16.9/0+/83.1	17.0/0+/83.0	10.9/2.4/86.7	5.5/36.6/57.9	1.9/69.8/28.3	0.5/93.2/6.3
7	88.1/0/11.9	91.3/0+/8.7	87.4/0+/12.6	73.2/1.0/25.8	46.6/14.8/38.6	21.1/41.7/37.2	7.8/76.0/16.2
8	98.4/0/1.6	98.7/0+/1.3	97.9/0+/2.1	93.4/0.3/6.4	79.7/4.3/16.1	53.6/18.0/28.4	28.3/46.9/24.9
9	100.0/0/0+	100.0/0+/0+	99.9/0+/0.1	99.2/0.1/0.7	95.4/1.1/3.5	82.5/6.6/10.9	58.6/22.3/19.1

Table 5: Rounded win/lose/draw percentages for Always Protected middle/endgame

W\B	**3**	**4**	**5**	**6**	**7**	**8**	**9**
3	82.9/16.9/0.2	13.5/0/86.5	0.2/0.4/99.4	0/2.8/97.2	0/47.3/52.7	0/90.8/9.2	0/100.0/0+
4	9.9/0.4/89.7	0+/0+/100.0	0+/0+/100.0	0+/6.4/93.6	0+/73.0/27.0	0+/93.0/7.0	0+/99.9/0.1
5	22.5/0/77.5	0.1/0+/99.9	0.1/0+/99.9	0+/6.7/93.3	0+/60.4/39.6	0+/87.6/12.4	0+/98.5/1.5
6	40.0/0/60.0	23.2/0+/76.8	23.9/0+/76.1	14.9/3.4/81.7	7.1/37.0/55.9	2.4/70.4/27.2	0.7/93.5/5.8
7	91.1/0/8.9	92.3/0+/7.7	87.9/0+/12.1	73.7/1.2/25.1	47.0/15.0/38.0	21.3/42.1/36.6	7.9/76.3/15.8
8	99.7/0/0.3	99.2/0+/0.8	98.1/0+/1.9	93.7/0.3/6.0	80.0/4.3/15.7	53.9/18.2/28.0	28.4/47.1/24.5
9	100.0/0/0+	100.0/0+/0+	99.9/0+/0.1	99.3/0.1/0.7	95.5/1.1/3.4	82.7/6.6/10.7	58.8/22.4/18.8

Table 6: Rounded win/lose/draw percentages for Always Capture middle/endgame

Appendix C. Maximum Plies to End of Game

Tables 7 and 8 show the largest number of plies to the end of the game for each middle/endgame subspace for the Always Protected and Always Capture rules. The number of White pieces on the board is shown down the left column and the number of Black pieces is shown across the top row.

W\B	**3**	**4**	**5**	**6**	**7**	**8**	**9**
3	26	33	31	6	38	40	40
4	32	9	28	300	300	214	142
5	3	29	57	312	308	302	232
6	7	301	311	321	**329**	323	321
7	39	299	303	328	318	321	321
8	41	213	285	322	320	315	320
9	39	95	163	318	312	319	319

Maximum = 329

Table 7: Middle/Endgame Always Protected maximum plies to end of game

W\B	**3**	**4**	**5**	**6**	**7**	**8**	**9**
3	26	33	31	6	30	34	30
4	32	9	28	156	112	112	110
5	3	29	57	162	160	160	114
6	7	157	163	167	184	186	173
7	31	111	159	185	181	**204**	202
8	33	111	153	185	203	196	202
9	25	103	113	172	201	201	191

Maximum = 204

Table 8: Middle/Endgame Always Capture maximum plies to end of game

Figure 28 shows middle game positions which have the largest number of plies until the end of the game for both the Always Protected and the Always Capture rules.

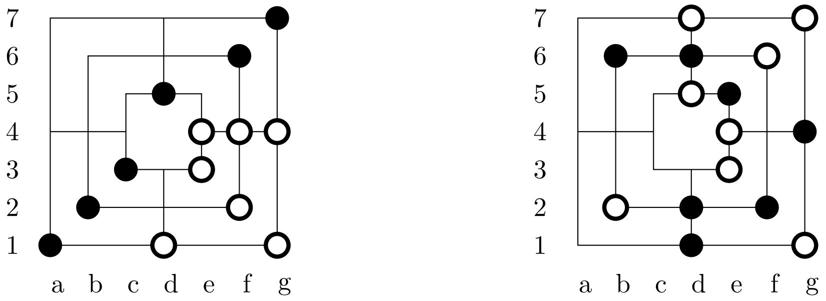

Figure 28: Maximum plies to end of game. Left: Black to move and win in 329 (Always Protected). Right: Black to move and lose in 204 (Always Capture).

Table 9 shows the maximum number of plies to the end of the game for each ply number in the opening database. These values are for the extended strong solution, so they include positions that are not attainable in actual play.

ply	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	**max**
Prot-1	-	35	38	125	245	263	283	297	330	329	**331**	330	329	**331**	330	329	328	329	**331**
Prot-2	-	35	38	125	245	263	283	297	330	329	**331**	330	329	**331**	330	329	328	329	**331**
Capt-1	-	35	37	125	124	179	187	186	188	197	201	204	203	202	**206**	205	204	204	**206**
Capt-2	-	35	37	125	124	179	187	186	188	197	201	**204**	203	202	201	203	**204**	**204**	**204**

Table 9: Opening game maximum plies to end of game

References

Allis, L. V. (1994). *Search for Solutions in Game and Artificial Intelligence.* Ph.D. thesis, Department of Computer Science, University of Limburg.

Brillant Mill Team (2016). private communication..

Cazenave, T., & Nowakowski, R. (2015). Retrograde Analysis of Woodpush. *Games of No Chance*, *63*, 47–56.

Edelkamp, S., Sulewski, D., & Yücel, C. (2010). GPU Exploration of Two-Player Games with Perfect Hash Functions. In *Proceedings of the Third Annual Symposium on Combinatorial Search (SOCS-10).*

Gasser, R. (1990). Applying Retrograde Analysis to Nine Men's Morris..

Gasser, R. (1993). Nine Men's Morris is a DRAW. rec.games.abstract.

Gasser, R. (1995). *Harnessing Computational Resources for Efficient Exhaustive Search.* Ph.D. thesis, Swiss Federal Institute of Technology Zürich.

Gasser, R. (1996). Solving Nine Men's Morris. *Games of No Chance*, *29*, 101–113.

Gévay, G. E., & Danner, G. (2014). Calculating Ultra-Strong and Extended Solutions for Nine Men's Morris, Morabaraba, and Lasker. *CoRR (to appear in IEEE Transactions on Computational Intelligence and AI in Games)*, *abs/1408.0032*.

Gilbert, E. (2005). Kingsrow. http://edgilbert.org/Checkers/KingsRow.htm.

Lake, R., Schaeffer, J., & Lu, P. (1994). Solving Large Retrograde Analysis Problems Using a Network of Workstations..

Lehner, D. (2016). private communication..

Lincke, T. (1994). Perfect Play using Nine Men's Morris as an Example. Master's thesis, Department of Computer Science, ETH Zürich.

Schaeffer, J. (2009). *One Jump Ahead: Computer Perfection at Checkers* (2nd edition). Springer.

Schaeffer, J., & Lake, R. (1996). Solving the Game of Checkers. *Games of No Chance*, *29*, 119–133.

Schürmann, H., & Nüscheler, M. (1980). *So Gewinnt Man Mühle.* Ravensburger.

Stahlhacke, P. (2003). The Game of Lasker Morris..

Stahlhacke, P. (2016). private communication..

Weltmühlespiel Dachverband (2008). Das Mühlespiel (The Morris Game). http://www.muehlespiel.eu/images/pdf/WMD_Spielregeln.pdf.

Weltmühlespiel Dachverband (2010). Turnierreglement (tournament regulations). http://www.muehlespiel.eu/images/pdf/WMD_Turnierreglement.pdf.