Theodore Tegos
103 Pembina Hall, UofA
Edmonton, Alberta
Canada, T6G2H8

Date: ____________

University of Alberta

Shooting The Last Arrow

by

Theodore Tegos

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Spring 2002

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Shooting The Last Arrow** submitted by Theodore Tegos in partial fulfillment of the requirements for the degree of **Master of Science**.

Jonathan Schaeffer

Martin Müller

Ryan Hayward

Andy Liu

Date: ________________

Abstract

Games provide a fruitful research area for Artificial Intelligence. For years, chess was considered the most significant field of computer game playing. However, in recent years the interest has shifted towards games that pose greater challenges than chess, such as Go and Amazons. The characteristic of such games is the large number of available moves in each position. Amazons offers an ideal test bed for existing and new search ideas.

This thesis presents the work behind Antiope, a strong Amazons program. The strength of Antiope lies in the use of endgame databases. Antiope is the first program to use endgame databases in Amazons. One type of databases store values related to Combinatorial Game Theory. This theory has been almost exclusively the domain of the mathematics community. This thesis presents the first investigation of adding Combinatorial Game Theory to a high-performance game-playing program.

Acknowledgements

Special thanks go to . . .

Jonathan Schaeffer, for being my supervisor and constantly encouraging me to get this thesis done.

Martin Müller, for being my supervisor and constantly encouraging me to get this thesis done.

The UofA GAMES Group, for their endless fascination with games.

All Amazons programmers, for expanding a small community among games researchers.

Finally, thanks go to **Brad Leith,** for putting up with my endless abstractions during the breaks from my work.

Contents

Chapter 1

Introduction

1.1 Games

Games are an important part of everyday life. They provide entertainment, education and mental stimulation. Humans have been aspiring to create mechanical players for hundreds of years. Early attempts to do so were proven to be clever hoaxes, such as the chess automaton *The Turk*, which was constructed by Baron von Kempelen in 1769. The advent of computer technology in the 1950's gave researchers the means to develop game-playing programs.

The development of game-playing programs has been associated with the research area of Artificial Intelligence (AI) from the very beginning. The words of the Russian mathematician Alexander Kronrod that "chess is the *drosophila*[1] of artificial intelligence" reveal how much interest it has attracted from AI researchers. The predominant reasons that computer game playing has attracted so much interest are as follows:

- Researchers use game-playing programs to gain insights into the thought processes of the human brain. Playing games requires mental activities that are not understood very well, such as generalizing, pattern matching, searching, learning, and reasoning by analogy.

- Games provide a finite domain with well-defined rules and goals. Success is easily quantifiable; algorithmic enhancements translate into a higher winning percentage. Research on computer game playing will provide insights into the complexity of AI's ultimate goal, constructing machines that exhibit the intellectual properties of human beings.

- Games are also fun and a favourite pastime of humans. Programming computers to play games combines the fun of programming with the fun of playing games.

A lot of progress has been made beginning with the early ideas of Claude Shannon [Sha50] and Alan Turing [TSBB53] on chess playing programs and Christopher Strachey's draughts program [TSBB53], to the achievement of world-champion status by programs such as Chinook in checkers [Sch97] and Deep Blue in chess [SP97].

A lot of research has gone into developing strong game-playing programs. There are three main elements that make a program a strong player. The first

[1]The fruit fly is to genetics, as chess is to AI.

one is hardware. The faster the underlying machine is, the more calculations a program will be able to perform within a given time limit. Apart from a faster CPU, another way to take advantage of hardware is by parallelizing a program.

The second element is the search algorithm together with the various search enhancements. Game-playing programs analyze ahead a number of moves in order to evaluate a position. If the search is efficient then programs will be able to look more moves ahead, thus evaluating a position better. So a more efficient search implies improved performance.

The third element is the amount of knowledge incorporated into the program. Knowledge can be used extensively to evaluate the positions reached in a search. It is easy to suppose that the more knowledge that is put into the search, the better it is. However, knowledge can be difficult to integrate in a program since sometimes it is hard to specify or quantify its effect, while it may also be expensive to calculate.

One of the major contributions of games to the field of AI has been the acceptance of brute-force search as an effective search method. Brute-force search methods examine all the possible paths that lead to a solution and then select the best one. They do not use much application-dependent knowledge to limit the search space.

The belief that brute-force search is dumb because it builds large search trees has been deeply rooted in AI and researchers have been striving for years to find effective methods to incorporate more application-dependent knowledge into their programs. Games have proven again and again that search is knowledge and brute-force search can be effective. The implicit knowledge contained in large search trees can make up for the imprecise heuristic knowledge that could be explicitly incorporated into a search algorithm.

The search method most often used by computers in games is *minimax search*, in which all the combinations of moves and replies are considered for the two players, extending to a certain number of moves into the future. A tree is constructed this way where each level represents the moves of a player, with players alternating at each level. One player is supposed to be trying to maximize the score of the game while the opponent tries to minimize it, hence the name minimax.

The minimax algorithm produces large trees since it takes into account all the move sequences. It turns out that many branches of the game tree need not be examined since they cannot alter the final result. *Alpha-Beta pruning* is an enhancement to the basic minimax search that takes advantage of this characteristic and reduces the size of the search tree significantly, allowing searches up to twice as deep under the best conditions.

Another major enhancement to the minimax algorithm has been *iterative deepening*. In iterative deepening the search analysis is performed repeatedly to an increasingly deeper depth. During each new search the whole of the previous search tree is retraced, this time extending the search depth (usually by one). Although this idea seems counterintuitive, in practice it proves to be very effective, often requiring less time than an equivalent non-iterative search.

Iterative deepening is efficient if it is combined with *memory-assisted search*, such as the use of transposition tables. Re-visiting nodes in a game tree is very common since positions repeat themselves often (there may be multiple ways to reach a position). Transposition tables are used to store previously analyzed positions so that when they are encountered again they do not have to be re-searched.

An important characteristic of games is the average number of moves in a position (i.e. *the average branching factor*), since it affects the size of the search tree that has to be built. Traditional search methods, such as Alpha-

Beta, work well for games like chess and checkers that have a moderate average branching factor (35 and 8 respectively). However, when it comes to games like Go and Amazons (see Section 1.2), which have an average branching factor of several hundred, computer programs face a serious search challenge. The large number of possible moves makes deep searches impractical, thus rendering Alpha-Beta ineffective.

So, what can be done in this case? One solution comes in the form of selective search techniques.

> **Selective Search.** This comes in two flavours. First, instead of searching all the branches in the search tree, only nodes that look promising are expanded. Second, a non-uniform search depth is maintained, which means that promising nodes are searched deeper than others.

Selective search methods can be applied throughout the whole game. In practice, however, there is some risk associated with them since some moves that look bad at shallow search depths may turn out to be very good.

Another solution to the problem of shallow global search in Go and Amazons comes in the form of a mathematical theory, called Combinatorial Game Theory (CGT).

> **Combinatorial Game Theory.** This is a mathematical theory for playing a sum of games that have specific properties (see page 9). CGT analyzes each game separately and then combines the analysis to come up with the overall best playing strategy.

In some cases CGT has proven to be an efficient alternative to minimax methods, being able to analyze positions that would be intractable by even the most efficient minimax algorithm.

CGT can be applied to the endgame phase of Go and Amazons, since the board tends to decompose into independent areas. Each of these areas can then be considered a separate (sub)game. For relatively small areas, the results of the CGT analysis can be permanently stored in endgame databases, so that the analysis does not have to be performed at run-time.

> **Endgame Databases.** Endgame databases are permanent databases that store the solution for subsets of positions near the end of the game, when the exact final value can be calculated.

Endgame databases have been used with great success in games such as checkers and chess. The apotheosis of endgame databases applications has been their contribution to solving Nine Men's Morris [Gas95], the first non-trivial game solved using a combination of Alpha-Beta search and Retrograde Analysis (it is a draw with perfect play). Retrograde Analysis is the most popular technique for constructing endgame databases.

> **Retrograde Analysis.** This is a method used for the exhaustive search of a search space. It works backwards from the final states enumerating all the predecessor states and computing their final value, continuing like that in a bottom-up way to generate all the positions further and further from the end positions and closer to the start state.

Minimax methods provide the solution for the starting position together with positions along the best line of play (i.e. *the principal variation*), assuming that the search can extend to the end of the game, something which is infeasible in practice. The advantage of Retrograde Analysis is that the optimal solution is determined for all the positions that can be enumerated with the available computing resources, albeit for positions close to the end of the game.

Amazons is a game that appears as an excellent test bed for research in games. It provides the perfect context to explore what AI can contribute to games with a large branching factor. Go, which is considered by many the most intractable game and has received a lot of interest from the games community, may seem like the obvious choice, but it requires lots of application-dependent knowledge. This is hardly the case with Amazons, with its simple rules and elegant strategy. Moreover, Amazons is a nice intermediary between chess and Go. On one hand, it does not have the strategic complexity of chess but it has a larger branching factor. On the other hand, it does not exhibit the difficulty of Go although it poses similar challenges to games researchers because of the high branching factor.

1.2 Amazons

Amazons is a two-player deterministic board game. It belongs to the category of zero-sum games, in which whatever is gained by one player is lost by the other (i.e. it is impossible for both players to win or lose). It is also a game of perfect information since at any instance all the state information for the game is available to both players. The game is a battle for territory allowing the development of various playing strategies.

1.2.1 History

Amazons was invented by the Argentinian Walter Zamkauskas in 1988. He was probably inspired by Greek mythology, which refers to Amazons as a warlike tribe of women who lived somewhere in Asia Minor and were very adept in archery.

The rules of the game where first published in Spanish in the 4^{th} issue of the puzzle magazine El Acertijo in December 1992. In 1993, Amazons was introduced to the postal gaming club "kNights Of the Square Table" [NOS93] by Michael Keller and started gaining in popularity. One year later a translation of the original El Acertijo article was published by Michael Keller in World Game Review, an infrequently appearing games magazine [Kel94].

In the following years, Amazons gained the attention of the games community. The first international match was a six-game friendly match played by fax between a team from Argentina and one from the USA in 1994. The final score was a 3-3 tie.

In 1997, Amazons was introduced to Richard's Play-By-eMail Server [Rog96] by John Williams. A lot of games have since been played on the server, which is attracting players of different skill levels. The well-known games researcher Michael Buro has recently created a web server that allows human players to play Amazons on-line using a graphical Java interface [Bur98].

Recently, Amazons has been included in the program of the annually held Computer Olympiad. There have been two tournaments so far that have attracted the best Amazons programmers from all over the world, one in Computer Olympiad 2000 and the other in Computer Olympiad 2001.

1.2.2 Rules

Amazons is an elegant board game with very simple rules. It is played on a 10x10 board and each player has 4 Amazons queens. As can be seen on the left side of Figure 1.1, initially the white queens are placed on a4, d1, g1, j4 and the black queens are placed on a7, d10, g10, j7.

White moves first. Each move consists of two mandatory parts. First, an Amazons queen moves one or more squares horizontally, vertically or diagonally, just like a chess queen. A queen cannot jump over other queens on the board. After a queen is moved it has to shoot an arrow, which in turn moves like an Amazons queen. The square where an arrow lands is blocked for the rest of the game. This means that a queen or an arrow cannot move to or over that square. Moreover, an arrow cannot move to or over a square occupied by a queen of either color.

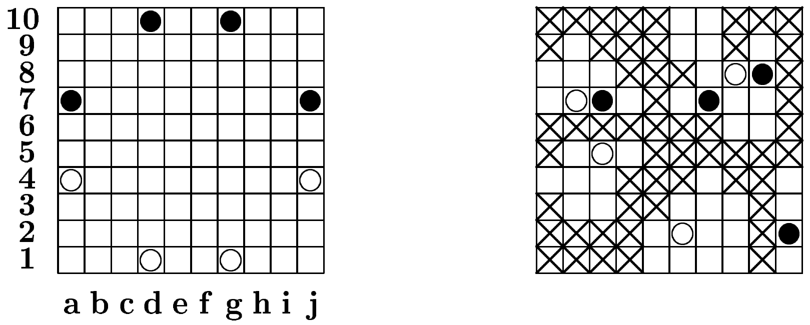


Figure 1.1: Starting Position (left) and Sample Endgame Position (right)

There is no capturing rule in Amazons. Since an arrow must be shot on each move, more and more squares get blocked off as play progresses. At some point, all the squares on the board will be occupied by either a queen or an arrow, so no more moves will be possible. The winner is the last player able to make a move.

1.3 Problem Description

Since the appearance of computer programs that play games in the 1950's, a lot of progress has been made in game playing. Until recently, research had concentrated mainly on chess, which was considered to be the most significant field of computer game playing. During the last decade many researchers have started looking into other games, sometimes with remarkable success. Most notable are the achievements made in Backgammon [Tes94], checkers [Sch97] and Othello [Bur97], where computer programs have equaled or even surpassed human play.

There is a category of games that do not exhibit the complex strategic elements of chess but pose a greater challenge for computers. These games characteristically have a large branching factor. Amazons is such a game. Playing strength has been found to be directly proportional to search depth for many games. A large branching factor limits the search depth thus weakening computer play. This makes Amazons a suitable test bed for existing selective search methods and new search ideas.

1.4 Objective

The objective of this thesis is to investigate issues in developing a high-performance Amazons program. The effort is concentrated on the endgame, focusing on endgame database creation and use.

Towards the end of an Amazons game, the board tends to be divided into independent areas. The right side of Figure 1.1 shows such a case. This decomposition into independent sub-games suggests the use of Combinatorial Game Theory in order to analyze a position. The combinatorial values of small sub-games with queens of both colors can be stored in permanent databases, allowing fast and almost perfect play towards the endgame.

Another important characteristic of endgames in Amazons is the creation of areas that contain only queens of the same color. These areas can be excluded from the minimax search since they are single-agent (i.e. one-player) problems. Special endgame databases can be created that allow perfect scoring of these areas.

Until now CGT has been almost exclusively the domain of the mathematics community. No one has moved this technology from the realm of theory into practice. The good of this thesis is the first investigation of adding CGT to a high-performance game-playing program.

1.5 Related Work

Amazons is a relatively new game and has only recently started attracting the interest of researchers in the games community. Michael Buro has been able to prove that Amazons endgames where each sub-game contains queens of the same color are NP-equivalent [Bur00b] (NP-equivalent problems have the same degree of difficulty as NP-complete problems in terms of polynomial time solvability). Many ideas about selective search in games have been proposed throughout the years. Multi-Probcut [Bur00a] is a pruning method based on shallow tree searches that was developed by Michael Buro and has been successfully used in his Amazons program called amsbot.

Endgame databases are widely used in games and have been especially successful in some of them. The use of databases allows programs to play optimally towards the end of the game, when there are a few pieces left on the board. In 1986, Ken Thompson pioneered Retrograde Analysis to build databases for chess [Tho86]. The same method has been widely used ever since. Martin Müller has incorporated an on-line combinatorial-endgame-database builder in his Go program (Explorer) and has developed a new search method called Decomposition Search [Mül99] to take advantage of them. Raymond Georg Snatzke has also undertaken the construction of combinatorial databases for Amazons as part of his Ph.D. thesis [Sna01].

Combinatorial Game Theory is a powerful mathematical theory that can be applied to certain games. The foundations of the theory were laid by John Conway in his book "On Numbers And Games" [Con76]. Several years later the theory was presented in "Winning Ways", which contained a gamut of games where CGT could be used [BCG82]. David Wolfe has built a useful toolkit that implements most of CGT and allows users to find the combinatorial values of complex game positions [Wol96]. Amazons started attracting the interest of researchers in CGT after Elwyn Berlekamp published an article about playing Amazons on rectangular boards with only two rows [Ber00].

1.6 Summary

The contributions of this thesis are as follows:

- **Antiope.** A strong Amazons program.
- **Endgame databases.** Antiope is the first program to make extensive use of large endgame databases in Amazons. There are various different types of databases that enable almost perfect play during the endgame.
- **New CGT results.** New positions with interesting combinatorial values were discovered during the construction of the CGT databases.
- **Novel way of storing Amazons positions.** A new, more space efficient representation of Amazons positions was developed (*Line Segment Graphs*). The new representation reduces the size of the databases significantly.

The thesis is organized as follows. Chapter 2 is an introduction to Combinatorial Game Theory, a mathematical theory for certain two-player games that can be applied to the game of Amazons. The general framework for constructing the endgame databases is presented in Chapter 3. The construction of endgame databases that store combinatorial game values is described in Chapter 4. Chapter 5 gives an analysis of *territories*, completely separated areas on an Amazons board that contain queens of only one color. Chapter 6 provides a description of Antiope, an Amazons program that implements the main ideas presented in this thesis. Various experimental results are presented in Chapter 7 and Chapter 8 contains the conclusions and future work.

Chapter 2

Combinatorial Game Theory

2.1 Introduction

When humans first see a position in a certain game, there are two questions that immediately come to mind:

- “What is the value of the position (i.e. which player is winning)?”
- “Which move is the best one to make next (i.e. how much is a move worth)?”

For some games (or subsets of a game), these questions can be answered by a mathematical theory called **Combinatorial Game Theory** (CGT). All these games share a common feature. They either start consisting of independent sub-games (Nim) or, as play progresses, they decompose into independent sub-games (Amazons, Go, Domineering, Konane). The sub-games are considered independent when a move in one of them does not affect the others.

This is illustrated in Figure 2.1, which shows an Amazons position consisting of three completely separated regions (sub-games). A move in one of the regions does not affect any of the other regions. CGT analyzes each sub-game separately trying to determine the overall best playing strategy.

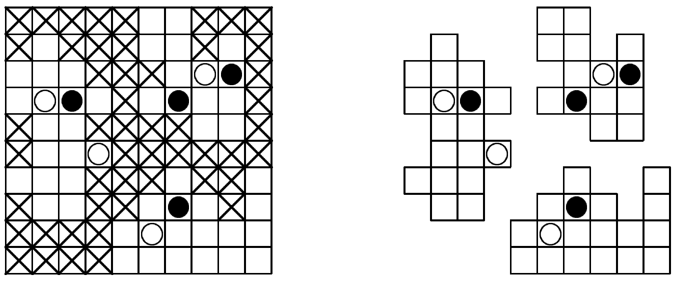


Figure 2.1: Sub-games in Amazons

The notion of sub-games can be generalized so that CGT can be applied when there are a number of different games to be played. For example, two players may be playing a game of Amazons together with a game of Go. Players move alternately and at each turn a player is allowed to move in one of the two games. The loser is the player who cannot make a move in any of the component games.

In general, the goal of CGT is to determine how to play the sum of a number of games [Con76, p. 74]. This is achieved by looking at each separate game independently and trying to combine this information in order to determine the outcome of the compound game.

2.2 Combinatorial Games

A game is considered to be combinatorial if it satisfies the following conditions [BCG82] :

1. There are just two players, often called Left and Right.
2. There are several, usually finitely many, **positions**, and often a particular **starting position**.
3. There are clearly defined **rules** that specify the **moves** that either player can make from a given position to its **options**.
4. Left and Right move alternately, in the game as a whole.
5. Both players know what is going on, i.e. there is **complete information**.
6. There are no **chance moves** such as rolling dice or shuffling cards.
7. In the **normal play** convention a player unable to move **loses**.
8. The rules are such that play will always come to an end because some player will be unable to move. This is called the **ending condition**. So there are no games which are drawn by repetition of moves.

The previous rules are general. CGT has been extended to cover games that violate some of these rules, but rules 5 and 6 must always hold. The original rules of Amazons satisfy all of the above rules without any modifications.

It should be pointed out that CGT deals primarily with finite games, which are games that have a finite number of positions [Con76]. However, the theory is much more general than that and has been extended to cover infinite games as well.

Games are classified as being **impartial** or **partizan**. **Impartial** games satisfy the condition that from any position both players have exactly the same moves. In **partizan** games, the two players may have different moves from a given position. Amazons is a partizan game.

According to CGT, each position in a game is assigned a value, which indicates which player (if any) is favoured by this particular position. A game G can be classified into one of four categories based on the final outcome:

Positive Left can always win no matter which player starts first ($G > 0$).

Negative Right can always win no matter which player starts first ($G < 0$).

Fuzzy The player who starts first can always win ($G \mid\mid 0$).

Zero The player who plays second can always win ($G = 0$).

Following CGT conventions, Left (or Black) tries to maximize the value of a game and Right (or White) tries to minimize it. So, the larger the value of a game, the happier Left is, while Right is contented with as small values as possible.

Consider a game G, where Left has n moves to positions $1, 2, \ldots, n$ and Right has m moves to positions $1, 2, \ldots, m$. The value of G can be written down as the following general expression:

$$G = \{G_1^L, G_2^L, G_3^L, \ldots, G_n^L \mid G_1^R, G_2^R, G_3^R, \ldots, G_m^R\}$$

$G_1^L, G_2^L, G_3^L, \ldots, G_n^L$ are the values of the positions that result after moves made by Left and they are called the **Left options**. $G_1^R, G_2^R, G_3^R, \ldots, G_m^R$ are the values of the positions that result after moves made by Right and they are called the **Right options**.

The Left and Right options are separated by a slash $|$. As a notational shortcut, curly brackets can be omitted and multiple slashes $||$ can also be used as a stronger separator. For example:

$$\{\{a \mid b\} \mid c\} = a \mid b \mid\mid c$$

Figure 2.2 shows a simple Amazons position. If Black starts first it has only one available move to a position where White owns the last empty square, so the resulting position has a value of -1. If White starts first the resulting position will have a value of 1, with Black getting the last empty square. So, the position is equal to $\{-1 \mid 1\}$.

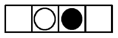


Figure 2.2: An Amazons position with CGT value $\{-1 \mid 1\}$

Let G be a game such that:

$$G = \{G_1^L, G_2^L, G_3^L, \ldots, G_n^L \mid G_1^R, G_2^R, G_3^R, \ldots, G_m^R\}$$

The **negative** of G is produced recursively by interchanging the moves for Left and Right:

$$-G = \{-G_1^R, -G_2^R, -G_3^R, \ldots, -G_m^R \mid -G_1^L, -G_2^L, -G_3^L, \ldots, -G_n^L\}$$

For example:

$$G = \{\{1 \mid 2\}, 1 \mid \{3 \mid -1\}, \{-2 \mid 0\}\}$$

$$-G = \{\{1 \mid -3\}, \{0 \mid 2\} \mid \{-2 \mid -1\}, -1\}$$

The sum of a game plus its negative is always a **zero** game, since the second player can win by copying the first player's moves.

The values of the simplest games are numbers. In this context, "numbers" stands for integers and dyadic rationals ($\frac{m}{2^n}$ m, n: integers), since these are the only numbers that appear in finite, loop-free games. All the numbers can be obtained from the following rules:

$$\begin{aligned} 0 &= \{\ \mid\ \} \\ n+1 &= \{n \mid\ \} \\ -n-1 &= \{\ \mid -n\} \\ \tfrac{2p+1}{2^{q+1}} &= \{\tfrac{p}{2^q} \mid \tfrac{p+1}{2^q}\} \end{aligned}$$

For example:

$$\begin{aligned} 7 &= \{6 \mid\ \} & -36 &= \{\ \mid -35\} \\ \tfrac{1}{2} &= \{0 \mid 1\} & -\tfrac{5}{2} &= \{-3 \mid -2\} \\ \tfrac{11}{64} &= \{\tfrac{5}{32} \mid \tfrac{6}{32}\} & 3 &= \{2\tfrac{1}{2} \mid\ \} \end{aligned}$$

It is important to note that the condition $n \geq 0$ must be true in order for the above rules to work. Suppose there is a game G where Left has only one move to a position with a value of 1, but Right has no legal moves. So, $G = \{1 \mid\ \}$. Since 1 is larger than 0, this game corresponds to the second rule: $\{n \mid\ \} = n+1$. So, $G = 2$. However, if in the same game Left had a move to -2 instead of 1, the condition $n \geq 0$ would not hold, making the second rule inapplicable. In fact, $\{-2 \mid\ \}$ is a **zero** game. Right wins if Left moves first (value of -2) and Left wins if Right moves first, for Right cannot make any move in this game.

If the Left and Right options of a game are numbers then the game is a number only if all the Left options are less than all the Right options [Con76, p. 4]. For example, the game $\{0, 1 \mid 3, 5, 6\}$ is a number, but the game $\{2 \mid 0, 4\}$ is not. When the value of a game is a number, this number can be calculated according to the **Simplicity Rule**:

1. If there is any number that fits,
   the answer is the simplest number that fits.

2. A number fits if it is greater than all the Left options and
   less than all the Right options.

3. None of the number's options must fit.

The options of the number referred to by the Simplicity Rule are the ones that appear in the four rules that produce all the numbers. As an example, let's look at the game $G = \{1\frac{1}{4} \mid 2\}$. Could its value be $1\frac{129}{512}$? First of all, $1\frac{129}{512}$ is greater than $1\frac{1}{4}$ and smaller than 2, so it fits. The next step is to check if the options of $1\frac{129}{512}$ fit. By applying the fourth rule, it can be seen that $1\frac{129}{512} = \{1\frac{1}{4} \mid \frac{321}{256}\}$. It is obvious that the right option fits in G so the third condition of the Simplicity Rule fails.

Can G be equal to $1\frac{1}{2}$? $1\frac{1}{2}$ is greater than $1\frac{1}{4}$ and less than 2, so it fits. Also, it is the value of the game $\{1 \mid 2\}$, whose options (1 and 2) do not fit in G. Finally, by trial and error, it can be seen that $1\frac{1}{2}$ is the simplest number that fits. So, $G = 1\frac{1}{2}$.

Things get more complicated with games whose value is not a number. A subset of these games present some interesting properties and were given special names:

star $* = \{0 \mid 0\}$.

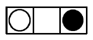
up $\uparrow = \{0 \mid *\}$.

down $\downarrow = \{* \mid 0\} = -\uparrow$.

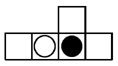
tiny-x $+_x = \{0 \mid \{0 \mid -x\}\}$.

miny-x $-_x = \{\{x \mid 0\} \mid 0\} = -+_x$.

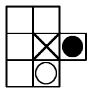
The previous values are called **infinitesimals**. Surprisingly, it can be proven that $*$ is less than all positive numbers, greater than all negative numbers, but **confused** with 0 [Con76, p. 100]. Moreover, it can be proven that $\uparrow$ and $+_x$ are strictly positive (Left can always win), but less than all positive numbers, whereas $\downarrow$ and $-_x$ are strictly negative (Right can always win), but greater than all negative numbers. Some Amazons positions whose value is an infinitesimal can be seen in Figure 2.3.



$G = *$



$G = \uparrow$



$G = +_2$

Figure 2.3: Infinitesimals in Amazons

Expressions of games can get very complicated. There are two straightforward ways of simplifying these expressions: **deleting dominated options** and **bypassing reversible moves**.

Let G be a game:

$$G = \{A, B, C, \ldots \mid D, E, F, \ldots\}$$

and $A \leq B$. Since Left tries to maximize the value of G, Left will always prefer to move to B rather than A. In other words, A is **dominated** by B. Similarly, Right will always prefer a smaller value to a larger one, so if $D \leq E$ it is said that E is **dominated** by D. All the dominated options can be deleted from a game. In this case G becomes:

$$G = \{B, C, \ldots \mid D, F, \ldots\}$$

For example, if $G = \{0, 1, 2 \mid 3, 4\}$ then the Left options $0, 1$ are dominated by 2 and the Right option 4 is dominated by 3. So, $G = \{2 \mid 3\} = 2\frac{1}{2}$.

A **reversible move** is a move whose effect can immediately be reversed by the opponent. Let G be a game:

$$G = \{A, B, C, \ldots \mid D, E, F, \ldots\}$$

Let's assume that Left moves to position A and, from A, Right has an option A^R, such that:

$$A^R = \{U, V, W, \ldots \mid X, Y, Z, \ldots\}$$

If $A^R \leq G$ then the move to A by Left is reversible since Right can get to a position at least as good as G. Left will move to A, Right will move to A^R and then Left will move to one of the $U, V, W, \ldots$ options. So, the move to A can be bypassed by replacing it with $U, V, W, \ldots$ After the substitution G becomes:

$$G = \{U, V, W, \ldots, B, C, \ldots \mid D, E, F, \ldots\}$$

The same principle can be applied for moves reversible by Left. If there is a position

$$D^L = \{K, L, M, \ldots \mid P, Q, R, \ldots\}$$

such that $D^L \geq G$, then D can be replaced by $P, Q, R, \ldots$ and we have:

$$G = \{A, B, C, \ldots \mid P, Q, R, \ldots, E, F, \ldots\}$$

Reversible moves can be difficult to analyze. The goal of bypassing reversible moves is to simplify a game G, but in order to check for reversible moves the value of G must be compared to other games. So, at first glance it might seem like begging the question. However, the value of a game can be compared to other games even if we do not know its simplest form [BCG82, p. 64].

There is one important difference between a game tree constructed using traditional minimax search and a game tree constructed following CGT rules. The former tree contains branches that are formed only by alternating moves made by the two players. The latter tree, however, contains branches with sequences of moves made by the same player, since the opponent is allowed to play in a different (sub)game.

A CGT game tree is presented in Figure 2.4. In any position, all the moves by Left are shown with an arrow pointing to the left and all the moves by Right are shown with an arrow pointing to the right. It can be seen that some branches in the tree contain successive moves made by the same player. Not all of the branches have the same length. This is because some positions decompose into independent sub-games that are numbers, so it is sufficient to just tally the numbers and the final number indicates who the winner is.

The root position is equivalent to the game:

$$G = \{-\frac{1}{2},\ 0,\ 0,\ 0 \mid *,\ -2,\ 2\}$$

Since Left is trying to increase the value of the game and Right to decrease it, the Left options are dominated by 0 and the Right options by -2. So G can be simplified to:

$$G = \{0 \mid -2\}$$

which is actually a fuzzy game since whoever moves first can win.

2.3 Nimbers

Nim is an impartial game. It is played with a number of heaps each one of which contains a number of tokens, as seen in Figure 2.5. A move consists of removing a number of tokens from a single heap. The player who can no longer remove a token loses the game.

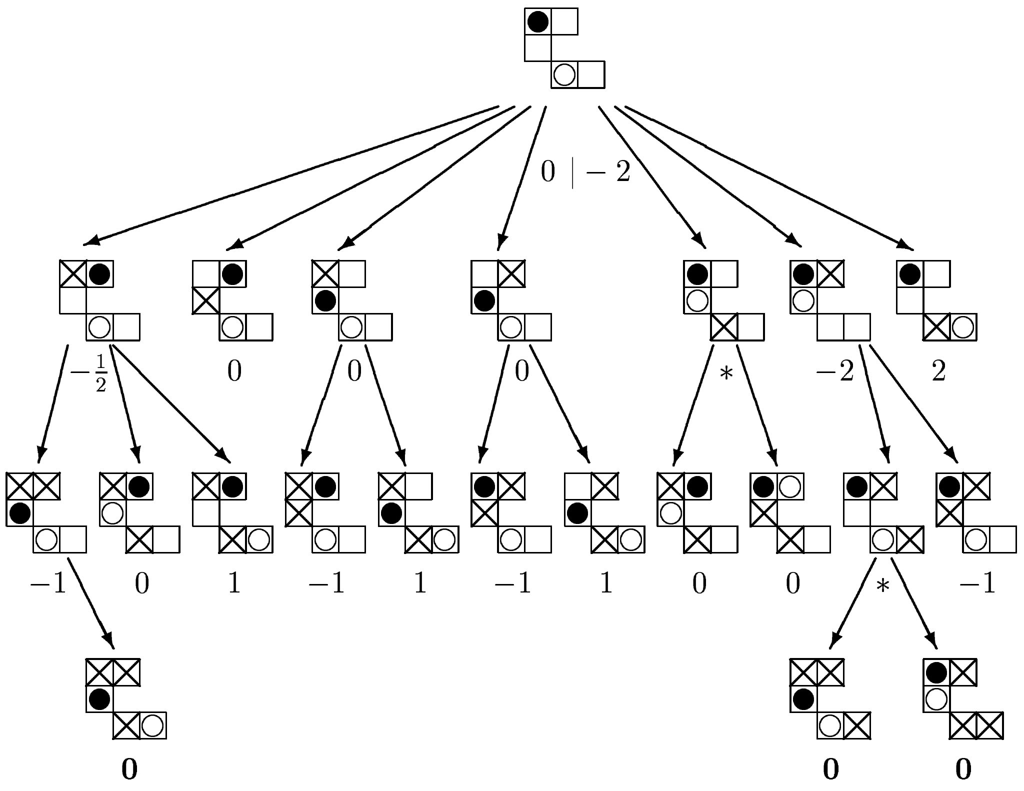


Figure 2.4: CGT game tree

Since in Nim both players have the same moves from any position, a Nim game is its own negative. As a result, the sum of two Nim heaps of equal size is equal to a **zero** game.

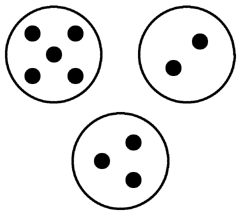


Figure 2.5: The game of Nim

The value of a Nim heap is called a **nimber**. In particular, a nimber $*n$ corresponds to a Nim heap with n tokens and is calculated according to the following formulas:

$$
\begin{aligned}
*n &= \{*0, *1, *2, \ldots, *(n-1) \mid *0, *1, *2, \ldots, *(n-1)\} \\
*0 &= \{\,\mid\,\} = 0 \\
*1 &= \{*0 \mid *0\} = \{0 \mid 0\} = *
\end{aligned}
$$

It turns out that the sum of two nimbers is another nimber. Since Nim is the sum of a number of heaps and the value of each heap is a nimber, it follows that the value of every Nim game is a nimber.

The game of Nim is very important because it plays a central role in the theory of impartial games. It can be proven that every finite, loop-free, impartial game is equivalent to a Nim heap [BCG82, p. 56]. So Nim encloses the theory of impartial games.

2.4 Thermography

When a game consists of many sub-games, it may be difficult for a player to choose the right sub-game to move in. Knowing how much moving in a sub-game is worth helps significantly.

If a sub-game is a number, say $G = \{1 \mid 2\} = 1\frac{1}{2}$, both players would like to avoid moving in it, since it would only worsen their position. Left can move to 1 therefore giving up half a move and the same holds for Right by moving to 2. This is true for every sub-game whose value is a number. This leads to the following simple principle:

Never move in a sub-game that evaluates to a number unless there is nothing else to do.

Another simple case is when a sub-game G is a **switch**:

$$G = \{x \mid y\} \quad x \geq y, \quad x, y : \textit{numbers}$$

In this case we can define the **temperature** t of G, which measures the value of moving in G, as:

$$t = \frac{1}{2}(x - y)$$

If all the sub-games are switches, then a player should move in the sub-game with the highest temperature.

Games in which both players are eager to move are called **hot** games. Switches are examples of hot games. On the other hand, numbers are examples of **cold** games because neither player is willing to move in them. Another example of cold games are **zugzwangs**, where both players are unwilling to move since it will only worsen their position. Zugzwang positions can occur in Amazons and playing them out well is not trivial [MT01].

Moving in numbers is bad for both players, since any move by Left will decrease the value of the game and any move by Right will increase it. So, a game can stop when all sub-games become numbers and the players can just add all the numbers to find out who the winner is. The positions of a game G that are equivalent to numbers are called the **stopping positions** of G.

The **Left stop** is the stopping position reached if Left starts first and both players move alternately in the same sub-game. The **Right stop** is defined similarly if Right starts first. Let's take the game $G = \{\{2 \mid 1\} \mid 3, 5\}$ as an example. Left can move to $\{2 \mid 1\}$ from which Right can then move to 1. So the Left stop of G is 1. Right can move to 3 or 5, but an intelligent Right player will choose 3 since it is smaller. So the Right stop of G is 3.

If a hot game is not a switch its temperature cannot be calculated trivially. In this case the temperature is calculated by repeatedly **cooling** the game.

Cooling involves imposing a tax t on each move and is denoted by G_t (G cooled by t). Since positive values are favourable to Left, the tax on Left's moves is imposed by subtracting t from each position, whereas for Right the tax is added to each position. So, $G_t = \{G_t^L - t \mid G_t^R + t\}$. Cooling stops when the value of the game is infinitesimally close to a number. This number is called the **mean** of the game and the corresponding tax is equal to the **temperature** of the game.

It can be shown that if m is the mean value and t the temperature of a game G then:

$$m - t < G < m + t$$

So, the values that G can take are centered around m, which is why m is called the mean of G.

When a game is complex, the temperature and the mean value can be found with the help of a graph, which is called a **thermograph**. A thermograph plots the Left and Right stops of the cooled game for all temperatures. Although the temperature is the independent value, it is traditionally plotted on the vertical axis and the dependent values are plotted on the horizontal one. Moreover, increasing positive values are traditionally plotted to the left of the horizontal axis, while negative values are plotted to the right. Figure 2.6 shows the thermograph for the game $\{2 \mid -1\}$, which has a mean value of $\frac{1}{2}$ and a temperature of $\frac{3}{2}$.

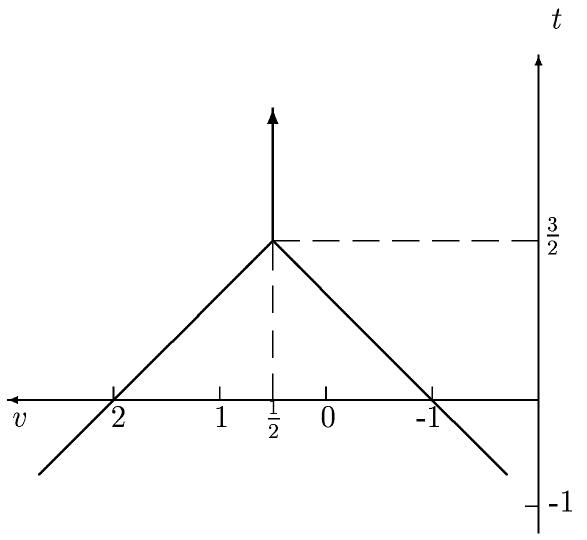


Figure 2.6: Thermograph for $\{2 \mid -1\}$

If $G = \{G^L \mid G^R\}$ is a game then its thermograph is constructed with the following procedure:

1. Construct the thermographs for G^L and G^R. The thermograph of a number n is a vertical line crossing the horizontal axis at $v = n$. In 1982, Elwyn Berlekamp introduced the notion of sub-zero thermography [Ber96], according to which, a sub-game whose value is a number has a negative temperature. Rational sub-games ($G = \frac{m}{2^n}$) have a temperature of $-\frac{1}{2^n}$, while integer sub-games have the lowest possible temperature (-1).

2. Impose the tax on the Left and Right options. This is done by rotating the Left thermograph clockwise by 45° and the Right thermograph counter-clockwise by 45°.

3. The two thermographs meet at a point whose x coordinate is the mean of the game and whose y coordinate is the temperature of the game.

4. The part of the thermograph above the meeting point is just a vertical line called the **mast**.

Figure 2.7 shows how the thermograph for the game $\{2 \mid -1\}$ is constructed from the thermographs of the Left and the Right options. It is important to note that the thermograph of a sum of games cannot always be determined just by knowing the thermograph of each game [BCG82, p. 163].

Thermographs model the trade-off between making a move locally and playing first in some other sub-game. The temperature is the point where a player is indifferent between playing locally and playing elsewhere.

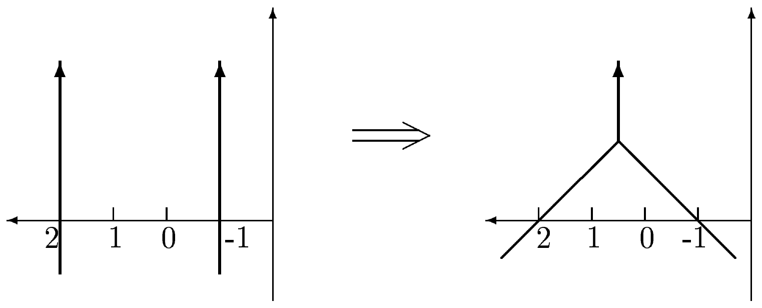


Figure 2.7: Construction of the thermograph for $\{2 \mid -1\}$

Thermographs are particularly useful when a player has to make a move in one of several hot sub-games that are not switches. CGT offers three strategies to handle this situation. All of them are based on thermography and are heuristic in nature. They do not guarantee optimal play but two of them guarantee a bounded error.

2.4.1 HotStrat

HotStrat is the simplest strategy. Whenever a move has to be made, HotStrat will choose the hottest sub-game and play the move which is best at the largest temperature.

2.4.2 SenteStrat

SenteStrat is a more sophisticated strategy [Ber96]. The basic idea is to decide whether to keep *sente* (i.e. the initiative to move in any sub-game) or play in the same sub-game as the opponent did. For this purpose, SenteStrat keeps track of a temperature called the **ambient**. The ambient never increases. An outline of the strategy is as follows:

1. If the opponent's last move raised the temperature of a sub-game G_i above the ambient, then reply with a move in sub-game G_i that is best at the ambient temperature.
2. Otherwise
   (a) find the largest temperature t of all sub-games
   (b) if t is less than the ambient make the ambient equal to t
   (c) play in the sub-game with temperature t, selecting the best move at the ambient temperature.

2.4.3 ThermoStrat

ThermoStrat is the most complex strategy. Like SenteStrat, it also keeps track of the ambient temperature. Every time a move has to be made, ThermoStrat will choose the best move at the ambient temperature at a sub-game which has the widest thermograph at the ambient temperature.

All three strategies can fail to select the optimal move [Ber96, p. 391]. However, SenteStrat and ThermoStrat guarantee a bounded error whereas HotStrat does not. In particular, they guarantee that the error for the whole sum game will be at most equal to the ambient.

Chapter 3

Endgame Databases

Towards the end of an Amazons game, the board usually decomposes into independent areas. Each of these areas can be viewed as a separate Amazons sub-game. Sub-games that contain only queens of the same color offer an uncontested number of moves to a player and are called *territories.* Sub-games that contain queens of both colors are called *active areas.* Extensive endgame databases have been built for both types of these sub-games.

3.1 Empty Areas

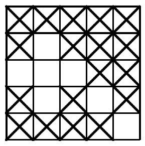


Figure 3.1: A 5x4 room of size 7

Before any databases can be computed, all the areas that contain only empty squares are built. These areas consist of a number of empty squares that are connected horizontally, vertically or diagonally (8-connected) and they are called **rooms**. The number of empty squares in a room is called the **size** of the room, while the **dimensions** of a room are equal to the dimensions of the smallest rectangle that contains the room. Figure 3.1 shows a 5x4 room of size 7.

The rooms are built using a variation of Retrograde Analysis [Tho86]. All the rooms with $n+1$ squares are built by adding an empty square to a room with n empty squares. Of course, all the empty squares must be connected in the resulting rooms.

The relative position of a room on the board is irrelevant. Only the shape of a room is important as can be seen in Figure 3.2. Symmetries caused by translation of the rooms are removed by encoding only the rectangle that encompasses a room rather than the whole board. Despite that, the databases built using the method previously described contain a lot of redundancy, since many rooms are symmetrical to one another (i.e. identical after rotation or reflection). For example, in Figure 3.3 it is clear that half of the rooms of size 2 are redundant.

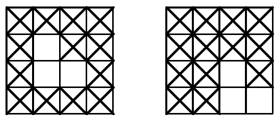


Figure 3.2: Two rooms that are considered identical

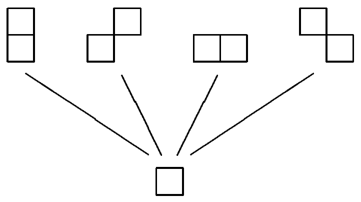


Figure 3.3: Construction of all rooms of size 2

A number of symmetry checks are performed on the generated rooms to remove redundancy. Figure 3.4 shows the 7 symmetry checks, which consist of 4 reflection and 3 rotation checks. All the rooms of size 2 and 3 after symmetries have been removed are shown in Figure 3.5. Table 3.1 contains the number of all different rooms with dimensions 4x5 or smaller.

3.2 Types of Databases

Four different types of databases have been constructed. The first one is the *territory* databases. The other three are all *active areas* databases and differ in the type of value they store. They are namely the *minimax*, *combinatorial* and *thermographic* databases.

All these databases are constructed with the help of the *room* databases. Territories are created by taking an empty room and placing one or more same-colored queens in it, in every possible combination. Active areas are created by taking an empty room and placing queens of different colors in it, in all possible combinations. In general, the databases are constructed by doing a 1-ply search. In particular, all possible queen moves are played out and the resulting positions are looked up in the smaller databases.

The size of the various databases that have been built differs because of resource limitations for both storage (i.e. memory and disk) and computation time. The following is an overview for each database.

- **Territories.** All the territories up to size 4x6 with 1, 2, 3 and 4 queens in them.
- **Minimax.** All the sub-games up to size 4x6 with 1 queen of each color.
- **Combinatorial.** All the sub-games up to size 4x4 with 1 queen of each color; sub-games up to size 4x5 and 4x6 with 10 empty squares and 1 queen of each color.

- **Thermographic.** All the sub-games up to size 4x5 with 1 queen of each color.

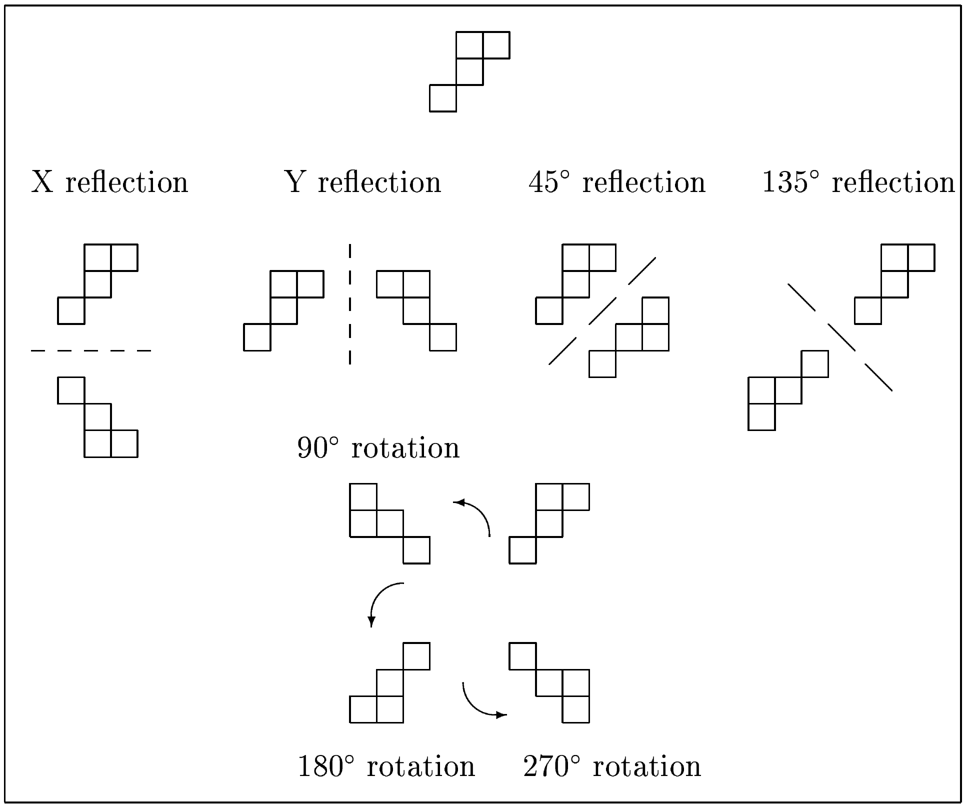


Figure 3.4: Symmetries

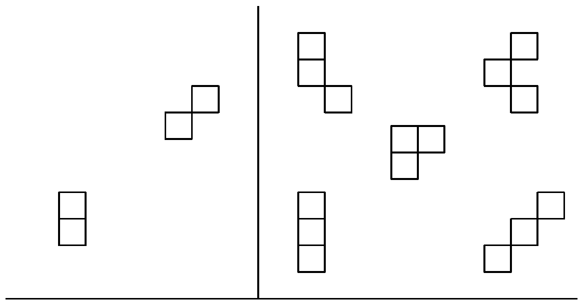


Figure 3.5: All rooms of size 2 and 3

Size	Rooms
1	1
2	2
3	5
4	22
5	93
6	439
7	1725
8	5205
9	11473
10	18904
11	23364
12	22070
13	15873
14	8841
15	3740
16	1233
17	293
18	56
19	6
20	1

Table 3.1: All 4x5 rooms

3.3 Storage and Retrieval

Each room is represented by a unique index consisting of 64 bits. The rightmost 56 bits encode the shape of a room as a rectangular bitmap. The width and the length of a room are encoded in the remaining 8 bits, taking up 4 bits each. This encoding limits the dimensions and the size of the rooms that can be stored in the databases. However, it can easily be extended by allocating more bits for the index.

Each room index is stored once in the databases. For each one there is a list of all the possible queen combinations in that room. For each entry in that list, depending on the type of the databases, there are one or more bytes storing the position of each queen and respective storage for the value stored in the databases.

The rooms are stored sorted in the databases according to their index number. The lookup of a room is performed using binary search. In order to find a particular combination of queens within a room, a binary or linear search is performed, depending on the type of the databases.

Chapter 4

Active Area Databases

4.1 Introduction

Amazons is a combinatorial game. It can be viewed as a sum of independent sub-games. This decomposition becomes apparent towards the end of a game when separate areas are formed by arrows on the board. These areas are independent since a move in one of them does not affect the other areas.

At this point, Combinatorial Game Theory can step in and provide an analysis of the game. Each local sub-game is analyzed separately and then the results are combined in order to achieve optimal or heuristic global play. Instead of doing the analysis of an endgame position anew each time it is encountered in a game, the values of small endgame positions can be calculated once and stored in permanent endgame databases.

The use of endgame databases has been widespread in game-playing programs and a similar idea can be applied to Amazons. Traditional endgame databases store the minimax value of a position. This can be done in Amazons too. However, doing a full minimax search on an Amazons board that consists of several sub-games is possible only when these sub-games are small enough, since global search does not take into account the decomposition and considers the board as a whole.

On the other hand, CGT can tackle a much larger number of endgame positions by doing a local analysis of each sub-game, just like pattern databases [CS98]. This enables larger sub-games to be analyzed, which can then be combined to give an endgame position of a much larger size. This is done by looking up the combinatorial value of each local sub-game in the databases and then combining those values to determine the global play. In most endgame situations CGT can provide an analysis faster than minimax approaches since there is no global search involved.

4.2 Related Work

The databases that have been constructed are pattern databases. They enumerate all subgoals required by a solution, with certain constraints on the subgoal size. The subgoals in Amazons are the sub-games formed towards the endgame. Pattern databases have been used before with great success in single-agent problems, such as the 15-puzzle [CS98].

There are many games where programs use endgame databases. Chess-playing programs have been using databases with few pieces on the board for years. Checkers is another game where endgame databases have been used extensively. Chinook, a checkers program that uses databases with 8 or fewer pieces, won the World Champion title in 1994 [Sch97], becoming the first program ever to achieve this in a non-trivial game.

Endgame databases have even enabled programs to solve games. Ralph Gasser used endgame databases together with Alpha-Beta search and was able to prove that the game of Nine Men's Morris is a draw [Gas95]. Awari is another game where endgame databases have proven very important. The game has not been solved yet but large endgame databases have already been built and it is just a matter of time until it is solved [vdG00].

4.3 Constructing The Databases

Three different types of databases have been constructed. They all store sub-games with 1 queen of each color. Their difference lies in the kind of values stored in them.

One kind of database stores the *minimax* score for each sub-game. The other two databases store CGT related values. One database stores the *combinatorial* value of a sub-game in canonical form. The other database stores the corresponding *thermograph* of a sub-game.

As mentioned in Chapter 3, all the above databases are constructed with the help of the room databases. A sub-game with opposing queens is created by taking an empty room and placing queens in it in all possible combinations.

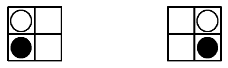


Figure 4.1: Symmetrical positions with two opposing queens

The databases are constructed by doing a 1-ply search. In particular, all the moves available to both players are played out and the value of the resulting positions is retrieved from the smaller databases. Since a move may result in a position which contains territories (i.e. sub-games with queens of one color), the territory databases must already be available and are also used.

Symmetries that appear after queens are placed in the rooms are not removed from the databases. For example, the two positions shown in Figure 4.1 are both present in the databases although they are symmetric.

4.3.1 Minimax Databases

The minimax databases store the minimax value for each player in a certain position. Currently, the minimax values for all the rooms up to size 4x6 have been calculated. Figure 4.2 shows a position from the databases with the corresponding minimax values for White and Black to move.

4.3.2 Combinatorial Databases

The combinatorial databases store combinatorial game theoretic values. When more than one sub-game is created after making a move, the combinatorial values of these sub-games must be added according to CGT rules. Because of this, the databases are calculated using David Wolfe's 'Gamesman's Toolkit' [Wol96], which provides various functions for computing combinatorial game values.

The combinatorial databases store three different values. First, they store the combinatorial value of a game in canonical form. This value is stored as a string and can be used for optimal play. It is also used during the construction of the databases in order to restore a game in the toolkit and later on for verification and data mining purposes.

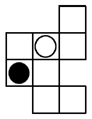


Figure 4.2: Minimax Score: Black = 1, White = 1

The other two values stored in the databases are the *mean* and the *temperature* of a sub-game. These values can be used to determine overall play according to CGT rules. The *mean* gives the average value of a sub-game and this value can be used in the evaluation function. The *temperature* can be used by various thermographic strategies to determine which sub-game to play in (see Chapter 2).

Currently, the combinatorial databases contain all the rooms up to size 4x4 and all the rooms up to sizes 4x5 and 4x6 that contain up to 10 empty squares. The reason the 4x4 databases are preferred (instead of an alternate size, e.g. 3x5) is because rooms of square shape seem to be more useful in an actual game. Figure 4.3 shows a position from the databases with the corresponding combinatorial value in canonical form.

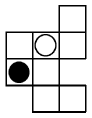


Figure 4.3: CGT Value: $0, \{1 \mid 0\} \mid -\frac{1}{2}$

4.3.3 Thermographic Databases

The thermographic databases explicitly store the thermograph of a sub-game. For this purpose, a toolkit that has been developed by Martin Müller for his work in Go has been used.

The thermographs are stored in their geometric form as a sequence of coordinates. CGT values, such as the mean and the temperature, are not explicitly stored. They can be calculated easily from the representation of the thermograph.

The whole thermograph must be stored because it provides much more information for CGT methods. For example, thermographic strategies need the left and right score at various temperatures on a thermograph, so storing only the temperature at the mast does not provide enough information.

Currently, the combinatorial databases contain all the rooms up to size 4x5. Figure 4.4 shows a position from the databases with the corresponding thermograph.

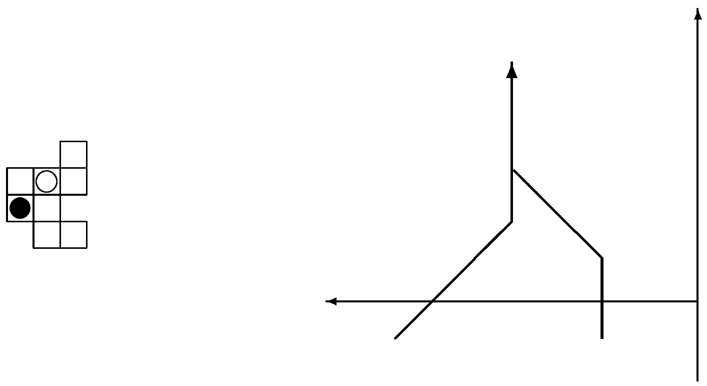


Figure 4.4: Thermograph for $0, \{1 \mid 0\} \mid -\frac{1}{2}$

4.4 Statistics

All the different kinds of databases contain the same number of positions. Only the values stored in the databases change. Table 4.1 shows the corresponding number of positions for all the rooms up to size 4x5.

The thermographs require a lot of memory to store and this makes calculating larger databases difficult. However, it turns out that many positions have the same thermograph, something that enables good compression of the thermographic databases. Table 4.2 shows the relevant figures for all the rooms up to size 4x5.

4.5 Limitations of the Databases

The construction of the endgame databases is not problem-free. There are currently two limitations. One is the size of the sub-games and the other is the number of queens contained in the sub-games.

Considering enough computer resources, there is no theoretical problem in constructing larger minimax databases with more than 1 queen of each color. Problems arise when the CGT databases, the combinatorial and the thermographic one, are considered.

The main problem with the combinatorial databases has to do with memory. The toolkit used to calculate the combinatorial values handles memory itself, so there is a limit to what kind of memory enhancements can be done

Size	Total Positions
2	2
3	15
4	132
5	930
6	6585
7	36225
8	145740
9	413028
10	850680
11	1285020
12	1456620
13	1238094
14	804531
15	392700
16	147960
17	39848
18	8568
19	1026
20	190

Table 4.1: Number of positions in rooms up to size 4x5

on the program that builds the databases. This is the reason why the 4x5 and 4x6 databases have not been completed yet. If memory was not an issue then these databases and even larger ones could be constructed.

The main problem with the thermographic databases involves the number of queens that can be placed in a sub-game. The databases with n ($n \leq 4$) queens of the same color against 1 opponent queen could be constructed given the necessary computer resources. However, in the case of n queens against more than 1 opponent queen, the thermographic databases cannot be built using the current framework.

The reason is that two thermographs corresponding to active areas cannot usually be added together. So, the thermographs of the sub-games will not be possible to add, which means that the 1-ply lookahead into the smaller databases will not work. The program that builds the databases has to be re-designed so that it performs a search until the end of the local sub-game (even if it is split into two or more active areas), when there are no other moves left for one of the players, and then back the calculated values up to the initial position in order to calculate the corresponding thermograph.

Size	Total TGs	Unique TGs	Reduction (%)
2	2	1	50.00
3	15	3	80.00
4	132	12	90.91
5	930	45	95.16
6	6585	181	97.25
7	36225	656	98.19
8	145740	2546	98.25
9	413028	9459	97.71
10	850680	27491	96.77
11	1285020	62141	95.16
12	1456620	111805	92.32
13	1238094	151896	87.73
14	804531	158503	80.30
15	392700	117586	70.06
16	147960	63201	57.29
17	39848	22030	44.71
18	8568	5244	38.80
19	1026	667	34.99
20	190	56	70.53

Table 4.2: Unique thermographs in rooms up to size 4x5

Chapter 5

Territory Databases

5.1 Territories

Areas on an Amazons board that only contain queens of a single color represent a number of free moves for that player. These areas are called **territories**. Territories can be viewed as single-agent puzzles where the goal is to determine how many squares a player can get.

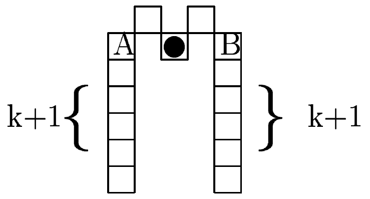


Figure 5.1: A k-defective territory

Usually, a territory can be filled completely allowing the player to get all the squares. However, there are cases where the number of moves that a player can make in a territory is less than the number of empty squares. In that case the territory is called **defective** [MT01]:

> A *k-defective territory* provides k less moves than the number of empty squares in the territory. A k-defective territory is said to have *k defects*.

Figure 5.1 shows that there is a k-defective territory for every k. The best that the queen can do in this situation is move to the left or to the right and shoot an arrow at A or B respectively. Either way an area containing k empty squares will be blocked off and lost. A territory can be defective even if it contains many Amazons as can be seen in Figure 5.2.

Michael Buro has shown that determining whether a territory can be completely filled or not is an NP-equivalent problem [Bur00b]. However, for small territories that fit on a 10x10 Amazons board it is possible to do an exhaustive search and build extensive endgame databases.

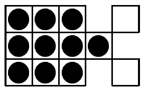


Figure 5.2: A defective territory with many amazons

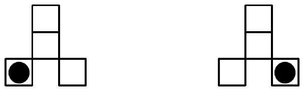


Figure 5.3: Symmetries in territories

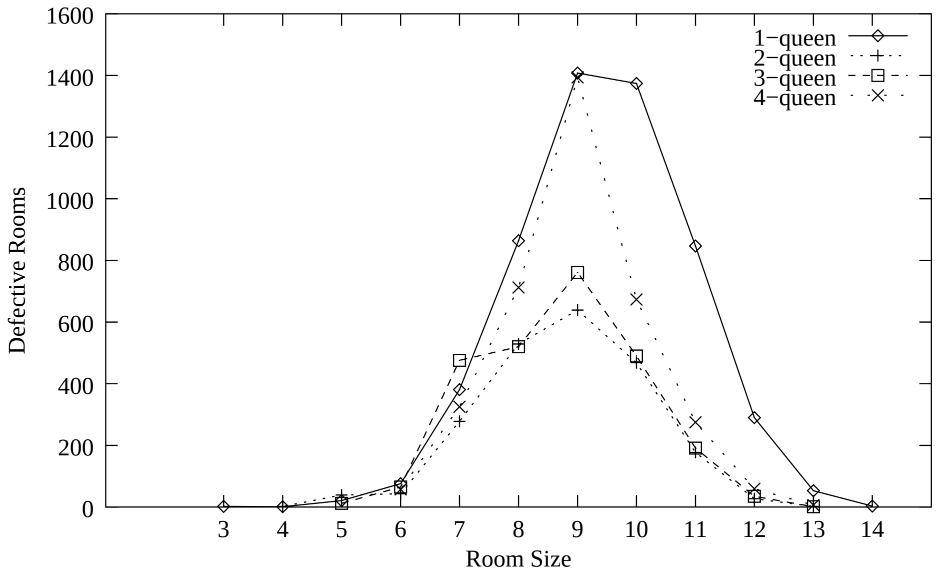


Figure 5.4: 4x5 defective rooms

Such databases have been built using a variation of Retrograde Analysis. As described in Chapter 3, starting with an empty room taken from the room databases, one or more queens are placed in it in every possible combination, creating a territory. Then a 1-ply search is performed, in which all the possible queen moves are played out and the resulting territories are looked up in the smaller territory databases giving a perfect score. The maximum score returned by all the moves, increased by one, represents the score of the initial position.

All the territories of size up to 4x6, containing up to 4 queens, have been constructed so far. Symmetries like the one seen in Figure 5.3 are removed by performing the same 7 symmetry checks as the ones applied to the construction of the room databases.

Defective territories offer a great way to reduce the size of the territory databases. Instead of storing all the territories, the databases store only the defective ones. If a search in the databases returns a hit then the territory is known to be defective and the exact number of obtainable squares is retrieved from the databases. Otherwise, if it is not found in the databases, a territory is assumed to be non-defective.

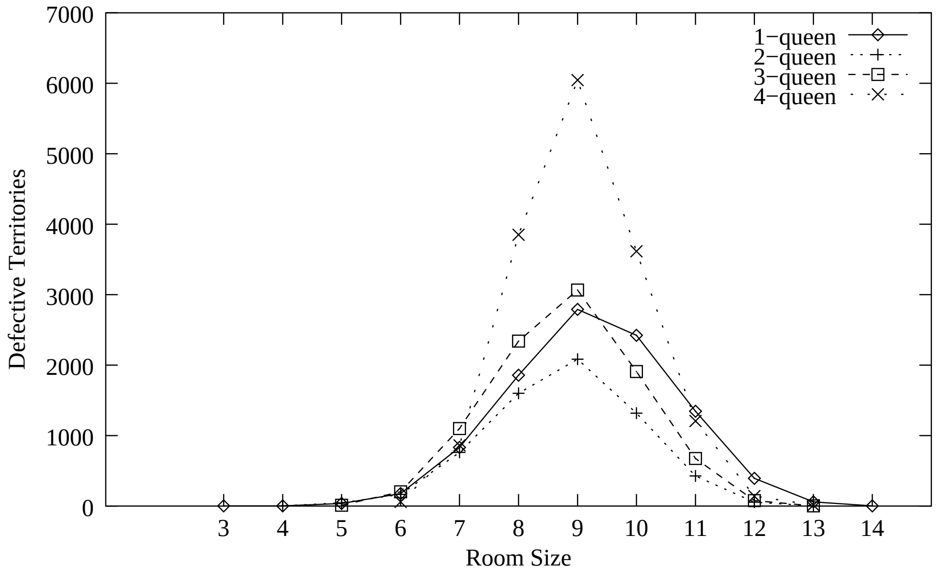


Figure 5.5: 4x5 defective territories

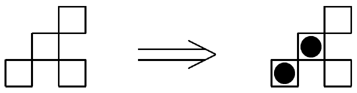


Figure 5.6: A room that can be defective with 2 but not with 1 queen

A very interesting characteristic can be seen in Figure 5.4 and Figure 5.5. Rooms are empty areas, while territories are rooms with same-colored queens. The corresponding numbers for the figures can be seen in Appendix A. Usually, it is assumed that more queens in a territory are more powerful, meaning that a larger part of the territory can be filled. However, this is not always true. There are cases where more queens in a territory are harmful. This is the case when some of the queens effectively act as arrows and usually block other queens, thus reducing the territory to a defective one with less squares. For example, Figure 5.6 shows a room that can never produce a defective territory with only one queen. However, when two queens are placed in it, a 1-defective territory may be created.

This is very important for both human and computer players to keep in mind, especially in the endgame. Figure 5.7 shows such a situation. This territory contains two queens and is not defective. However, the only way to get all the squares is by moving the queen at b3 to a4 and shooting an arrow at c2, thus blocking off the queen at c1 and creating two separate territories with one queen in each of them. Figure 5.9 shows various defective territories with up to 4 queens.

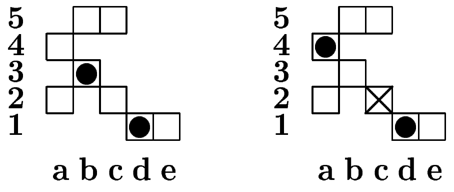


Figure 5.7: Two queens can be harmful

5.2 Line Segment Graphs

All the rooms and territories are stored in the databases as bitmaps. The symmetry checks performed on these bitmaps eliminate a lot of redundancy from the databases. Still, there are rooms that are not geometrically symmetrical but are strategically identical as far as the game is concerned (i.e. the move tree and decomposition behaviour are the same).

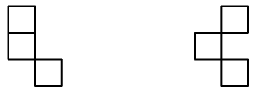


Figure 5.8: Two strategically identical rooms

Figure 5.8 shows two rooms that are not symmetrical but offer the same opportunities for playing them out no matter where any queens are placed in them. By looking at the figure it is clear that what the two rooms have in common is the number of straight lines formed by adjacent squares. The orientation of the lines is irrelevant. Only the number, the relative order of squares in each line, and their incidence relation matters.

Based on this observation, a different structure for representing rooms has been devised [MT01]. It is called a Line Segment Graph (LSG) and it is based on line segments created in a room by squares that are adjacent horizontally, vertically or diagonally.

Figure 5.10 shows two Amazons rooms and their corresponding LSG. The main difference between a LSG and an ordinary undirected graph is that a LSG stores information about line segments. In all the figures that depict LSG, all consecutive edges that have the same slope belong to the same line segment. In order to label the vertices, the squares in a room are numbered going from left to right and from bottom to top.

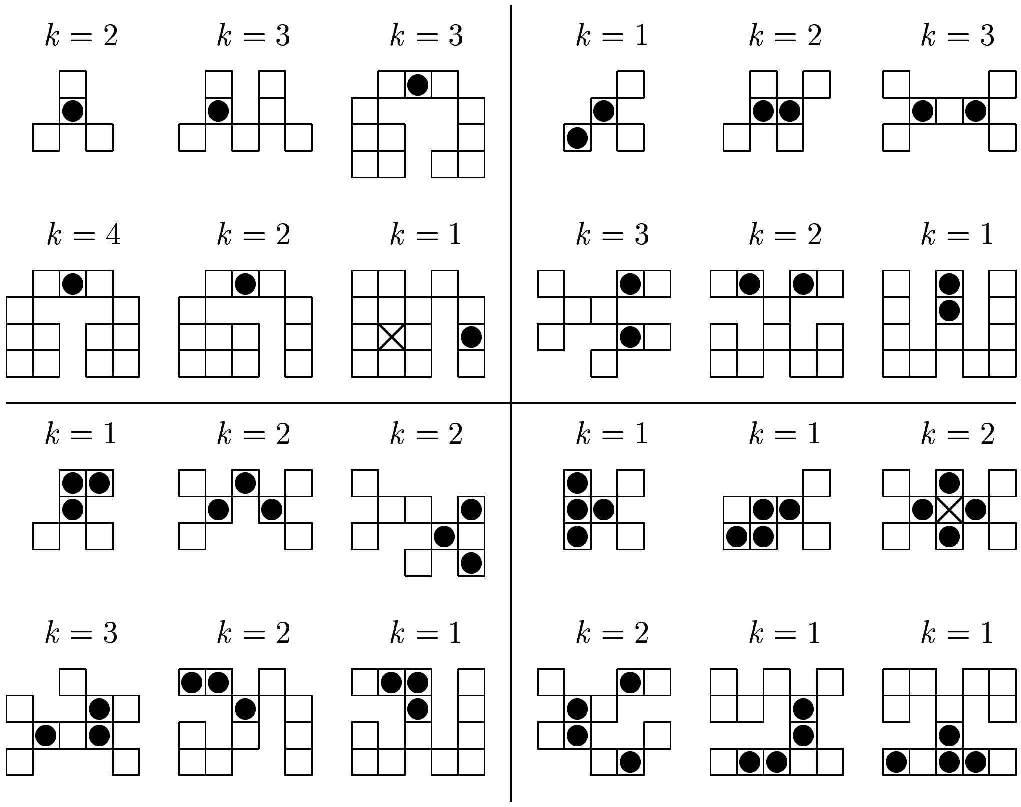
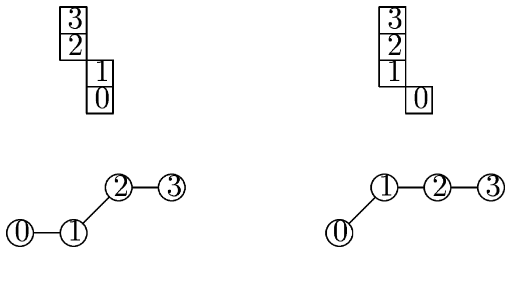


Figure 5.9: Various k-defective territories



3 lines: 0-1, 1-2, 2-3 2 lines: 0-1, 1-2-3

Figure 5.10: Two rooms and the corresponding LSG

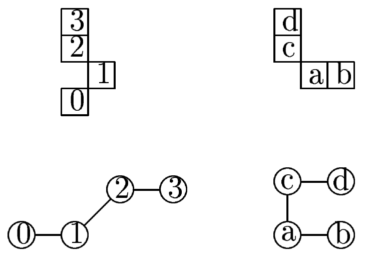
One important property of a LSG is that the points in a line form an ordered set, which means that the order of the points is important. LSG created from Amazons positions obey two more properties:

1. Every point is contained in at most four lines.

2. No line is longer than the size of the board.

5.2.1 Line Segment Graph Isomorphism

The advantage of storing rooms as LSG is that different rooms that are not symmetrical may have the same LSG representation, so there can be a reduction in the number of rooms stored in the databases. This reduction can be achieved by checking for isomorphic LSG. Figure 5.11 shows an example.



Mapping: $0 \Leftrightarrow b,\ 1 \Leftrightarrow a,\ 2 \Leftrightarrow c,\ 3 \Leftrightarrow d$

Figure 5.11: Two rooms with isomorphic LSG

The distinguishing characteristic of an LSG is the notion of a line. This leads to a definition of isomorphism between two LSG that is slightly different from the one pertaining to ordinary undirected graphs:

> A LSG G is isomorphic to a LSG H iff we can map (re-label) the vertices in G to those in H so that the lines in G are identical to the lines in H.

Checking for graph isomorphism is an NP-complete problem. Richard Krueger has come up with a polynomial time LSG analysis as part of his M.Sc. thesis but this approach has not been taken for the problem at hand. The reason is that for relatively small graphs, graph isomorphism can be done efficiently using existing algorithms. A very good program that can handle graph isomorphisms is *nauty*, written by Brendan McKay [McK01][McK81].

In order to check for isomorphisms, the LSG must be converted to a format suitable for *nauty*. For this reason, a somewhat larger graph, the *extended graph*, is built from an LSG according to the following algorithm [McK00]:

1. The extended graph consists of two parts. The “points” part and the “lines” part.

2. The “points” part is identical to the LSG.

3. The “lines” part contains an extra vertex for each line in the LSG.

4. Each vertex in the "lines" part is connected to the points (vertices in the LSG) in the line that it represents. For efficiency, only lines with 3 or more points need to be represented in the "lines" part since lines with fewer points are already contained in the LSG.

5. The "points" part must be distinguished from the "lines" part. In order to do that, all the vertices in the "points" part are assigned the same color and all the vertices in the "lines" part get another color. If there are any queens in the LSG then the color of the vertices that contain them is changed to the color of the queen.

An example of an extended LSG is shown in Figure 5.12. This figure shows two two-node lines at nodes 6 and 9. These nodes could be omitted. The reason that such an extended graph has to be constructed is to ensure two important properties of the LSG isomorphism. First, the "points" part ensures that isomorphisms can only preserve the relative order of all points on a line. On the other hand, the "lines" part ensures that any isomorphism between two LSG maps the lines of one to the lines of the other.

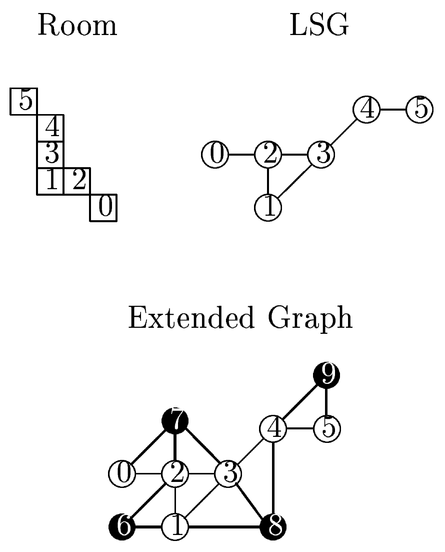


Figure 5.12: Extended LSG

Nauty does canonical labeling rather than isomorphism testing. So, the test for isomorphism consists of three conditions. First, two simple conditions are checked to save computation; that the two LSG have the same number of points and lines. Second, the canonically labeled extended graphs produced by nauty have to be identical.

Table 5.1 shows the numbers for all 4x5 rooms and their corresponding LSG. Figure 5.13 shows the ratio between the number of LSG and the number of bitmap rooms. Although initially there is a significant reduction, gradually it starts decreasing and after about 13 squares it becomes insignificant. This can be attributed to the small dimensions of the rooms. All the rooms are restricted to a 4x5 area which does not allow the presence of long and varied line segments.

Size	Bitmaps	LSG
1	1	1
2	2	1
3	5	3
4	22	11
5	93	42
6	439	191
7	1725	812
8	5205	3027
9	11473	8348
10	18904	16037
11	23364	21507
12	22070	21242
13	15873	15634
14	8841	8800
15	3740	3737
16	1233	1233
17	293	293
18	56	56
19	6	6
20	1	1

Table 5.1: 4x5 room bitmaps and LSG

The reduction is much more promising for rooms of larger dimensions. Table 5.2 shows the numbers for all rooms and LSG up to size 10 that fit on a 10x10 board and the product of their dimensions does not exceed 56. The corresponding ratio is depicted in Figure 5.14. Since the rooms are now more spread out, the reduction achieved by using the LSG representation appears fairly good.

Size	Bitmaps	LSG
1	1	1
2	2	1
3	5	3
4	22	11
5	94	42
6	524	199
7	3031	960
8	18769	4945
9	118069	25786
10	755047	137988

Table 5.2: 10x10 room bitmaps and LSG

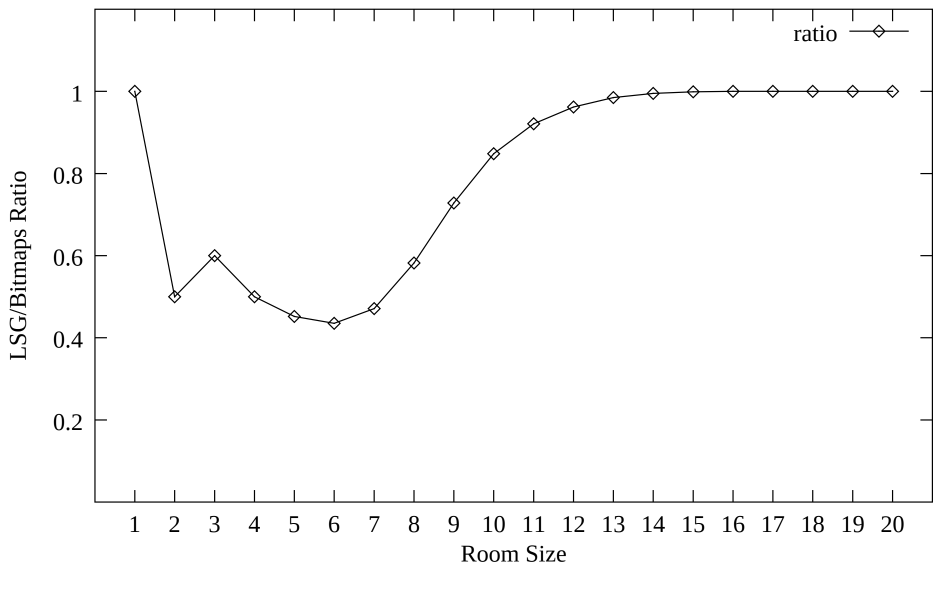


Figure 5.13: 4x5 LSG/Bitmaps ratio

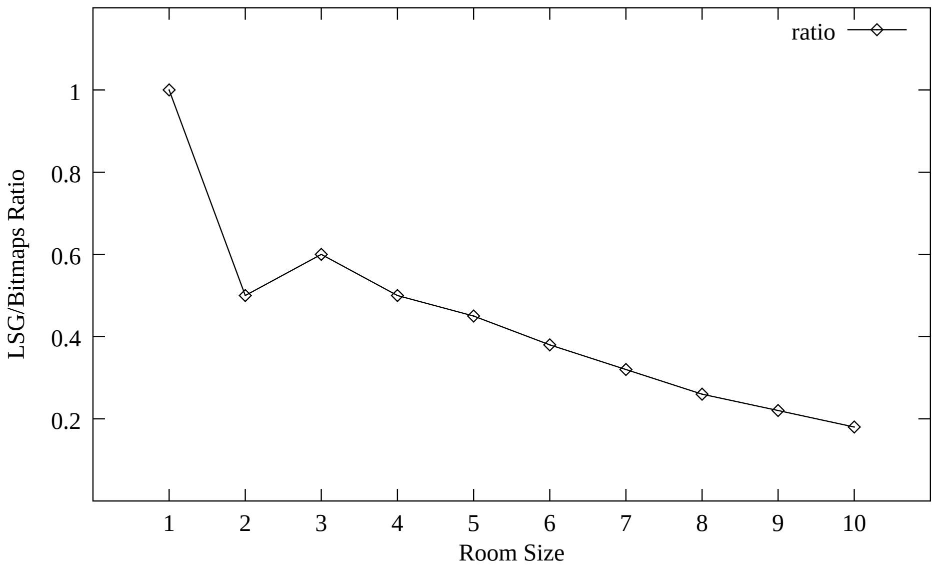


Figure 5.14: 10x10 LSG/Bitmaps ratio

Chapter 6

Antiope

Antiope is a state-of-the-art computer program that plays the game of Amazons. It has been developed over the course of two years. It provides a test bed for researching the ideas on evaluation functions, search techniques and endgame databases presented in this thesis.

6.1 Search Algorithm

The most widely used search algorithm for two-player perfect-information games (such as Amazons) is the *minimax* algorithm. It builds a tree with all the combinations of moves and replies, extending the branches up to a fixed number of moves in the future. Each level in the tree represents the moves of a player and players alternate at consecutive levels. One player is supposed to try to maximize the score of the game and is called the maximizing player, while the opponent is the minimizing player and tries to minimize the score of the game.

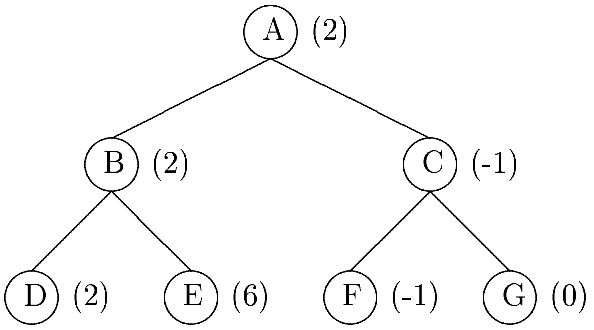


Figure 6.1: A Minimax Tree

The minimax algorithm builds the search tree by expanding all the branches to a fixed depth. All the terminal positions (i.e. leaf nodes) are evaluated using the static evaluation function and the values are then propagated, through maximizing and minimizing levels, up to the root node of the search tree. Figure 6.1 shows a small minimax tree that looks ahead two moves (i.e. *plies*) in the game. Node A is the maximum of B and C; Node B is the minimum of D and E; finally, node C is the minimum of F and G.

If a tree with a branching factor of b is searched to depth d with the minimax procedure, then the size of the tree will be $O(b^d)$. So, the size of a minimax tree grows exponentially with the depth of the tree. However, many of the branches of the search tree may not need to be examined at all since they cannot alter the final score. This characteristic is exploited by a standard enhancement to the minimax algorithm called Alpha-Beta pruning.

The Alpha-Beta algorithm keeps track of two values at each node. One is a lower bound on the score and is called *alpha*, and the other is an upper bound and is called *beta*. If the current value of a node falls outside the alpha and beta bounds, this node is not expanded further. This is illustrated in a simple example shown in Figure 6.2. After searching the left subtree of the root node A, the maximizing player is guaranteed a score of at least 2. When node F is expanded at the right subtree, the minimizing player is guaranteed a score of at most -1 at node C. This means that node G does not need to be searched at all since no matter what its score is going to be, the player at C will have a score of -1 or less. However, the maximizing player at node A will never choose the move leading to C since node B gives a larger score (2).

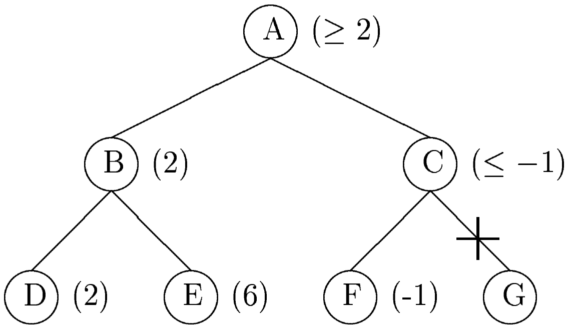


Figure 6.2: Alpha-Beta Pruning

Alpha-Beta pruning may reduce the size of the search tree by a large factor. The best case scenario occurs when at every node the best possible move appears as the leftmost child of that node. In that case, it has been elegantly proven [KM75] that Alpha-Beta builds a tree of size $O(b^{\frac{d}{2}})$, which is approximately the square root of the size of the full minimax tree. In the worst case, Alpha-Beta pruning has no effect and produces the same search tree as a standard minimax algorithm. However, in practice Alpha-Beta will usually have some kind of payoff and the average case seems to be closer to the best case than to the worst one.

The search algorithm used in Antiope is a standard Alpha-Beta implementation, with several search enhancements:

- **Negascout variation.** The Negascout algorithm [Rei83] resembles Alpha-Beta pruning. It further incorporates a minimal window search. In this case, all the moves except for the first one are searched with an Alpha-Beta window of size 0 (beta = alpha + 1), since they are expected to be inferior. If these searches return a value higher than the beta bound, the corresponding subtrees have to be re-searched with a normal search window.

- **Transposition Table.** A transposition table with 2M of entries is used in order to avoid re-searching positions that have already been examined.
- **Fail-soft condition.** This enhancement was introduced by J. Fishburn [Fis81]. It provides an upper (lower) bound on the minimax score when it happens to be less (greater) than the alpha (beta) bound. In general, this enhancement improves search performance by about 5%.

Amazons has a large branching factor, a characteristic that makes deep exhaustive searches impossible. The average branching factor is around 400 and programs typically reach a search depth of 3-4 plies in the opening phase of a game, which increases during the midgame. Still, these search depths are small compared to other games.

Because of the large branching factor, a program that examines all possible moves at each position will not be able to reach a depth large enough for strong play. Thus, some type of selective search scheme must be used to reduce the search effort and Amazons provides the ideal environment for testing the efficiency of selective search techniques.

One selective search method that has been successfully applied to Amazons is Michael Buro's Multi-Probcut [Bur00a]. Multi-Probcut is a forward pruning method and it is based on the idea of predicting the score of a position at depth d by doing a shallow search at a much smaller depth d'. This idea is recursively applied to every node in the search tree. Multi-Probcut is used by Michael Buro in his Amazons program *amsbot*, which is one of the strongest Amazons programs.

Antiope does not use a sophisticated selective search scheme. Since looking at all the possible moves in a position is impractical, Antiope implements a *beam search*. At each position only the k best moves are examined and the rest are ignored, where k has a value of around 20. In order to do this, all the moves in every internal node are sorted according to the static evaluation function value of the resulting positions. This is a very crude pruning method, as it is completely dependent on the static evaluation function, however it is very simple and inexpensive to implement.

6.2 Evaluation Function

Amazons is a territorial board game. The goal of each player is to get more uncontested empty squares than the opponent. The larger the number of squares "owned" by a player, the higher the "probability" that this player will get the last move in the game and therefore win.

In the opening phase of an Amazons game, the number of possible moves is very large. There are 2176 moves in the start position. It is difficult to determine what the optimal sequence of moves is at this stage. There are many good moves that will get a player to the middle game with a reasonable position. The best strategy seems to be to distribute one's queens around the board in such a way that they control (i.e. have access to) as large a part of it as possible.

The evaluation function used to estimate the score of a position must strike a balance between overall control of the board and securing exclusive territory for a player. At the same time, however, it must be mindful of the cost of the evaluation.

6.2.1 Min-distance Heuristic

The most popular evaluation function used in Amazons programs is the min-distance heuristic. This is a simple heuristic that takes into account only territory. It is based on the assumption that a player who can reach a square first with a queen, ignoring arrow shots, owns that square.

The heuristic is based on the following function:

- for each square p, compare db = dist(p,Black) and dw = dist(p,White);
- if $db < dw$: Black owns the square;
- else if $db > dw$: White owns the square;
- else the square is neutral.

Figure 6.3 shows an example of the min-distance function. Note that some squares do not have a min-distance score; the player cannot reach these squares. Figure 6.4 shows the evaluation of the position in Figure 6.3. Squares marked with '?' are the ones that both players can get to in the same number of moves. These squares are considered neutral. In a variation of the min-distance heuristic, neutral squares can be assigned to the player whose turn it is to move.

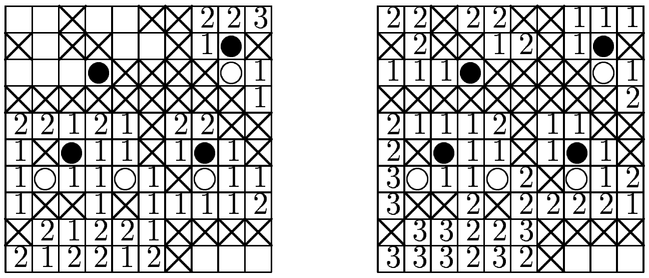


Figure 6.3: Min-distance Heuristic for White (left) and Black (right)

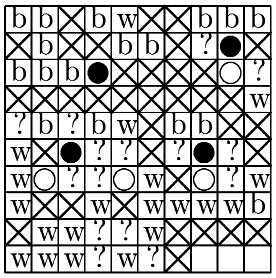


Figure 6.4: Evaluation Using The Min-distance Heuristic

The min-distance heuristic is simple-minded but fast. It can be implemented efficiently and it does a reasonable job of evaluating most endgame positions [MT01]. It usually fails in positions where a single queen has to face many opponent queens, since the heuristic is overly optimistic for the side who has only one queen.

6.2.2 Antiope's Evaluation Function

The min-distance heuristic is very good for the endgame. However, during the opening phase the battle is wide open and a territorial heuristic is not going to be accurate. Antiope uses different evaluation functions for each phase of a game; opening, middle and end game.

Opening Phase

During the opening stage of a game (first 10 moves, where a move is a pair - one by each player), Antiope uses an evaluation function that is based on mobility. The final evaluation function is a weighted sum of four characteristics:

1. **Mobility.** The mobility of a queen is defined as the number of different directions in which it can move, multiplied by the actual number of moves it can make in each direction. A penalty is imposed if there are at most two directions of movement.

2. **Distribution.** During the opening, the queens should be distributed uniformly to the four quadrants of the board so that they can control the whole board. Distribution measures this factor by taking into account how many queens are in each quadrant. Ideally, there should be one queen in each of the four quadrants.

3. **Territory.** This is a combination of two related factors. First, it takes into account the number of half-moves that a player can make. A half-move is defined as a queen move without shooting an arrow. Second, it uses a simplified version of the min-distance heuristic on the squares reachable by half-moves. This characteristic is used to have a smoother transition to the pure territory evaluation later in the game.

4. **Board Control.** The board is divided into 9 roughly equal rectangular areas by the 4^{th} and the 7^{th} row and column. A queen is supposed to have some control of an area if it can get to that area with a half-move. It is not important for a queen to already be in the area. This is taken into account by the *distribution* factor. A player would like to control as many areas as possible during the opening.

A weighted linear sum of the four features above is calculated for each player. The final evaluation score is the difference of these two sums. The most important features are *distribution* and *board control*; the goal of the evaluation function at this stage is to get to the middle game with a balanced position. In that sense, this evaluation function could be considered defensive.

In practice, the disadvantage of the above evaluation function is that the weights need a lot of fine tuning in order for the four features to be combined in the best possible way. The weights could possibly be automatically learned by applying machine learning techniques, such as TD-learning, but this has not been tried for Antiope. On the other hand, in many cases Antiope manages to maintain a reasonable position after the opening phase, which can possibly be improved using the territory evaluation.

Middle Phase

In the middle game Antiope switches to the min-distance heuristic. This happens 10-12 moves into the game. The squares that are considered neutral are not assigned to either player.

End Phase

In the endgame (after 20-22 moves), Antiope continues to use the min-distance heuristic, which gives a fairly good evaluation. However, min-distance is just a heuristic and as such it can give a bad evaluation in certain situations. The ideal situation would be to have a perfect evaluation, especially since the board tends to be split up into separate sub-games of relatively small size. This is where the endgame databases come into use. If a sub-game is small enough to be contained in the databases, then instead of calculating a heuristic score using the min-distance, the exact minimax score for both players is retrieved from the databases and then combined with the score of the static evaluation function.

This is done by assuming that the first player will move in the sub-game where the difference in the minimax scores is the largest; then the opponent will reply in a sub-game that is favourable for him in the same way and so on with the players alternating choosing which sub-game to move on. The differences of the two minimax scores for each sub-game are then summed up, with sub-games for the opponent having negative differences, and added to the heuristic score.

Once the board has been fully decomposed into sub-games that are stored in the databases, then the evaluation of a position is instantaneous and gives an almost perfect score. In that case, the sub-game in which to move is selected using a greedy approach. Every sub-game is examined and a move is made in the one where the difference between the two minimax scores is the maximum.

When a position in the game is reached where the board has not been fully decomposed into sub-games, a decision has to be made on which part of the board to make a move. In a local sub-game or in the rest of the board? After a sub-game is selected using the greedy approach described above, a shallow global search is performed on the rest of the board. The two moves (the one returned by the global search and the one in the best sub-game) are both played out and the static evaluation is used on the resulting positions. The move that results in the position with the highest evaluation score is the one that gets selected.

Conclusion

Mobility and territory can be considered to be two opposite sides of the same coin. Thus, it is not correct to consider an evaluation function based on mobility as completely ignoring territory and vice versa. A major problem with using different evaluation functions at different stages in the game is how to make sure there is a smooth transition from one function to the other.

6.3 Endgame Databases

Antiope makes extensive use of endgame databases. There are currently two types of databases being used. One is the territory database, which stores the defective areas for rooms that contain queens of one color only. The other database contains minimax values for rooms with queens of both colors. The territory databases contain up to 4 same-colored queens, whereas the minimax databases contain a single queen of each color. Both databases store rooms of size 4x6 or smaller.

The databases are used in the evaluation function and give exact values that are added to the evaluation function score. Towards the end of the game, the board usually decomposes into separate independent areas, as can be seen in Figure 6.5. It is unusual for a separate room to be formed early in the game and even if that happens the probability of it being small enough to be in the databases is rather low. So, the evaluation function starts using the databases when the game gets into the endgame phase. This typically happens about 20 moves into the game.

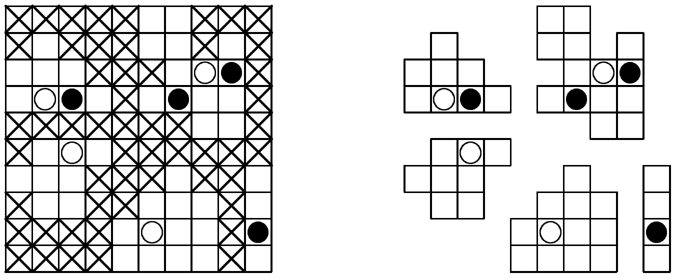


Figure 6.5: Board Decomposition

In every position in the search, the board is scanned so that independent rooms may be identified. Once such a room is recognized and is found stored in the databases, all the queens in the room are excluded from the minimax search. This means that no global minimax search is performed in that room for the rest of the game and search of that particular line of play. This results in less moves to consider, which implies a smaller search space and enables deeper searches. For example, once the position in Figure 6.6 is reached, the queens in rooms 1, 3, 4 and 5 will never be considered in the search again for the remainder of the game.

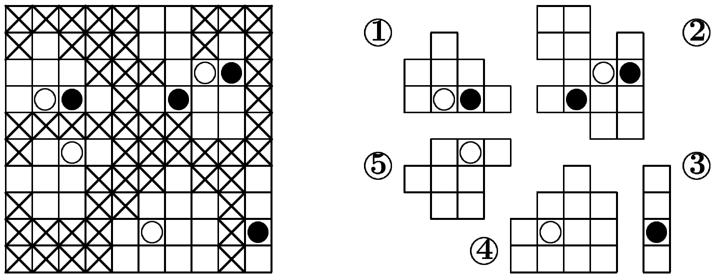


Figure 6.6: Excluded Rooms

The exact value of the local game in an identified room is retrieved from the databases and added to the global evaluation function. Since the room will always decrease in size if a player moves in it, it will always be found in the databases for the rest of the game.

The databases only store the score of a position; they contain no information about the best move. If a move has to be made in a room that is excluded from the global search, then a one move lookahead is performed in the local room. All the moves are generated and played out. The resulting positions are looked up in the databases and the one that offers the highest minimax score determines the move to be made.

Chapter 7

Results

The purpose of developing Antiope is to create a strong Amazons program. The strength of Antiope lies in the endgame databases. The experiments that have been performed aim at evaluating the contribution of the endgame databases to the program's overall play. At the same time, some interesting CGT results were discovered during the construction of the databases, while mining the minimax databases revealed useful information about the game in the form of zugzwangs.

7.1 Interesting Amazons Positions

7.1.1 Combinatorial Values

Nimbers play an important role in Combinatorial Game Theory, since every impartial game can be represented by a Nim heap and thus a nimber. Until recently, no partizan game had ever been found to contain a position with a combinatorial value that is a nimber.

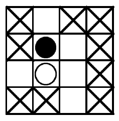


Figure 7.1: The smallest $*2$ position

Amazons was the first partizan game for which a position that is a nimber was discovered. The first such position was discovered by Raymond Georg Snatzke [Sna01] and it was a $*2$ position of size 2x7 with 1 queen of each color and 8 empty squares.

While constructing the combinatorial databases, many positions were discovered whose combinatorial values are the nimbers $*2$ and $*3$. The smallest currently known $*2$ position in Amazons, of size 3x4 with 5 empty squares and 1 queen of each color, was discovered this way and it is shown in Figure 7.1.

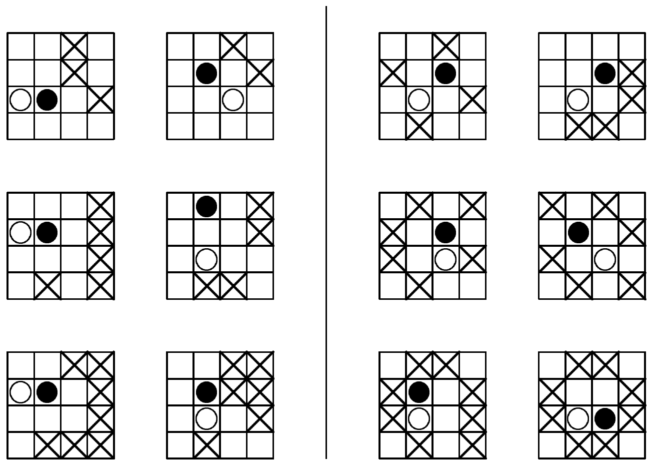


Figure 7.2: Various $*2$ (left) and $*3$ (right) positions

Figure 7.2 shows more positions with values $*2$ and $*3$. Table 7.1 shows the number of $*2$ and $*3$ positions for all the rooms in the 4x4 combinatorial databases.

Size	$*2$	$*3$
7	13	0
8	23	7
9	38	0
10	21	2
11	21	0
12	13	3
13	7	0
14	1	0

Table 7.1: 4x4 small nimbers

According to the rules of nimber addition from Combinatorial Game Theory, $*3 = *2 + *1$. This means that an Amazons position with a combinatorial value of $*3$ can be easily constructed by creating a position with two sub-games, one with a value of $*$ and one with a value of $*2$. However, the databases revealed positions that have a value of $*3$ but consist of only one sub-game. The current challenge is to find an Amazons position whose combinatorial value is $*4$, if such a position exists.

7.1.2 Zugzwang Positions

Zugzwang positions occur quite often in Amazons, especially in the endgame. In some games zugzwangs do not have to be played out. However, in Amazons they do have to be played out since players cannot pass. What makes the positions interesting is that it is non-trivial to play them well [MT01].

Definition of zugzwang positions

Zugzwangs were originally identified in the game of chess but subsequently the definition was extended to cover all games.

> **Zugzwang:** *A position in which a player is forced to make a move that will worsen that player's position.*

The definition can become more precise for Amazons by borrowing some terminology from Combinatorial Game Theory.

> **Amazons Zugzwang:** *A simple zugzwang in Amazons is defined as a game* $a \mid b$ *with* a, b *integers such that* $a < b - 2$.

Figure 7.3 shows an example of *zugzwang* in Amazons. It is a room with one queen of each color. If White is to move, the white queen must shoot back to its origin square to prevent black from moving in. So, one side or the other of the territory below the white queen is lost. White would prefer if Black moved first since Black can only retreat to the territory above and block the entrance, thereby giving White control over all four squares in the territory below. The combinatorial game value of this zugzwang position is $3 \mid 7 = 4$. This means that Black is guaranteed four more moves than White locally, but can get more if White is forced to move first.

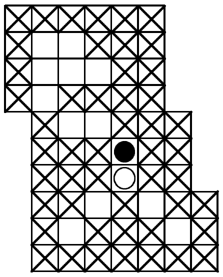


Figure 7.3: Zugzwang in Amazons

In Amazons, the first player to get into zugzwang can still win the overall game. This happens because a zugzwang occurs in a local sub-game and there might be other sub-games that will make up for it. Zugzwang positions do not affect the main question of which player can win a sum game. However, zugzwangs do matter if we want to determine the exact score with optimal play. Optimizing the score is important since it is used as the tie-breaking method in tournaments.

Finding Zugzwang Positions

Zugzwang positions can be identified by performing two minimax searches, one with each player going first. This is exactly how the minimax databases are built so zugzwang positions were extracted from these databases by finding cases where moving first leads to a score that is worse by at least 2 than moving second.

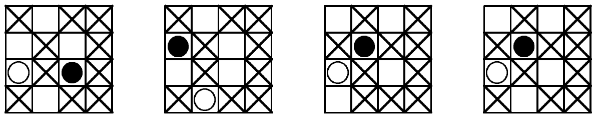


Figure 7.4: Zugzwangs of size 6

Size	Zugzwangs	Total
6	6	7560
7	16	51240
8	216	281820
9	2233	1152612
10	8216	3601665
11	18981	8616135
12	31223	16106574
13	34505	23570274
14	28960	27298089
15	18843	24988740
16	9076	18245160
17	2982	10564752
18	797	4896153
19	119	1773099
20	14	509580

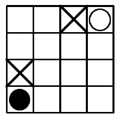
Table 7.2: Zugzwangs in 4x6 rooms

Figure 7.4 shows the four smallest zugzwang positions, which have size 6. Table 7.1 shows the number of zugzwang positions for all rooms up to size 4x6. The numbers under the column called "Zugzwangs" are not exact in the sense that they exclude positions where the black and white queens are interchanged.

7.1.3 Interesting Thermographs

An examination of the 4x4 thermographic databases revealed some interesting positions in terms of temperature and the corresponding thermograph.

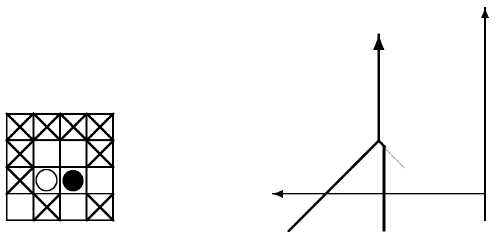
- The largest positive temperature is 11, for the position shown in Figure 7.5.



$g = 11 \mid -11$

Figure 7.5: A position with a temperature of 11

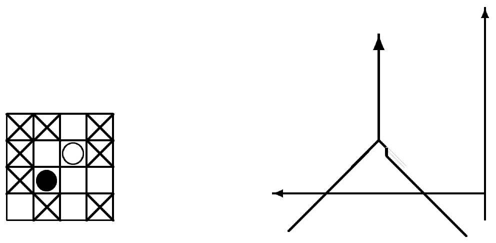
- The largest negative temperature is $-\frac{1}{16}$, for the position shown in Figure 7.6.



$g = \frac{1}{16}$

Figure 7.6: A position with a temperature of $-\frac{1}{16}$

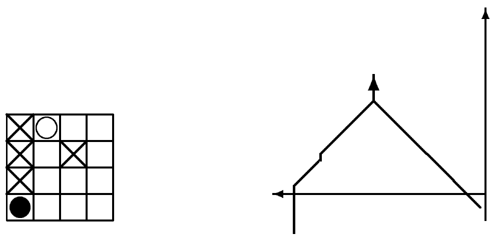
- The smallest positive temperature is $\frac{1}{32}$, for the position shown in Figure 7.7.



$g = \uparrow \mid \frac{1}{16}$

Figure 7.7: A position with a temperature of $\frac{1}{32}$

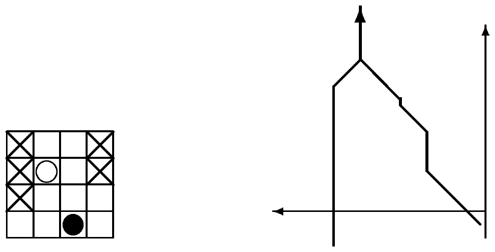
- The largest numerator in a game with fractional temperature is 911, for the position shown in Figure 7.8.



$T = \frac{911}{128}$

Figure 7.8: A position with a temperature of $\frac{911}{128}$

- The largest denominator in a game with fractional temperature is 256, for the position shown in Figure 7.9.



$T = \frac{345}{256}$

Figure 7.9: A position with a temperature of $\frac{345}{256}$

7.2 Antiope

This section presents some data on Antiope's performance. The aim is to illustrate the value of the endgame databases in comparison to plain Alpha-Beta search.

7.2.1 Tournament Results

Antiope has participated in two Amazons tournaments so far. The tournament version of Antiope uses the 4x6 minimax databases and the 4x6 territory databases.

The first tournament was the Amazons tournament at the Computer Olympiad 2000 in London. This tournament had 6 participants and consisted of two games against every opponent, with each player starting with White once. Each program had 30 minutes for all its moves in each game.

Opponent	White	Black
8QP	-26	-6
Yamazon	-14	-9
Anky	-2	-6
ASKA	7	2
Otrere	24	0

Table 7.3: Computer Olympiad 2000

Opponent	Score
De Koning	5 (W)
Tanaka	6.5 (B)
Hensgens	-4.5 (W)
Giles	23.55 (W)
Lerner	8 (B)
Coll	12.5 (B)

Table 7.4: Jenazon Cup

The second tournament was the Jenazon Cup, a tournament with arbitrary Human+Computer teams organized on Michael Buro's online GGS server [Bur98] in November-December 2001. There were 7 participants. Each team played against each of the other teams once. The total time for each game was 90 minutes. In this tournament, Antiope formed a team with its author and would in general play all the moves after about 10 moves into the game.

Tables 7.3 and 7.4 show the corresponding results. A positive score means that Antiope won the game by that many squares, whereas a negative score means that Antiope lost the game by that many squares. The half points reported in the results of the Jenazon Cup are due to komi bidding. Antiope finished 4th in the Computer Olympiad and tied for 1st place in the Jenazon Cup.

There are two reasons that Antiope performed better at the Jenazon Cup than at Computer Olympiad 2000. The most important one is the correction of several bugs in the search algorithm and the use of the transposition table, as well as several other minor enhancements in the code. The second reason is that Antiope's author played out the opening of the games, thus allowing Antiope a very balanced game in the midgame. The use of the min-distance heuristic during the opening phase would have got Antiope in a contestable position after the opening but a knowledgeable human can always outperform current Amazons programs in the opening phase of the game.

7.2.2 Search and Endgame Databases

Self-play games do not illustrate the full potential of the databases, since most of the time there are few rooms formed in the endgame, at least not ones that are stored in the databases. This is especially true for rooms with opposing queens. Territories appear much more often. Human games seem to be even worse in that respect since they do not usually end with close battles between the two players.

However, practice has shown that the databases can make a difference. Antiope has participated in two Amazons tournaments so far and there were games where the databases came into play.

Impact on the search

Figure 7.10 shows a position from a recent game played for the Jenazon Cup against 8QP, which is one of the strongest programs in the world.

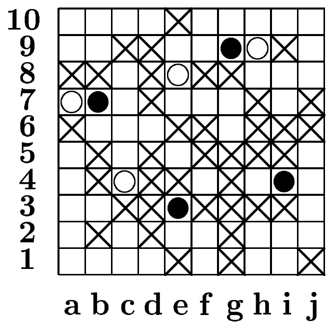


Figure 7.10: Test endgame position

This position was used in an experiment to illustrate the impact of the endgame databases in the search. Two different versions of Antiope were used. One (AB) did not make any use of databases. The other (AB+DB) used the 4x6 minimax and territory databases, just like the tournament version of Antiope.

At each intermediate position till the end of the game (for a total of 43 one-player moves), the two programs were used to evaluate the position. Each performed a 6-ply fixed-depth search. All the settings in the two programs were exactly the same, except for the use of the databases.

Figure 7.11 shows the size of the search tree at each position. When the board is fully decomposed into sub-games contained in the databases, the search tree includes only one node since there is no search involved; the score for each sub-game is retrieved from the databases.

The graph shows that the databases reduce the size of the search tree considerably. This happens because all the queens contained in sub-games identified during search are excluded from the rest of the search. Notice that the program with the databases formed a larger search tree in two positions. This is expected since the databases can provide more accurate values and make the program search more nodes to prove the main line of play.

Figure 7.12 shows the score of the evaluation function for the two programs at each position. The program with the databases finds the final score of the game 30 one-player moves (i.e. plies) before the end of the game. The plain Alpha-Beta version though, discovers the true score only 7 plies before the end. For most of the game, the non-database version of Antiope underestimates the final score. This happens because towards the end, the board contains defective territories and plain Alpha-Beta cannot possibly identify them.

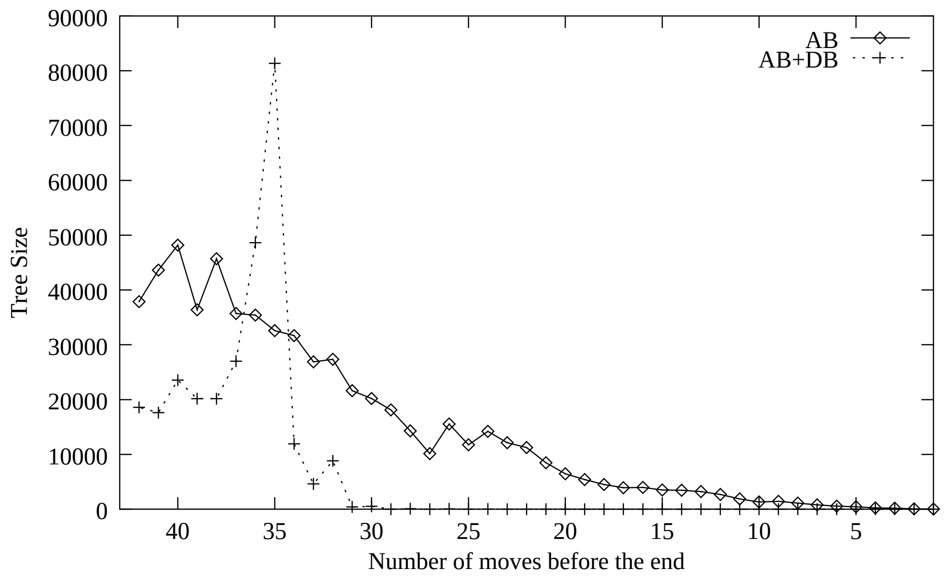


Figure 7.11: Size of search tree

Example endgame

The same game against 8QP that was used in the previous section contains an endgame position that provides a good example for analysis of the impact that the databases can have in a game.

Figure 7.13 shows the endgame position, which occurred 24 moves into the game. This position contains two interesting rooms. Room A contains the queens at $c5$ and $b6$, while room B contains the queens at $e3$ and $d1$.

Room A is a typical Amazons zugzwang, with a value of $\{1 \mid 4\}$. If Black moves first, the queen at $b6$ has to move up to one of the three squares and shoot back at $b6$ in order to prevent the white queen from getting more squares. However, this means Black will lose the square at $a5$. In that case, the score of this sub-game is $+1$ (for Black). If White moves first, then Black will be able to get all the four remaining squares, for a local score of $+4$ (for Black).

So, neither player would like to move a queen in room A. Antiope, who was playing White, could see the room in the databases and would never move the queen at $c5$ unless there was no other move to make. Since in this position Black has fewer overall squares than White, Black would eventually be forced to move the queen at $b6$ and lose two squares.

Room B, on the other hand, will become a territory for Black. Black has to block the $e2$ square to prevent White from entering the room. The only move that does this, and ensures that Black gets all the remaining squares in the room, is for the black queen to move to $e2$ and shoot at $f1$. Any other move makes the room a defective territory. In the actual game, the opponent program made a blunder (e3-f2-e2), which resulted in the room becoming defective by one square. The reason is that the min-distance heuristic cannot see that possibility. Antiope would have made the correct move since the resulting territory can be found in the databases.

In this specific endgame Black eventually scored 1 square less than it could have achieved. In a close game, this can make the difference between winning

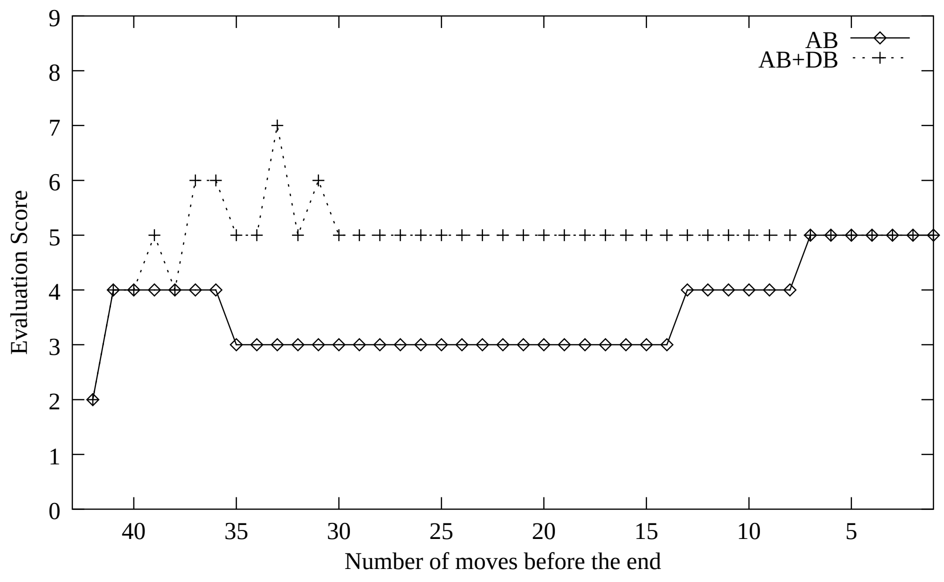


Figure 7.12: Evaluation function score

the game and losing it. The databases ensure that in cases like this Antiope's play will be better than what any possible heuristic evaluation function can do.

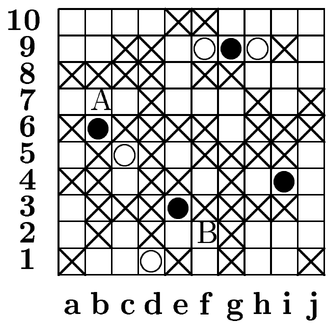


Figure 7.13: Example endgame - Antiope (White) vs 8QP (Black)

In a game against an opponent without endgame databases, it is quite likely that the opponent will play into defective areas, since they look better for them than for us. So, minimax search will tend to select the defective territories.

7.2.3 SenteStrat vs Full Board Search

SenteStrat is a thermographic strategy used to choose a move when a player has to select among several hot sub-games. A set of experiments were carried out to compare the performance of SenteStrat against that of Alpha-Beta search on a set of endgame positions.

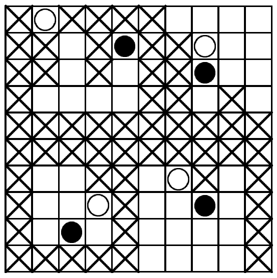


Figure 7.14: Example setup position

Both programs use the territory databases. In theory, a game can end when every sub-game has a temperature of -1, (e.g. territories and zugzwangs). However, for practical purposes a game ends when all the sub-games contain only one queen and the final score is calculated by subtracting the squares belonging to White from the squares belonging to Black.

The Alpha-Beta program uses a negascout variation of Alpha-Beta pruning. The heuristic evaluation used is the min-distance heuristic, where squares that have the same distance from both players are not assigned to any player. This program searches for 40 seconds per move, a time allocation that is common in Amazons tournaments. The typical search depth at the initial position is 5 or 6 plies and it increases to more than 10 plies towards the end.

Position	Empty Squares	Alpha-Beta	SenteStrat
1	36	+0	4
2	34	-3	4
3	38	-1	-0
4	38	-1	4
5	36	-1	6
6	34	+0	4
7	45	+0	1
8	43	-6	1
9	41	5	6
10	52	-1	4

Table 7.5: Alpha-Beta vs SenteStrat

The program that implements SenteStrat uses the thermographic endgame databases. It plays instantaneously since all the sub-games are already contained in the databases. The version used does not have any special knowledge about zugzwangs. It will consider moving in every sub-game which contains two opposing queens, even if it has a temperature of -1. However, moves in territories are ignored since they represent undisputed areas.

In order to get experimental results a set of 10 endgame positions was constructed. All the positions initially consist of 4 independent sub-games, each of size up to 4x5. Every sub-game contains one queen of each color. Figure 7.14 shows one of the positions. For each of the 10 positions two games were played out, with each of the two programs starting with Black. This amounts to a total of 20 games, 2 for each position. Table 7.5 shows the results. A positive value indicates a win for Left (Black) and a negative value a win for Right (White). The values represent the difference in squares. A win for Black by 0 squares is indicated as $+0$ and a respective win for White by -0. The two columns (Alpha-Beta, SenteStrat) indicate the player who plays Black.

The results show that SenteStrat outperforms Alpha-Beta. Even when the two methods share the wins in a position, SenteStrat gets a better score. However, this should not be surprising. The positions contain too many squares for Alpha-Beta to search very deep, whereas SenteStrat plays instantly by using the thermographic databases.

The previous experiment shows the power of Combinatorial Game Theory in the form of thermographic strategies. In practice though, it is unlikely for the board to fully decompose into four independent sub-games with opposing queens in them. The decomposition usually appears late in the endgame and often consists only of a couple of rooms with opposing queens.

Chapter 8

Conclusions and Future Work

The general conclusion at the current stage in the development of Antiope is that it is a strong program and tournament games have shown that it is probably the strongest Amazons program in the endgame. However, the program exhibits a weakness in the opening phase of the game, especially against programs that use a territory heuristic.

8.1 Discussion

Antiope is the first program to use endgame databases in Amazons. During the construction of the databases several interesting results were discovered by mining the databases: new Combinatorial Game Theory values and interesting Amazons positions that provide some insight on the complexity of the game.

However, the important question has to do with the impact of the databases in the playing strength of the program. Do the databases make a difference in the search and consequently in the performance of Antiope?

Experiments with the size of the search tree and the score of the evaluation function have shown that the databases reduce the search space considerably and provide a more accurate evaluation of the positions. Also, tournament games have shown that the endgame databases do make a difference. The territory databases outperform any territory heuristic and often Alpha-Beta programs miss defective territories. Active areas may not appear very often but when they do the databases are likely to help.

One possible question is what sub-game size should be stored in the databases. For territory databases this should not be a problem. The defective territories are a small fraction of the total number of positions so larger databases could easily be constructed. This would enable programs to fill territories optimally.

For the active area databases things are different. There are memory restrictions that limit the databases. The current size of 4x6 intuitively seems good but of course it is not large enough. It is always painful to see a sub-game formed that is not in the current databases because of one extra square. The key point probably lies in constructing databases with various room sizes so that they cover the largest portion of the board possibly.

One problem with the current version of Antiope is the integration of the databases into the Alpha-Beta search. There is no clear-cut best solution to this problem; only heuristic approaches and experimentation can provide insight on the best approach.

Another problem is the evaluation function at the opening phase of the game. At this stage it considers mostly mobility. Early tournament games revealed that Antiope performed badly against programs that used a territory heuristic from the beginning. This was mainly due to the fact that the evaluation function was very shaky because the weights had not been set properly. Adjusting the weights is a tedious task to do manually. Instead, practice showed that the use of the min-distance heuristic from the beginning of the game was good enough to allow Antiope to enter the midgame with a fairly balanced position.

In general, considering two Amazons program of comparable strength, the positions reached in the endgame should be quite balanced. In this case, the use of the endgame databases is what will give the winning edge to the programs and this has been verified in practice.

8.2 Future Work

There are many interesting things to do with the game of Amazons. Current work can always be improved upon. Moreover, many ideas were not implemented as part of this thesis but would make interesting research projects.

8.2.1 More Databases

The current endgame databases are limited in two ways. First, the size of the sub-games; second, the number of queens in them.

The current size of 4x6 (for the minimax databases) was chosen based on available memory. These databases take up about 400 MB of RAM. Building larger rooms is restricted mainly by available memory and not so much by computation time. As more computer resources become available, building larger databases should become easier.

Apart from larger room sizes, it is very important to build databases with rooms of various sizes. For example, a room of size 3x7 would not be found on the current databases. However, it is feasible to calculate all the databases with rooms of size up to 3x7 since the total number of squares is less than that for the 4x6 databases. Rooms of different sizes could be built covering many portions of the board.

As far as the number of queens is concerned, there is no problem for territories since the databases already contain rooms with up to 4 queens, which is the number of queens used in Amazons.

However, the current databases with active areas contain only 1 queen of each color. Although sub-games with more queens appear considerably less frequently, they would be useful nonetheless. All the databases with up to 4 queens for each player could possibly be constructed.

This is fairly straightforward for the minimax and the combinatorial databases. All it requires is extra coding and of course the territory databases. However, as mentioned in Section 4.5, this is not so easy to do with the thermographic databases. It can be done easily for the $n$vs1 databases, but for more queens there is a problem. The whole framework of the program that builds the databases has to be changed and instead of 1-ply lookaheads, an exhaustive search until there are no more moves left in the sub-game must be performed.

8.2.2 Better Integration of the Databases in the Search

The tournament version of Antiope is using the minimax databases. This is mainly done because those databases contain larger rooms, although overall play is not as good as could be achieved by the CGT databases. The CGT databases are used in the version of Antiope that uses SenteStrat for fully decomposed rooms. However, there is a lot of work remaining on how to use the databases efficiently in the global Alpha-Beta search.

The scores from the minimax databases are integrated in the heuristic evaluation function in a greedy manner. There are other ways of doing it apart from the one described in Section 6.2.2. One variation is to consider the players alternating moving in sub-games where their minimax score is the maximum. Another variation could take the mean of the difference of the minimax scores in each sub-game and add it to the score reported by the static evaluation function.

As for the CGT databases, they offer better possibilities. The sum of means for every sub-game can be calculated and added to the score of the evaluation function. A variation of this approach would additionally add one half of the maximum temperature in all sub-games. As for selecting which sub-game to move in, there are various alternatives. The selected sub-game could be the one with the maximum temperature; or the one with the largest mean value.

The possible alternatives of integrating the databases in the global search are numerous. None of them is theoretically optimal. The best one should be defined through experiments of playing actual games to see which approach performs the best.

8.2.3 Not Completely Isolated Rooms

The current databases contain rooms that are completely separated. However, sometimes a queen can be standing on the border of an area as in Figure 8.1.

In this case, the white queen controls all the squares inside area A and can actually improve its position by moving to area B. So, the squares in area A are a lower bound on the score for White. However, this position is not contained in the 4x6 databases, since area A, together with the white queen on the border, form a 6x5 room.

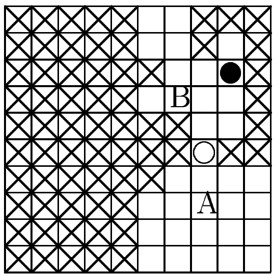


Figure 8.1: Non-separated room

This situation can be handled by expanding the databases so that they include all the cases with queens on the borders of a room. This should not be too difficult since the empty rooms are constructed first and the queens are subsequently placed in them. However, the difficulty lies in scanning the board for sub-games during a game. Rooms with queens on the border have to be properly identified.

8.2.4 Articulation Points

The min-distance heuristic is the most popular territory heuristic used in Amazons programs. In general it gives good evaluations, especially towards the end of a game. However, this heuristic can be inaccurate even for very simple positions. Figure 8.2 shows two such examples.

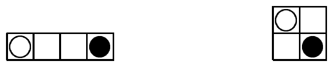


Figure 8.2: Inaccurate min-distance evaluations

The left side of Figure 8.2 shows a position where ignoring neutral squares leads to the evaluation of the position as a draw (0 square difference), whereas the first player to move clearly has the square advantage. The right side of Figure 8.2 shows a position where assigning neutral squares to the player whose turn it is to move leads to the position being evaluated as a +2 square advantage for the first player, whereas the square difference is clearly 0.

One problem with the min-distance heuristic is that it does not take into account arrows. If arrows were considered, the position on the left side of Figure 8.2 would be evaluated correctly. One way to take arrows into account is by considering *articulation points.*

Articulation points, similar to their definition for graphs, are points whose blocking with an arrow separates a room into two separate areas. Figure 8.3 shows an example. Assuming that Black is to play first, the ordinary min-distance heuristic would evaluate the position as a draw. However, square A is an articulation point. Black can move to square B and shoot at A, thus completely blocking off White.

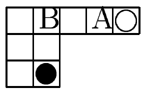


Figure 8.3: A room that contains articulation points

Once articulation points have been identified, the min-distance heuristic can be improved by extending the search by one ply for moves that block an articulation point.

8.2.5 Selective Search

Full-width search in Amazons is too expensive (i.e. slow) because of the large average branching factor. This makes Amazons a good test bed for selective search techniques.

There is an apparent contradiction though. The wins by Johan de Koning's program 8QP in the last two Computer Olympiads have shown that full-width search can be effective even in Amazons. However, it would be interesting to see how effective various selective search methods can be and whether they can produce a stronger program.

One selective search method that has already been applied in Amazons is Multi-Probcut [Bur00a]. It was first used in Othello and subsequently Michael Buro implemented it in his Amazons program (see Section 6.1).

Another method that could prove useful is Singular Extensions [ACH90]. This is a search extension technique that tries to identifies a move that is "much better" than the alternative moves. This means that it returns a score that is better by a significant margin than the score returned by all the other moves. Once a move is identified as singular, the search depth for the resulting position is extended by one ply.

Finally, some programs use beam search as a rough pruning method. A variation of beam search is *tapered search*, where k moves are considered at each node in the search tree but k diminishes as the search depth increases. Although these methods are risky since good moves can be ignored, it remains to be seen whether the increased search depth provides considerable pay off.

Bibliography

[ACH90] T. Anantharaman, M.S. Campbell, and F. Hsu. Singular Extensions: Adding Selectivity to Brute-Force Searching, 1990.

[BCG82] E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning Ways for Your Mathematical Plays.* Academic Press, London, 1982.

[Ber96] E. Berlekamp. The Economist's View of Combinatorial Games. In R. Nowakowski, editor, *Games of No Chance: Combinatorial Games at MSRI*, pages 365–405. Cambridge University Press, 1996.

[Ber00] E. R. Berlekamp. Sums of $N \times 2$ Amazons. In *Institute of Mathematics Statistics Lecture Notes*, number 35 in Monograph Series, pages 1–34, 2000.

[Bur97] M. Buro. The Othello Match of the Year: Takeshi Murakami vs. Logistello. *ICCA Journal*, 20(3):189–193, 1997.

[Bur98] M. Buro. Generic Game Server. `http://www.neci.nj.nec.com/homepages/mic/ggsa/ggsa.html`, 1998.

[Bur00a] M. Buro. Experiments with Multi-ProbCut and a New High-Quality Evaluation Function for Othello. In J. van den Herik and H. Iida, editors, *Games in AI Research*, pages 77–96, Maastricht, 2000. Universiteit Maastricht.

[Bur00b] M. Buro. Simple Amazons Endgames and their Connection to Hamilton Circuits in Cubic Subgrid Graphs. In *Proceedings of Second International Conference on Computers and Games*, 2000.

[Con76] J. H. Conway. *On Numbers and Games.* Academic Press, London, 1976.

[CS98] J. Culberson and J. Schaeffer. Pattern Databases. *Computational Intelligence*, 14(3):318–334, 1998.

[Fis81] J. Fishburn. *Analysis of Speedup in Distributed Algorithms.* PhD thesis, University of Wisconsin, Madison, 1981.

[Gas95] R. Gasser. *Harnessing Computational Resources for Efficient Exhaustive Search.* PhD thesis, ETH Zurich, 1995.

[Kel94] M. Keller. World Game Review. `http://members.aol.com/wgreview/`, 1994.

[KM75] D.E. Knuth and R.W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6:293–326, 1975.

[McK81] Brendan McKay. Practical graph isomorphism. Congressus Numerantium, 30:45–87, 1981.

[McK00] Brendan McKay. Personal communication, 2000.

[McK01] Brendan McKay. The nauty page. `http://cs.anu.edu.au/bdm/nauty/`, 2001.

[MT01] M. Müller and T. Tegos. Experiments in Computer Amazons. In R. Nowakowski, editor, *More Games of No Chance*. Cambridge University Press, 2001. To appear.

[Mül99] M. Müller. Decomposition Search: A Combinatorial Games Approach to Game Tree Search, with Applications to Solving Go Endgames. In *IJCAI-99*, pages 578–583, 1999.

[NOS93] NOST. kNights Of the Square Table. `http://www.nostgames.com/`, 1993.

[Rei83] A. Reinefeld. An Improvement of the Scout Tree Search Algorithm. *ICCA Journal*, 6(4):4–14, 1983.

[Rog96] R. Rognlie. Richard's Play-By-eMail server. `http://www.gamerz.net/pbmserv/`, 1996.

[Sch97] J. Schaeffer. *One Jump Ahead: Challenging Human Supremacy in Checkers*. Springer-Verlag, New York, 1997.

[Sha50] C. E. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41:256–275, 1950.

[Sna01] R. G. Snatzke. Amazons. In R. Nowakowski, editor, *More Games of No Chance*. Cambridge University Press, 2001. To appear.

[SP97] J. Schaeffer and A. Plaat. Kasparov Versus Deep Blue: The Rematch. *ICCA Journal*, 20(2):95–101, 1997.

[Tes94] G. Tesauro. TD-Gammon, A Self-teaching Backgammon Program, Achieves Master-level Play. *Neural Computation*, 6:215–219, 1994.

[Tho86] K. Thompson. Retrograde Analysis of Certain Endgames. *ICCA Journal*, 9(3):131–139, 1986.

[TSBB53] A. M. Turing, C. Strachey, M. A. Bates, and B. V. Bowden. Digital computers applied to games. In B. V. Bowden, editor, *Faster Than Thought*, pages 286–310. Pitman, London, 1953.

[vdG00] R. van der Goot. Awari Retrograde Analysis. In *Proceedings of Second International Conference on Computers and Games*, 2000.

[Wol96] D. Wolfe. The Gamesman's Toolkit. In R. Nowakowski, editor, *Games of No Chance: Combinatorial Games at MSRI*, pages 93–98. Cambridge University Press, 1996.

Appendix A

Territory Tables

Tables A.1, A.2, A.3 and A.4, show the number of defective rooms, defective territories and total territories in the 4x5 databases. The difference between a room and a territory is that the former consists only of empty squares whereas the latter is a room with same-colored queens placed in it. This explains why the number of territories in the tables is always larger or equal to the number of rooms; one or more territories may correspond to the same room. As can easily be seen in those tables, the number of defective territories is significantly less than the total number of territories.

Size	Defective Rooms	Defective Territories	Total Territories
3	2	2	15
4	1	3	88
5	21	37	465
6	76	178	2634
7	381	831	12075
8	864	1856	41640
9	1408	2791	103257
10	1374	2423	189040
11	847	1345	257004
12	290	393	264840
13	53	60	206349
14	3	3	123774

Table A.1: 4x5 1-queen territories

Tables A.5, A.6, A.7 and A.8, show the number of k-defective territories in the 4x5 databases. It is important to notice that there are only territories

Size	Defective Rooms	Defective Territories	Total Territories
4	2	3	132
5	39	45	930
6	43	165	6585
7	278	762	36225
8	529	1599	145740
9	639	2084	413028
10	468	1318	850680
11	177	427	1285020
12	27	54	1456620
13	2	2	1238094

Table A.2: 4x5 2-queen territories

Size	Defective Rooms	Defective Territories	Total Territories
5	12	12	930
6	65	202	8780
7	476	1100	60375
8	520	2342	291480
9	761	3066	963732
10	490	1908	2268480
11	191	674	3855060
12	35	77	4855400
13	1	2	4539678

Table A.3: 4x5 3-queen territories

with 4 or less defects. Also, as pointed out in Chapter 4, having more queens in a territory does not necessarily mean that it is easier to completely fill the territory. This can easily be seen in some of the tables where the corresponding number of defective territories is larger when there are more queens.

Size	Defective Rooms	Defective Territories	Total Territories
6	57	59	6585
7	325	862	60375
8	712	3849	364350
9	1394	6045	1445598
10	673	3617	3969840
11	275	1207	7710120
12	59	140	10924650
13	6	13	11349195

Table A.4: 4x5 4-queen territories

Size	1-defective	2-defective	3-defective	4-defective
3	2	-	-	-
4	2	1	-	-
5	37	-	-	-
6	155	23	-	-
7	783	43	5	-
8	1705	151	-	-
9	2618	170	3	-
10	2256	161	6	-
11	1238	88	19	-
12	364	24	5	-
13	58	1	-	1
14	3	-	-	-

Table A.5: 4x5 1-queen Defective Areas

Size	1-defective	2-defective	3-defective
4	3	-	-
5	45	-	-
6	142	23	-
7	727	31	4
8	1467	132	-
9	2021	60	3
10	1285	33	-
11	415	12	-
12	53	1	-
13	2	-	-

Table A.6: 4x5 2-queen Defective Areas

Size	1-defective	2-defective	3-defective
5	12	-	-
6	202	-	-
7	1076	24	-
8	2191	151	-
9	3016	47	3
10	1889	19	-
11	671	3	-
12	77	-	-
13	2	-	-

Table A.7: 4x5 3-queen Defective Areas

Size	1-defective	2-defective
6	59	-
7	862	-
8	3814	35
9	5968	77
10	3575	42
11	1207	-
12	140	-
13	13	-

Table A.8: 4x5 4-queen Defective Areas

Index