國立臺灣師範大學
資訊工程研究所碩士論文

指導教授：林順喜　博士

六子棋中一個結合迫著搜尋的防禦性策略

A Defensive Strategy Combined with Threat-Space Search for Connect6

研究生：劉雲青　撰

中華民國九十八年六月

誌謝

碩士的學習生涯轉眼間就要告一個段落，在這些日子裡，我非常感謝指導教授林順喜博士的細心指導，使我對學術研究從茫然無知到略有所悟，並在過程中獲得一生受用不盡的經驗和教誨。除了學術專業上的指導外，其做人處事的原則與態度亦為學生所景仰和效法。

感謝口試委員許舜欽教授、 徐讚昇教授、 吳毅成教授、 陳善泰教授對學生研究成果的肯定，並提出諸多寶貴的指正與建議，使得研究與論文品質更能更趨於完善。特別感謝許舜欽教授、 徐讚昇教授、 和吳毅成教授於西班牙比賽期間的照顧與協助。另外也特別感謝顏士淨教授於北京比賽期間的幫忙與照顧。

感謝我們研究群中所有學長、同學和學弟在研究上的建議和幫忙。 特別感士傑學長、立德學長和義雄學長平時的指點與勉勵，你們對研究的投入與熱情和認真謙和的態度永遠是我的榜樣。 另外也謝謝曜安學長、 信翰、 聖群、 俊廷、 傑淳、 政孝、 昱臣、 啟峰、 俊佑 、明真和冠明在這些日子裡的陪伴，願大家同窗之誼永誌長存。

最重要的是感謝我的父母家人在背後默默的支持，使我可以無後顧之憂的專心從事研究，順利完成論文。

其他尚有許多的人無形中對此研究提供諸多的協助，這裡也表達對他們的我最誠摯的感謝之意，期許自己未來加倍努力，望能不辜負大家對我的付出和期待。

摘要

k 子棋相關的研究一直有許多有趣的研究成果被發表出來，而其中於 2005 年所提出的六子棋則格外受到注目。 從 2006 年開始，六子棋一直被列為 ICGA 電腦奧林匹亞中重要的比賽之一，投入六子棋的研究和參與比賽的團隊逐年增加。

這篇論文裡，從簡介現今應用在六子棋的相關技術和研究成果開始，接著提出六子棋上的一個防禦性策略以及一個策略性的審局方案。 迫著搜尋演算法在電腦六子棋中有相當重要的地位，論文裡亦提出提升其實作上效能和精確度的技術。 防禦性策略和迫著搜尋的結合也會被探討。

防禦性策略和迫著搜尋的結合的表現足以跟現今的一些頂尖的六子棋軟體分庭抗禮 ，可見其為一個相當有效的方法。我們整合這些技術所實作出的程式 Kagami，於第十四屆 ICGA 電腦奧林匹亞中獲得第四名。

Abstract

The study of k-in-a-row games has produced a number of interesting results, and one of its sub-category Connect6 proposed in 2005 has been of particular interest. Since 2006, Connect6 has been included as one of the major competition in the ICGA Computer Olympiad, and is gaining more popularity every year.

In this thesis, we briefly review current methods applied in Connect6 and related results. A defensive strategy is introduced along with a more strategically sensitive evaluation scheme. Threat-space search is an important algorithm applied in Connect6, some techniques for gaining more efficiency and accuracy will be introduced. The integration of the defensive strategy and threat-space search will also be investigated.

The combination of the defensive strategy and threat-space search is proved to be effective, and is able to compete with other top Connect6 programs. The program Kagami, which was implemented with these methods, won the fourth place in the 14th Computer Olympiads.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Playing games is always considered to be one of the most significant feature of intellectual behaviors, thus the study of games is considered to be a possible key to the secrets of intelligence. It serves as a vehicle to conduct research in a wide range of fields, such as cognitive science, game theory, artificial intelligence, and so on.

Games are ideal domains for exploring the capabilities of artificial intelligence (AI), since they have a clear way to measure and evaluate their performances. They are also much simpler and more well-defined than most of the real-world problems, but not so simple as to be trivial. Many of the techniques developed over the years in this domain have been applied to a variety of different real-world applications. Although chess has long time been called the Drosophila of AI[1], the research of other games has also provided fruitful results.

Divergent games, are the games that are immune to retrograde analysis, and thus search and knowledge-based methods must be used[2]. The family of connection games, such as Connect-Four, Qubic, Go-moku, Hex, is one of the most widely studied category of games. They are widely played around the world, and their rules are simpler than most of the other board games, while retaining sufficient complexity to be a challenging and interesting game.

1.1 Connect6

Connect6 belongs to the family of k-in-a-row games. It was first proposed by Wu and Huang in 2005[3][4], and has been included as one of the major competitions in the ICGA Computer Olympiad since 2006.

The basic settings are the same as the game of Go, where two players play with black and white stones alternately on a 19×19 board. The first player plays with black, and the second player plays with white. Apart from the first move, which the first player places only one black stone, both players place two stones of their own color alternately. The player to get six or more consecutive stones in a row, column or diagonal first wins. If all the squares on the board are filled, and neither player cannot connect, the game is declared a draw. Figure 1.1 shows a game of Connect6, where the first player (black) wins.

Although a 19×19 Go board is mostly used, a 59×59 board is recommended for professional players, and computer tournaments. The current board size used in the ICGA Computer Olympiad is 19×19.

1.2 Goal and Motivation

Many artificial intelligence methods applied in Connect6 are largely under the influence of Allis's work on solving Go-Moku[5][6], where threat-space search was proposed and applied. Wu and Huang proposed a Connect6 version of threat-space search, and generalized it to the family of k-in-a-row games[3][7]. It became the foundation of today's Connect6 programs, and much effort has been made to increase the speed and accuracy. The stage between the start

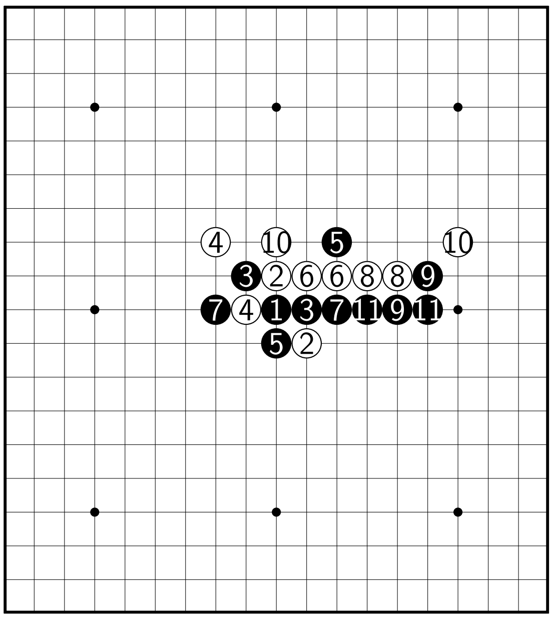

Figure 1.1. A game of Connect6

of the game and a winning sequence found by threat-space search is where the evaluation heuristics have more influence. Allis used proof number search for this stage in Go-Moku, while in Connect6 most programs used $\alpha\beta$-search.

Connect6 is a relatively new board game, and the development of the evaluation heuristics is still in its infancy, therefore there are still many unknown territories to explore.

The goal of this thesis is to introduce a defense-oriented strategy in Connect6, and present a new evaluation scheme. Besides demonstrating its capabilities, we also make some efforts in improving the implementation of threat-space search in Connect6.

1.3 Thesis Outline

Chapter 2 reviews the current state-of-the-art in Connect6 and k-in-a-row games, and briefly describes some known results and methods. Some preliminary observations and a defensive strategy with an evaluation scheme is proposed in Chapter 3. Chapter 4 will present threat-based strategy and threat-space search, with discussions on the issues of improving efficiency and accuracy in the implementation. Chapter 5 will give a conclusion and point out some directions for further investigations.

Chapter 2

Related Results

In this chapter we briefly describe and review past results and current methods. Section 2.1 explores the family of k-in-a-row games. Section 2.3 investigates its various variations. Section 2.3 discusses the complexity of Connect6.

2.1 The Family of k-in-a-row Games

Tic-Tac-Toe is a popular game played around the world, and most people are familiar with it. Figure 2.1 shows a game of Tic-Tac-Toe. It is played on a 3×3 board, by two players. One player plays with naughts and the other plays with crosses, taking turns to place their pieces on the board. The first player to get three of their pieces in a row, column, or diagonal wins. From this point on, the phrase "k-in-a-row" will mean getting k consecutive stones in a row, column, or diagonal, unless explicitly stated otherwise.

Tic-tac-toe can be characterized as a kind of m,n,k-games, with m and n denoting the number of rows and columns of the board, and k giving the length of straight chains to be obtained. Therefore the game of tic-tac-toe can be characterized as the 3,3,3-game, whereas the popular game Go-Moku

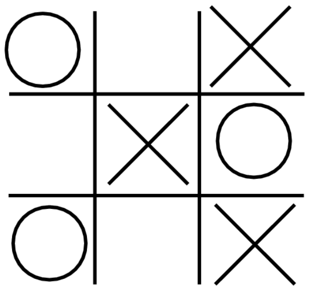

Figure 2.1. The 3,3,3-game, Tic-Tac-Toe

is the 15,15,5-game.

A number of game-theoretic values has been obtained for m,n,k-games. By applying the strategy stealing argument, it can be shown that the second player cannot have a winning strategy, thus the theoretic values can only be a win for the first player or a draw. It is discovered that the first player has a greater advantage on larger-sized boards or with shorter chain lengths[8][9]. The trivial 1-in-a-row game is a win for the first player on any board size of at least 1×1. The 2-in-a-row game is a win for the first player on any board with a size of at least 2×2, and the 3-in-a-row game is a win for the first player on any board with a size larger than 3×3[8][9].

As for 4-in-a-row game, it is shown that all m,n,4-games with $m \geq 6$ and $n \geq 5$ are game-theoretic wins for the first player. The board size which 4-in-a-row game changes its theoretic value from win to draw for the first player lies in m,4,4-games, where the precise value of m lies between 9 and 29, both inclusive[9]. Moving on to 5-in-a-row game, it is demonstrated by using the Hales-Jewett pairing strategy that the second player can achieve a draw in the 5,5,5-game[8]. Figure 2.2 provides an example of the strategy. In Figure 2.2 any possible straight chain of five squares contains two “paired” squares indicated by a marker in the direction of the chain. The second player can guarantee the draw by always playing the second square of the pair as soon as the first player plays the first. It is also shown that 5-in-a-row game is drawn on 6×5 and 6×6 boards. Larger boards are not yet investigated. Allis demonstrated that Go-Moku (i.e., the 15,15,5-game) is a first player win[5][6]. Same as 4-in-a-row game, the borderline between a draw and a win for the first player is still an open question[9].

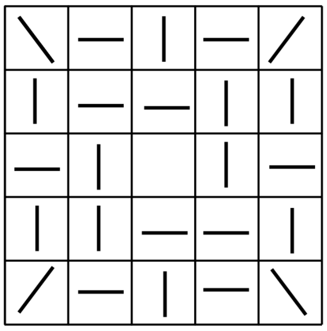

Figure 2.2. Hales-Jewett pairing strategy for the 5,5,5-game

In 1954, Henry Oliver Pollak and Claude Elwood Shannon showed that 9-in-a-row game is drawn on an infinite board, and as a consequence on any finite board and m-in-row game with $m \geq 9$[8]. Later, Zetters demonstrated the same result for 8-in-a-row game[10]. The two proofs use similar methods. They both show that if the second player follows a certain strategy, the first player can't achieve some goal on a smaller-sized tiles, and by arranging these tiles on the infinite-sized board, one can demonstrate that the second player can always retain a draw.

At the present time, 6-in-a-row game and 7-in-a-row game are still open questions. But from 8-in-a-row and onwards the second player is able to draw the game[9].

2.2 Variants of k-in-a-row Games

Variants of k-in-a-row games have been proposed and investigated, one of which is k-in-a-row game on higher-dimensional board, another is to play

Table 2.1. Game values of k-in-a-row games

Game	Game Value
m,n,k-games($k = 1, 2$)	Win for the first player
3,3,3-games(Tic-Tac-Toe)	Draw
m,n,3-games($m \geq 4, n \geq 3$)	Win for the first player
m,4,4-games($m \leq 8$)	Draw
m,n,4-games($m \leq 5, n \leq 5$)	Draw
m,n,4-games($m \geq 6, n \geq 5$)	Win for the first player
m,n,5-games($m \leq 6, n \leq 6$)	Draw
15,15,5-game(Go-Moku)	Win for the first player
m,n,6-games	Unknown
m,n,7-games	Unknown
m,n,k-games($k \geq 8$)	Draw

multiple number of stones on each turn.

2.2.1 Higher-Dimensional Boards

Qubic is a 4-in-a-row game played on a $4 \times 4 \times 4$ cube. Different from the game Connect-Four, no gravity conditions apply. In 1980 Patashink weakly solved the game. Later, Allis and Schoo weakly solved Qubic again, confirming Patashnik's result that Qubic is a first-player win[11]. For higher dimensions, we turn to the Hales-Jewett theorem[12].

The Hales-Jewett theorem is a fundamental result in the field of Ramsey theory. An informal geometric statement of the theorem is that for any positive integers k and c there is a number h such that if the cells of a h-dimensional $k \times k \times k \times \cdots \times k$ are colored with c colors, there must exist one row, column, or diagonal of length n all of whose cells have the same color. In other words the higher dimensional, multi-player, k-in-a-row game cannot end in a draw. By the strategy stealing argument, one can conclude that if two players play alternately, then the first player has a winning strategy when h is sufficiently large. Furthermore explorations on k-in-a-row game on different dimensions and geometric spaces have been conducted by Solomon and Hales[13].

2.2.2 The $Connect(m, n, k, p, q)$ Family

In 2005, Wu and Huang introduced the $Connect(m, n, k, p, q)$ family of k-in-a-row games, where two players place p stones in each turn on an $m \times n$ board except for that the first player places q stones for the first move, and the player who gets k consecutive stones wins. Under this context, Go-Moku is a $Connect(19, 19, 5, 1, 1)$ game. The major difference from k-in-a-row games is that $Connect(m, n, k, p, q)$ has an extra parameter q which significantly influences its fairness. It is demonstrated that $p = 2q$ is a necessary condition for fairness, where one player always has q more stones than the other after making each move, and initial breakaway does not favor the second player. In particular, the game of Connect6 is the game $Connect(19, 19, 6, 2, 1)$. Connect6 is argued not to be definitely unfair, monotonically unfair, and empirically unfair, but the final and formal judgement is yet to be given[3][4].

Further studies are conducted on the family class of $Connect(\infty, \infty, k, p, p)$ games. Chiang et al.[14] demonstrated that, for $k_{draw}(p) = 11$ and for all $p \geq 3$, $k_{draw}(p) = 3p + 3d + 8$ with d as a logarithmic function of p, $Connect(\infty, \infty, k_{draw}(p), p)$ is drawn.

2.3 The Complexity of Connect6

Connect6 is originally stated as a game played on infinite board, thus making the state-space and game-tree complexities infinite too. But in practice, games are played on a 19 × 19 Go board, therefore complexity analysis is typically done on $Connect(19, 19, 6, 2, 1)$.

The state-space complexity of $Connect(19, 19, 6, 2, 1)$ is about 10^{172}, roughly the same as that in Go. The averaged game length is 30, and the number of choices for one move is about $(300 \times 300/2)$, therefore the game-tree complexity is about $(300 \times 300/2)^{30}$ which approximates to 10^{140}[3][4]. Table 2.2 shows the complexities of various games.

Table 2.2. Complexities of various games

Game	State-space complexity	Game-tree complexity
Checkers	10^{21}	10^{31}
Chess	10^{46}	10^{123}
Chinese Chess	10^{48}	10^{150}
Connect6	10^{172}	10^{140}
Connect-Four	10^{14}	10^{21}
Go(19 × 19)	10^{172}	10^{360}
Go-Moku(15 × 15)	10^{105}	10^{70}
Hex	10^{57}	10^{98}
Othello	10^{28}	10^{58}
Qubic	10^{30}	10^{34}
Renju(15 × 15)	10^{105}	10^{70}
Shogi	10^{71}	10^{226}

Chapter 3

A Defensive Strategy for Connect6

A defensive strategy for Connect6 is presented in this chapter. Section 3.1 points out a significant property and some observations in Connect6. Section 3.2 investigates the influence of a stone on the board, and then an evaluation scheme is proposed, and a comparison with current techniques is given. Section 3.3 introduces a strategy based on these observations and results. Experimental results are given in Section 3.4.

3.1 Locality in Connect6

A breakaway move is to place stones far away from the major field of battle, where most stones have been placed. An initial breakaway move is a breakaway move made at the first move of the second player. If the first player does not follow the second player, and continues to play near the initial stone, it is effectively a $Connect(\infty, \infty, 6, 2, 3)$ game in that local area. It is shown that $Connect(\infty, \infty, 6, 2, 3)$ is a win for the first player, therefore implying that initial breakaway may not benefit the second player in any way, and might even give the first player a huge advantage[3].

From many games, it can be observed that breakaway moves do seldom occur and are rarely beneficial for the side that applies it. An informal explanation is that breakaway moves gives the opponent the initiative and significant advantage, and effectively transpose the game to another unfavorable *Connect*(m, n, k, p, q) in that local area of the board. The previous argument concerning the initial breakaway is an example. Therefore, the distance between the stones played in each move and those that are already on the board can not be too large, showing that Connect6 has high locality. Thus, despite the huge complexity of Connect6, its massive branching factor can be effectively lowered by filtering breakaway moves (i.e. moves that are too far from the existing stones on the board) from the candidate move list.

A natural question arises: exactly how far away from the major battle field should a move be characterized as a breakaway move? There are not any formal answers or statistical figures available yet.

For naming the openings, we basically follow the DIF naming system, authored by Wen-Jing Hsu[15]. It is already known and verified that the openings TX-H9, TX-H8, XX-J6, and I7-G9 are instant wins for black[15]. Observing these openings, it can be discovered the same characteristic of all of them is that white distanced its stones from black in its first move. For both XX-J6 and I7-G9, shown in Figure 3.1, the white stones are obviously placed too far away from the black stone.

As for TX-H9 and TX-H8, Shown in Figure 3.2, although the white stones are placed nearer to the black stone, especially TX-H9 in which both stones are placed only with the geometric distance of $\sqrt{5}$ to the black stone, black can still claim victory. Another major characteristic is that both stones are cluttered together on the same side of the black stone. If the white stones are more spread out on different sides of the black stone with the same distance to limit black's freedom, whether black is still able to win is

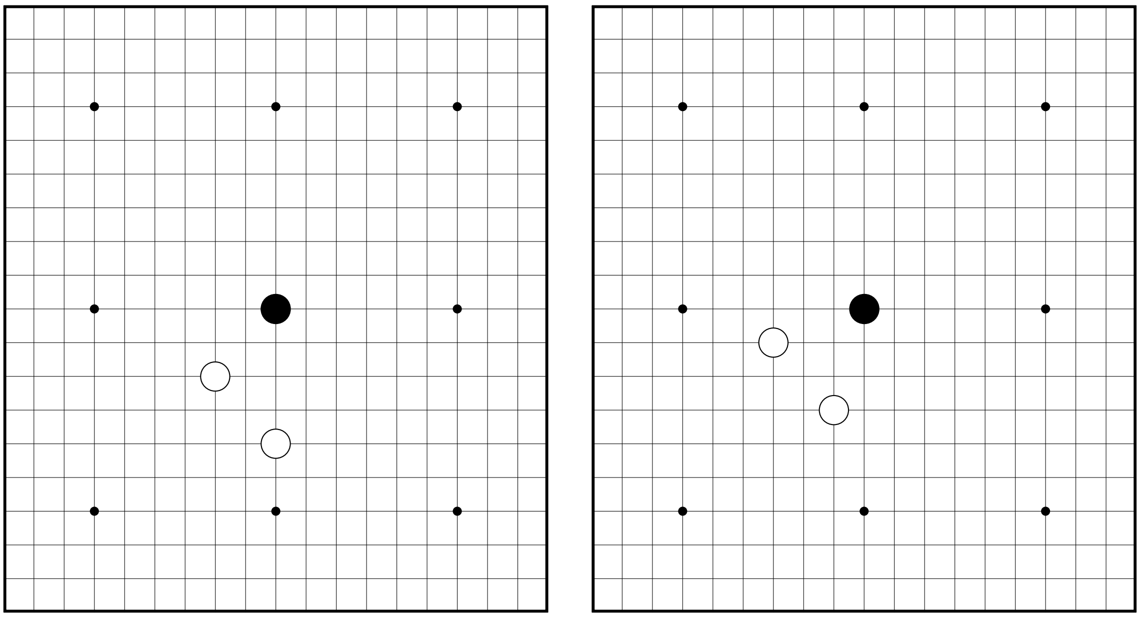
Figure 3.1. The openings XX-J6 (left) and I7-G9 (right)

uncertain. This possibility makes it difficult to determine whether distance is the key contributing factor to the loss of white in TX-H9 and TX-H8, but it is still a significant characteristic of both openings.

From these four openings, we can observe that white doesn't need a great distance to turn a move into a blunder in the early stage of the game. But as the game progresses, if the black and white stones are bonded together and evenly spread in the main battlefield, the chances of forming threats with the existing stones is low. Therefore, a move which is far away from the main battlefield, may play the role of opening up a new combat area rather than jeopardize the position, and thus such move will not be considered as a breakaway move in the context of our previous discussion. Hence in such conditions, the distance that characterize a breakaway move should be increased.

In conclusion, due to the properties of breakaway moves, Connect6 has

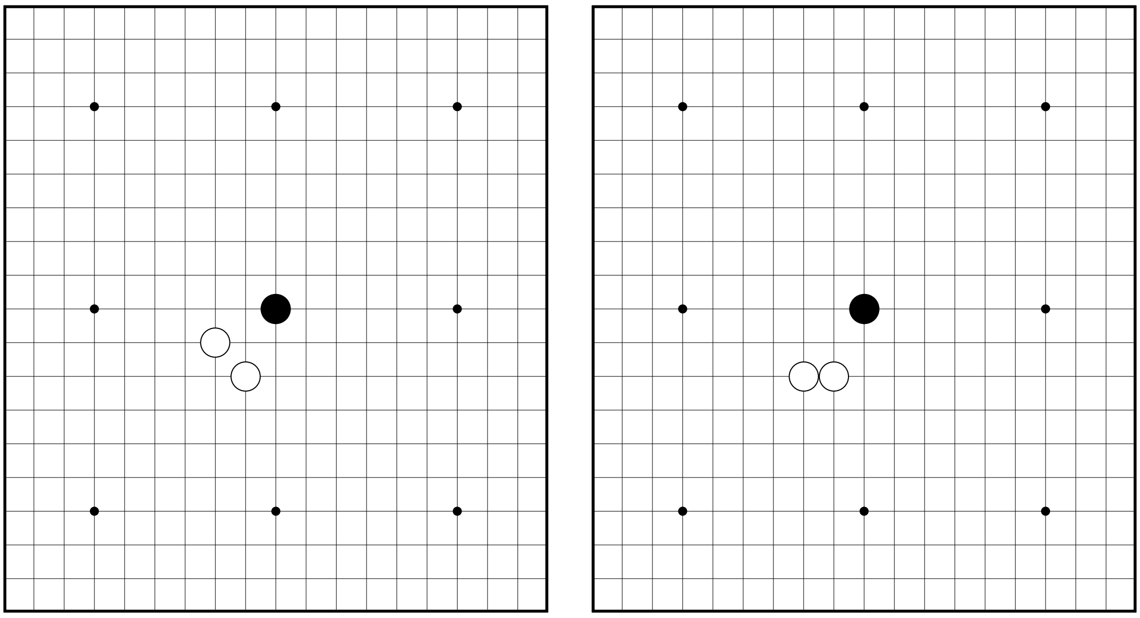

Figure 3.2. The openings TX-H9 (left) and TX-H8 (right).

high locality. But the question of how to characterize a breakaway move is yet to be answered, and the answer may vary from position to position or from stage to stage in the game.

3.2 An Evaluation Scheme

Taking the idea of locality further, we will begin by investigating the influence of a stone on the board, then introduce a new evaluation method based on these observations, and make a comparison to the current evaluation methods that are applied in other Connect6 programs.

3.2.1 Influence of a Stone

Apart from the first move, two stones are placed on the board with each move, and these stones change the development and nature of the previous position. What does the presence of a new stone alter?

A potential line is a line with a length of 6 on the board, with no stones or only a single kind of stone on it. Thus, it is a potential candidate to connect six for one of the players. If there are two kinds of stones on the line, then it can be sure that a winning six will not occur on it. We will regard a line which contains six consecutive stones that win the game as a winning line.

A straight forward and tactical point of view is that a stone played on the board increases the chance of connecting six for one side, or reduces the chance of winning for the opponent. But what is the scale or range of its influence? Instead of viewing the board as a collection of points, we regard it as a collection of potential lines. Therefore, the presence of a stone increases the probability of a potential line, which contains the stone, to become a winning line.

Hence, a stone's influence area should be the area that consist of all the potential lines which contains the stone. The largest possible area of influence is made up of four lines (horizontal, vertical, and two diagonals) of length 11, with the stone in the center as depicted in Figure 3.3.

Other points which are not included in the area are by no means not important at all. Rather, they differ from the included points by meaning. The included points are under more direct influence of the stone, thus may have more tactical meaning, whereas the ones that are not included are less directly related to the stone, and may only retain some strategic importance.

Connectivity is another major issue in Connect6, since if one's stones are sparsely spread across the board, the chance of connecting six is low. If two stones are close to each other, there are less points between them for the

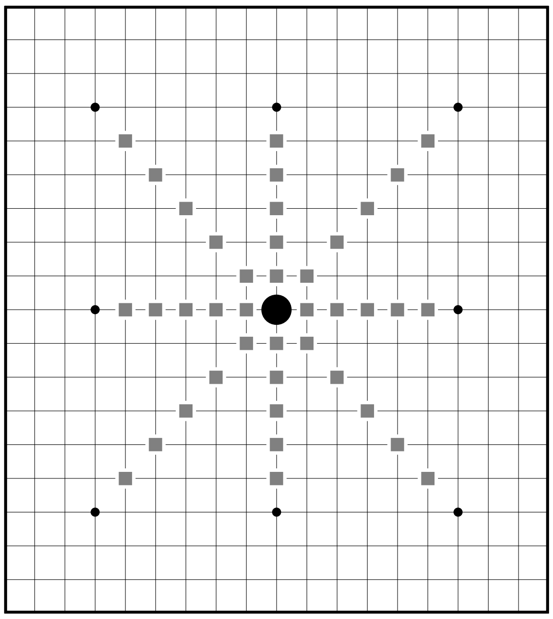

Figure 3.3. Influence of a stone

opponent to cut their connection. Thus, the relation between two stones decreases with increasing distance, and this relation is in a sense formed by the influences of the two stones imposing on each other. It is therefore natural to imply that the influence of a single stone decreases with increasing distance. An simple analogy of this phenomenon would be some forces of nature such as gravity or electromagnetic force, they too decline in "influence" with distance.

3.2.2 Evaluation of a Half-move

We are set to propose a model that will reflect the properties we observed. Some essential questions need to be addressed first.

The influence of a stone should stop when the border or an opponent's stone is encountered, since it can't make any connections with any stones beyond them. Therefore, the length of the lines in the statement of the influence area should be refined to lines with a length of 11 or less, as shown in Figure 3.4.

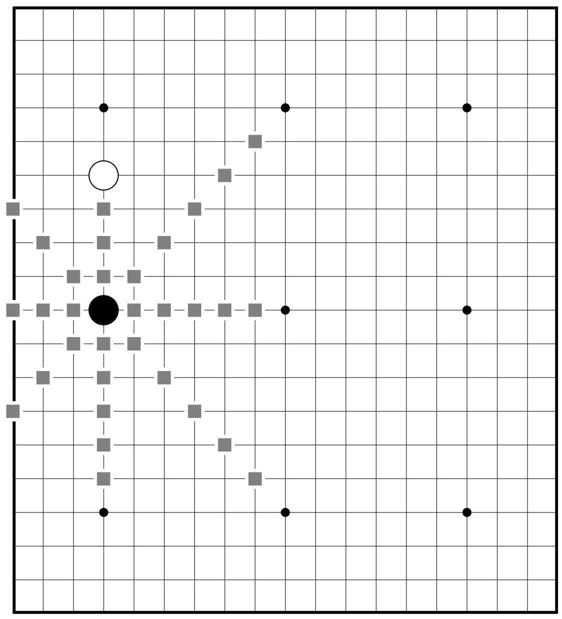

Figure 3.4. The influence area encountering borders or opponent stones

We will not try to take the points that are not in the influence area into consideration, since we believe that their strategic values may vary for different positions, and a perfect model may not exist. Even if it does exist, it may introduce unnecessary complexities.

Earlier we made the analogy to field forces in physics, a same problem also arises in our model, namely the n-body problem. For the sake of simplicity, we simply assume our model acts locally. Only the influence area of a stone or move is considered, therefore avoiding the problem which will arise when "the influence" ripples through each point and onto the entire board.

Two stones are played in a single move in Connect6, a half-move refers to the situation where only one stone is under consideration. Since our observation is of a single stone, our model will be based on the evaluation of a half-move.

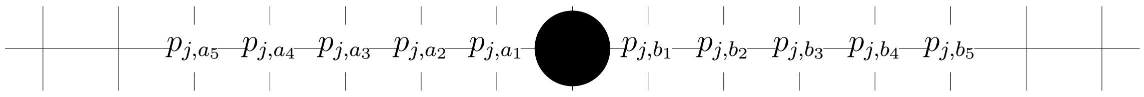

Figure 3.5. Evaluation of a half-move

Suppose that the opponent is playing black, then the evaluation score for a black stone or a half-move is given by

$$E = d_i \sum_{j=1}^{4} (\prod_{k=1}^{5} p_{j,a_k} \prod_{k=1}^{5} p_{j,b_k}).$$

E is the evaluation score of the half-move, reflecting the influence it has. i is the number of directions that have no opponent stones, and d_i is the weighted value in that respect. The value j corresponds to the index of directions to be considered, so there are 4 directions (1 horizontal, 1 vertical,and 2 diagonal). p_{j,a_k} and p_{j,b_k}, where $1 \leq k \leq 5$, are the states of the points, which corresponds to the points marked in Figure 3.5 on each respective line.

The values of p_{j,a_k} and p_{j,b_k} are determined in an orderly decreasing fashion from near to far, i.e. from p_{j,a_1} to p_{j,a_5} and from p_{j,b_1} to p_{j,b_5}. If the point is an empty point, the value ϵ is given, and if the point is occupied by an own stone, then a corresponding weight w_k is given. The values w_k decreases with increasing k. Once the border or a white stone is encountered at a certain k, the remaining values $w_k, w_{k+1}, ..., w_5$ are set to 1.

Note that the values on the same line are combined by using multiplication and the four directions are combined by using addition. This reflects the fact that connectivity has higher priority over directional liberty. Although in some cases, directional liberty may have higher priority, especially in earlier stage of the game, these cases can be addressed by applying the degree parameter d_i.

Algorithm 1 Half-move evaluation Algorithm

```
procedure HMOVE-EVALUATION($position, h - move$)
    $E \leftarrow 0$
    $E_{directional} \leftarrow 1$
    for $j \leftarrow 1, 4$ do                                        ▷ for four directions
        for $l \leftarrow a, b$ do                                    ▷ for each half of a line
            for $k \leftarrow 1, 5$ do                                ▷ for each point
                if $p_{jlk}$ is an opponent's stone or border then
                    break
                else if $p_{jlk}$ is an empty point then
                    $E_{directional} \leftarrow E_{directional} \times \epsilon$
                else if $p_{jlk}$ is an own stone then
                    $E_{directional} \leftarrow E_{directional} \times w_k$
                end if
            end for
            $E \leftarrow E + E_{directional}$
        end for
    end for
    return $E$
end procedure
```

A rough sketch of the entire algorithm is exhibited in Algorithm 1. All that remains is to determine the value of each parameter. A fundamental relation needs to be preserved: a half-move which contains occupied stones in its influence area should not be overpassed by a half-move which only has empty points in its influence area. Please see Figure 3.6 for a depiction. This may occur since empty points also have a weight.

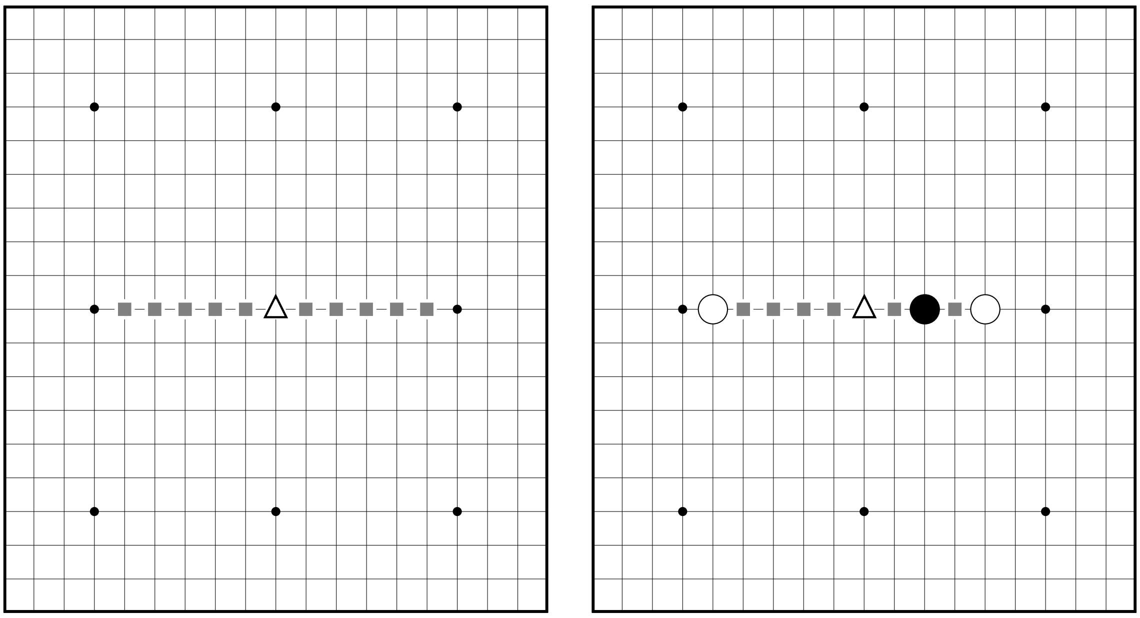
Figure 3.6. Evaluation of the triangle position in the right should be higher than that in the left.

A simple solution would be to let a portion of the weight of a stone be larger than the value of a line that contains only empty points. Therefore, the relation

$$\epsilon^{10} \leq w_l$$

must be held, where $l \leq n \leq 5$. w_n is the weight that is equal to the value of the line with only empty points.

We decided that the weight of a empty point $\epsilon = 2$, then we use it as a base to determine the values of w_k. From experiments and experiences, we determined that $n = 3$, and $w_1 = 2^{12}$, $w_2 = 2^{11}$, $w_3 = 2^{10}$, $w_4 = 2^9$, $w_5 = 2^8$.

The values of degree weights are determined and tuned purely from experiments, which are $d_1 = 1.0$, $d_2 = 1.00000181862$, $d_3 = 1.00000363725$, $d_4 = 1.00000726562$. The values are fairly small, due to the exponential figures of the empty point weight and stone weights, and due to their nature of being mostly adjustments for some positions.

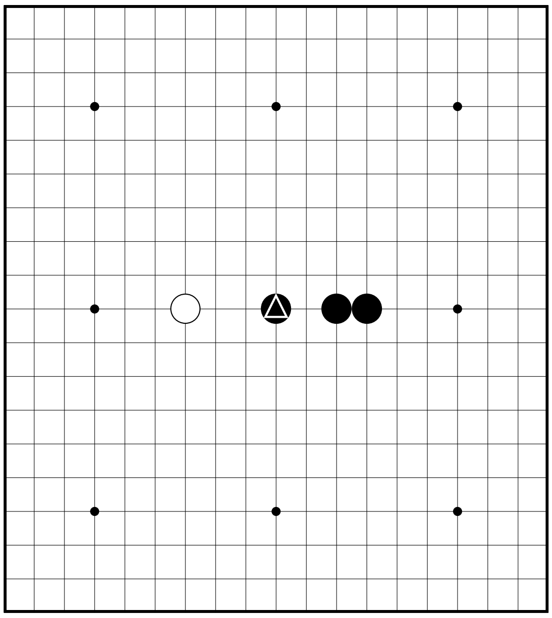

Figure 3.7. Example of evaluation

An example is shown in Figure 3.7, the evaluation value of ▲ is determined by

$$\begin{aligned} E =& d_3 \times [(\epsilon \times \epsilon \times 1 \times 1 \times 1 \times \epsilon \times w_2 \times w_3 \times \epsilon \times \epsilon) + (\epsilon \times \epsilon \times \epsilon \times \epsilon \times \epsilon \times \epsilon \times \epsilon \times \epsilon \times \epsilon \times \epsilon) \\ &+ (\epsilon \times \epsilon \times \epsilon \times \epsilon \times \epsilon \times \epsilon \times \epsilon \times \epsilon \times \epsilon \times \epsilon) + (\epsilon \times \epsilon \times \epsilon \times \epsilon \times \epsilon \times \epsilon \times \epsilon \times \epsilon \times \epsilon \times \epsilon)] \\ =& 1.00000363725 \times [((1 \times 1 \times 1 \times 2 \times 2) \times (2 \times 2^{11} \times 2^{10} \times 2 \times 2)) + 2^{10} + 2^{10} + 2^{10}]. \end{aligned}$$

There is one opponent stone in the horizontal direction, thus the degree parameter d_3 is applied. Apart from the horizontal direction, there are no stones in the vertical and diagonal directions, therefore their values are the product of 10 empty point weights ϵ, that is 2^{10}. The value determined by the horizontal direction is thus $(1 \times 1 \times 1 \times 2 \times 2) \times (2 \times 2^{11} \times 2^{10} \times 2 \times 2)$.

3.2.3 Comparison to Current Evaluation Techniques

The evaluation schemes used in many programs are mostly pattern-based, for different configurations of stones on a line, a score is assigned[7] [16]. Then the scores of the four directions are combined (mostly by addition) to make up the evaluation score.

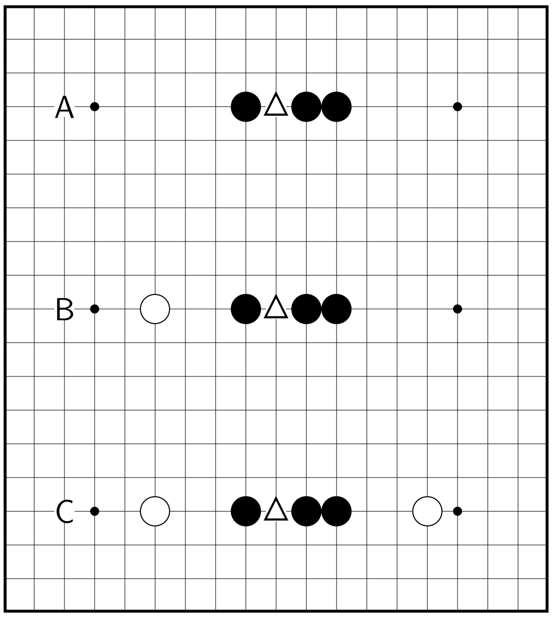

Figure 3.8. Comparison of different evaluation scheme

For example, if black places a stone at the triangle positions in the patterns A, B, and C in Figure 3.8, the patterns they create are all usually classified as a Live4 pattern, and its main characteristic is that white needs two stones to defend the threat. Hence, the half-moves at the triangle positions have the same value in most of current program's evaluations.

But with our proposed evaluation scheme, the three half-moves have different scores. Pattern A has the highest score, pattern B is next, and pattern C is the lowest. If the half-move haves been played, they would create the same amount of threat. But due to the different number of empty points on the examined directions, the evaluation values varies since empty points also have a weight. And this reflects the fact that although they have the same tactical meaning (threats), they are distinct in strategic context, namely liberty or space.

Therefore, the proposed evaluation scheme is much more delicate, and may distinguish the strategic importance of the same pattern. It is more strategic sensitive than the traditional approaches, but retains the same tactical sensitivity at the same time.

3.3 A Defensive Strategy

To defend from opponent's attack, a way is to lower the influence of opponent's stone on the board. A point which has a high evaluation score from the opponent point of view, means that the opponent's stone can impose a large influence. Therefore, if we play at the points where the opponent has a high evaluation score, we may cut off the connection of the opponent's stones and limit his liberty or space effectively.

The branching factor in Connect6 is large, due to the fact that two stones are placed in each move, and the board is very large. Although the property of high locality can effectively reduce the number of candidate moves, further considerations are still called for.

We will make an assumption that the opponent has a purpose with every moves. If the opponent's intention is to make some abstract connections to other stones to produce an attacking pattern, it would be best to cut off these connections as soon as possible. That is by "following" the opponent's move to wherever he goes, and cut off potentially dangerous connections, one can minimize the chance that the opponent can make an attack. Of course, if the opponent's purpose is to defend, we may waste some stones on unnecessary defense if we follow this strategy. But then again, our purpose is to make a solid defense, thus the waste of stones is tolerable.

So we only consider the half-moves (or points) that are in the 5×5 square with the last two stones that the opponent played in the center as candidate half-moves. Therefore, at most 48 points or half-moves are considered. Figure 3.9 shows a position with white to move, and candidate half-moves are marked by filled squares.

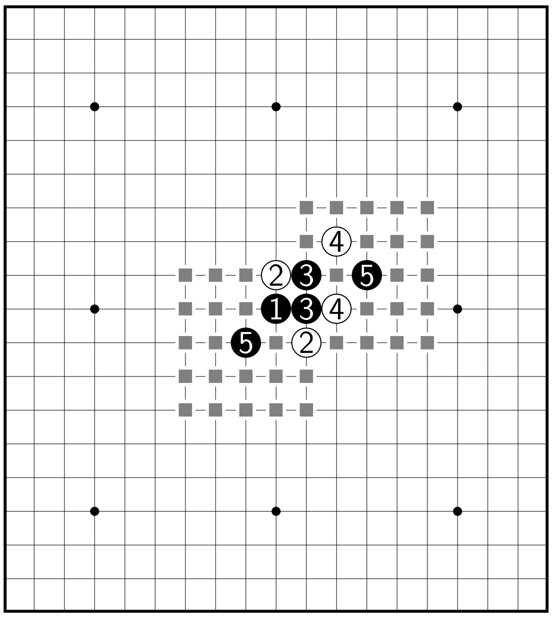

Figure 3.9. Candidate half-moves (white to move)

The defensive strategy consist of two phases. First, it generates a half-move list and sorts the entries according to the value which is given by the evaluation scheme proposed earlier. Then it plays the half-move with the highest value, and do the same for the second half-move.

The overview of the proposed strategy is given in Figure 3.10. Note that it is only a simple greedy method based on the proposed evaluation, and no search technique of any kind is applied.

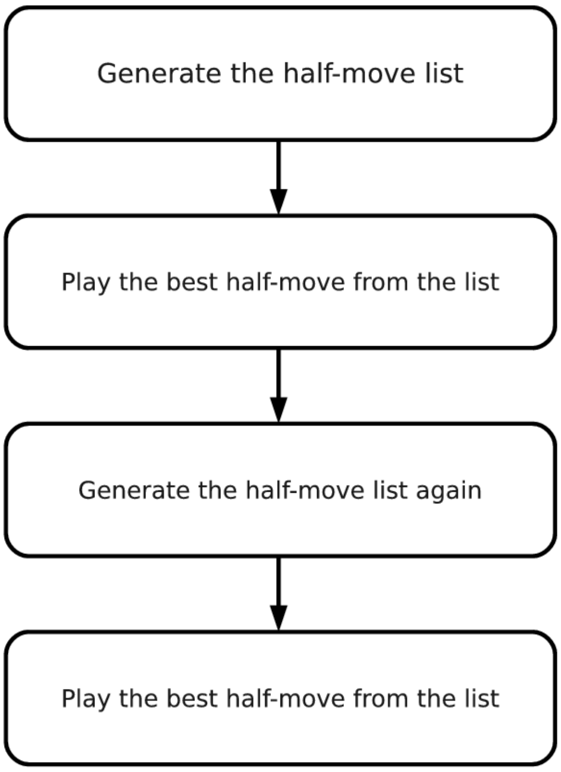
Generate the half-move list
Play the best half-move from the list
Generate the half-move list again
Play the best half-move from the list

Figure 3.10. Flowchart of a defensive strategy

3.4 Experimental Results

To test how effective our defensive strategy is, we use it to play against two famous Connect6 programs:

- **NCTU6**: NCTU6 is a program developed by the Internet Application

Laboratory led by Professor I-Chen Wu at the National Chiao Tung University. It won a gold medal in the 11th Computer Olympiad and another gold medal in the 13th Computer Olympiad. The version we did our experiments on is the 2006 public release version 1.0, with playing strength set to level 3 (the maximum strength is level 5).

- **X6**: X6 was developed by Shih-Yuan Liou and Professor Shi-Jim Yen at the Artificial Intelligence Lab. of the National Dong Hwa University. It won a silver medal in the 11th Computer Olympiad and a gold medal in the 12th Computer Olympiad. Experiments are conducted with version 1.4.0f, with playing strength set to 9 (Kill-Defend Search depth was set to 11, and 100 seconds to timeout)

Our program was run on a machine with AMD64 3000+ and 1GB RAM under Ubuntu Linux 9.04, kernel version 2.26.1. The opponent programs were run on a machine with Intel Core2 Duo 1.66GHz and 1 GB RAM under Windows XP Professional Service Pack 2. Ten games are played with each program. Our program plays black in the first 5 games and plays white in the last 5 games. The results are listed in Table 3.1 and 3.2. In the tables, D, W, and L, means we draw, win and lose, respectively. Game length is the number of moves that are played. Although there are still empty points available on the board, we declare the game drawn around move 120, since the space left on the board is insufficient for either side to get a six.

Table 3.1. Results of our defensive strategy against NCTU6 (level3)

Game	1	2	3	4	5	6	7	8	9	10
Result	D	D	D	D	D	D	D	D	D	D
Game Length	121	120	119	124	120	120	123	121	120	123

We can see that our program can draw perfectly against NCTU6 (level3). Our program can draw 60% of the games against X6 in its full power without

Table 3.2. Results of our defensive strategy against X6

Game	1	2	3	4	5	6	7	8	9	10
Result	L	D	D	D	L	L	D	D	D	L
Game Length	31	121	123	131	47	18	118	128	120	34

using any kind of search, showing that our defensive strategy is very effective. However, it still loses 40% of the games, so there is still room for further improvement. Some selected games are shown in Appendix B.

Chapter 4

Threat-Space Search

Threat-space search is an important method in Connect6. Section 4.1 gives an introduction to the concept of threats in Connect6. Threat-space search is presented in section 4.2. Some implementation techniques we applied is presented in Section 4.3. Section 4.4 combines it with the defensive strategy proposed earlier. Finally, an analysis of the experimental results is presented in Section 4.5.

4.1 Threat-Based Strategy

In Connect6, a player is said to have t threats, if and only if the opponent needs to place t stones to prevent the player to connect six and win the game[3].

In Figure 4.1, pattern A is an example of one threat, pattern B is an example of two threats or double threats, and pattern C is an example of three threats. Therefore, white needs at least one stone to stop the threat imposed in pattern A, and two stones to defend the threat in pattern B. But in pattern C, white needs at least three stones to stop black from winning.

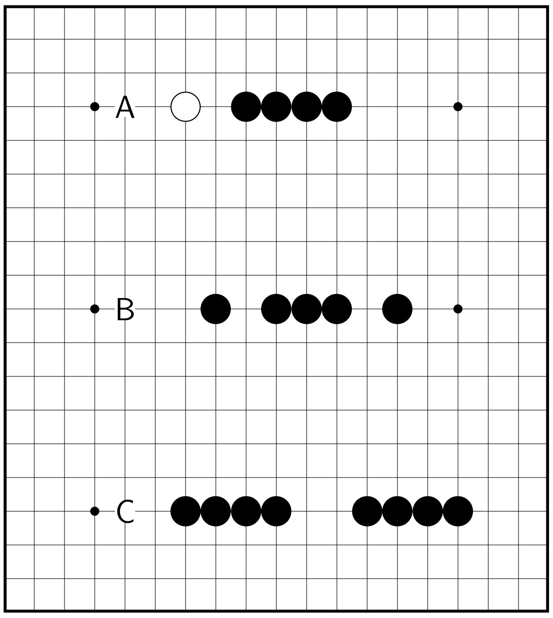

Figure 4.1. Example of threats in Connect 6

Since a player can only place two stones on each move, white cannot stop black from connecting six, therefore loses the game. Hence, the winning strategy in Connect6 is to have at least three threats.

4.2 Threat-space Search

Threat-space search was first proposed by Allis[6], and it was used as an evaluation function accompanying proof number search to solve Go-Moku. The main idea of threat-space search is similar to that of quiescence search, in which only forced moves are extended and explored. In Connect6, the forced moves are the moves that can create at least a threat. The class of two threats is of special attention since it forces the opponent to use both stones to defend, depriving him of the opportunity to create attacking chances or to carry out strategic ideas of his own. In some positions there exists a series of two threats that leads to a three or more threats. Therefore one can claim victory if such a series is found.

Although the branching factor is significantly lower than the usual uniform search, it is still undesirable to explore every possible moves. Therefore, a way to simplify and lower the complexity of threat-space search is not to consider the defensive moves separately, but to consider them all at the once. Conservative defense is to play all defensive moves at the same time. An example is shown in Figure 4.2.

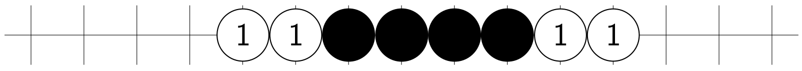

Figure 4.2. Conservative defense

Conservative defense effectively transforms threat-space search into a single agent search. Since for every threat pattern, all possible defensive moves are played at the same time, therefore a pattern combined with its respective defensive moves can be combined into a single pre-defined pattern.

However, by applying conservative defense, the defense side is presumed to play all possible defenses at the same time, and thus more than two stones could be placed on each move. In Figure 4.2, the white's conservative defense plays four stones. Therefore, the set of solutions found by the threat-space search that applied conservative defense can only be a subset of the set of true solutions, since the extra defense stones can cause an early search failure.

In some variations, the forcing series consists of the class of two threats mixed with the class of one theat. But we only consider the series that contains two or more threats for the sake of simplicity. Hence the move generation procedure only generates the moves that can create two or more threats. Algorithm 2 sketches the idea of a conventional threat-space search.

Algorithm 2 Threat Space Search Algorithm

```
procedure THREAT-SPACE-SEARCH($attacker$)
    if defender has a win or a threat then
        return $FAIL$
    end if
    if own-side can connect six or create 3 threats then
        return $SUCCESS$
    end if
    $Movegen(list)$
    if $list$ is empty then
        return $FAIL$
    end if
    $result \leftarrow FAIL$
    for Every move in $list$ do
        $Move()$
        $result \leftarrow Threat\text{-}Space\text{-}Search(attacker)$
        $Undo()$
        if $result = SUCCESS$ then
            return $SUCCESS$
        end if
    end for
    return $FAIL$
end procedure
```

4.3 Implementation Issues

Both accuracy and efficiency are of utmost importance in the implementation of threat-space search. A quick method of identifying whether a move creates a threat is applied. The sliding window algorithm is used for defining the number of threats in a pattern, and the defending moves can also be defined from it. As for efficiency, we implement a pattern table to store the threat number of every pattern.

4.3.1 Defining Threats in Linear Patterns

It is essential to correctly decide the number of threats in a pattern, since the correctness of threat-space search depends upon it. A formal way of defining a pattern's threat is to apply the sliding window algorithm[3][7], as shown in Algorithm 3.

Algorithm 3 Sliding Window Algorithm

```
procedure SLIDING-WINDOW(lpattern)
    for slide window on lpattern left to right do        ▷ window is of size 6
        if full with attacker stones then
            $threat \leftarrow \infty$
            break
        else if no marked points or defender and attacker stones$\geq 4$ then
            mark all empty squares
            $threat \leftarrow threat + 1$
        end if
    end for
    return threat
end procedure
```

The *window* is essentially a potential line to connect six. If *window* is full with the attacker's stones, it means that the attacker has connected six and won, therefore we define the threat number to be ∞ in this situation. The marking of the empty points in Algorithm 3 makes sure that a threat is not counted twice due to overlapping.

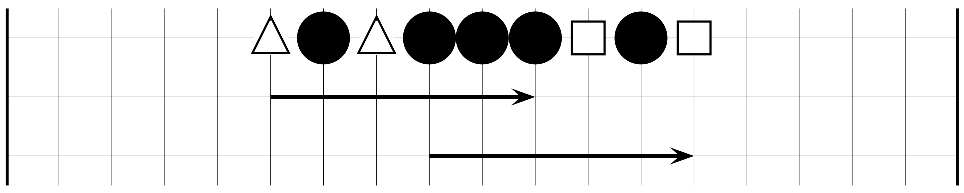

Figure 4.3. Example of Sliding Window Algorithm

Figure 4.3 is an example of the algorithm being applied. The windows that increase the number of threats are represented by arrow lines. In this example, the pattern is a two threat pattern. The triangle positions are the empty points that are marked in the first window, and the squares are the ones that are marked in the second window.

4.3.2 Finding Defensive Moves

The definition of defensive move is also of importance. Apart from correctness, since conservative defense is applied, the accuracy of the search is also affected. Therefore, there is also a need for formally defining the defensive moves of a linear pattern.

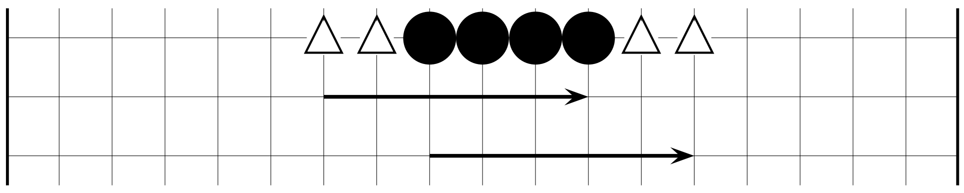

Figure 4.4. Defining defensive moves by sliding window algorithm

In the sliding window algorithm, all empty points in a window are marked if the window contains at least 4 stones of the attacker. These empty points can also be regarded as the points to defend, since if the defender can place a stone in one of those empty points, the defender can liquidate the threat, and stop the attacker from winning. Hence, by applying the sliding window algorithm, defensive moves can also be defined. Figure 4.4 is an example where the algorithm is applied to a Live4 pattern. Again, the arrow lines indicate the windows that threats are counted. In this example, there are two threats. The points that are marked with a triangle are the empty points that are marked during counting. Notice that they also correspond to the defending points mentioned earlier in Figure 4.2. Therefore the marked empty points can also be regarded as defensive moves.

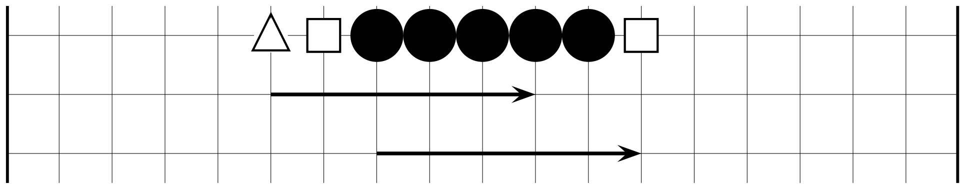

Figure 4.5. Redundant defensive moves: Live5 pattern

But some complications arise as well. The main downside of conservative defensive moves is that the number of defender's stones grows rapidly, since it plays all possible defensive moves at the same time. By the sliding window algorithm, some patterns may be "over-protected", i.e. some defensive moves are redundant. This kind of redundancy may severely impact the accuracy of the search, and cause the search to miss some winning series.

Figure 4.5 displays a Live5 pattern in which one such redundancy occurs. The marked triangle and squares are the defensive moves decided by the algorithm. But the triangle-marked move is not necessary, it is even not a valid defensive move. No matter which of the points marked with a square is combined with it, it won't be sufficient to defend the threat.

Another example is given in Figure 4.6. The pattern is a Dead4, and has only one threat. The points marked with a triangle are the defensive moves, and again the defensive move between the white and black stone is redundant. Another problem is that point A is also a valid defensive move, but it is excluded.

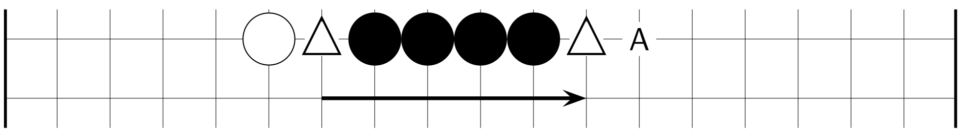

Figure 4.6. Redundant defensive moves: Dead4 pattern

Notice that if we slide the window from right to left, instead of from left to right, the defensive moves would be derived correctly as shown in Figure 4.7.

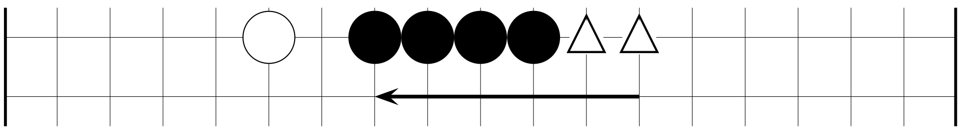

Figure 4.7. Reverse slide window: Dead4 pattern

It is clear that the problem is caused by the fact that the window slides only in one direction. Therefore, a simple solution would be to slide the window twice, that is, once in each direction, and the empty points that are marked twice are the essential defensive moves. Algorithm 4 describes this simple solution.

Even with the revision, it is still not a hundred percent accurate, since there are still corrections to be made. For example, the defensive move A in Figure 4.6 will still be eluded. But these kinds of patterns are few and can be treated as special cases.

An effort of enhancement is made to increase the accuracy of the search. We always play the inner defensive moves, and the outer defensive moves are played under certain conditions. Similar methods are also applied in some of

Algorithm 4 Defensive Sliding Window Algorithm

```
procedure DEFENSIVE-SLIDE-WINDOW($lpattern$)
    for slide $window$ on $lpattern$ left to right do        ▷ $window$ is of size 6
        if no marked points or defender and attacker stones$\geq 4$ then
            for all $empty - point_i$ in $window$ do
                $mcounter_i \leftarrow mcounter_i + 1$
            end for
        end if
    end for
    for slide $window$ on $lpattern$ right to left do
        if no marked points or defender and attacker stones$\geq 4$ then
            for all $empty - point_i$ in $window$ do
                $mcounter_i \leftarrow mcounter_i + 1$
            end for
        end if
    end for
end procedure        ▷ The output are the points with $mcounter_i \geq 2$
```

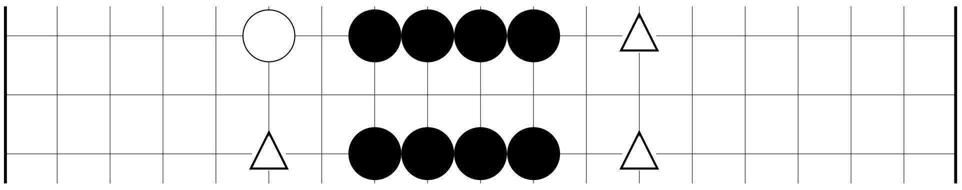
Figure 4.8. Outer defensive moves

today's Connect6 programs[16]. The points marked with a triangle are the outer defensive moves in Figure 4.8. The outer defensive moves are played only if there are two or more attacking stones in the other three directions.

The rationale is that it is better to leave less room for the opponent if possible, and playing the outer defensive moves means leaving more space for the attacker. Therefore, the condition specified earlier reflects the assumption that outer defensive moves are played only if the opponent may have the chance to achieve multiple goals.

4.3.3 Linear Pattern Table

A pattern table is applied in our implementation of threat-space search in order to save computing time. Each entry in the table contains a key and the number of threats. The defensive moves are not saved in the table, and are computed during run-time. All possible configurations of stones on a line of length 11 are pre-processed, and the numbers of threats of these patterns are stored.

Full hashing is used, the hash function maps each configuration into a base-3 number ranging from 0 to $3^{11} - 1$ acting as a hash key. The weights of the corresponding positions on the line are shown in Figure 4.9.

If a point is empty, its value is 0. If a point has a white stone, its value is 1, and if a point has a black stone, its value is 2. The point value is multiplied by its respective positional weight and then all weighted values are summed

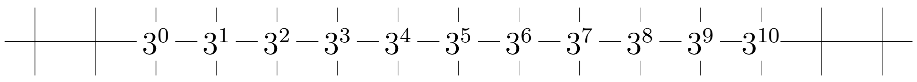

Figure 4.9. Positional weights of hash function

up to get the hash key. Borders are treated as the opponent's stones.

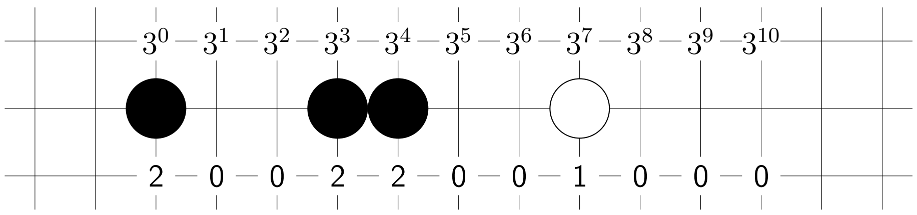

Figure 4.10. Example of hash key

An example is given in Figure 4.10. The positional weights are given above the pattern, and the respective weights of the stone or empty point are shown below. Hence the hash key is calculated as follows:

$$HashKey = 2 \times 3^0 + 2 \times 3^3 + 2 \times 3^4 + 1 \times 3^7.$$

4.3.4 Combining Information

The pattern table is applied to retrieve the number of threats a stone can create in one direction. Since two stones are played in a single move, possible errors may occur when the two stones are placed on a single line, for they both may "contribute" to the same threat pattern. Thus the threat may be erroneously counted twice.

An example is pattern A in Figure 4.11, for the stones a and b they both retrieve a threat count of 1, therefore a move consists of stones a and b will be wrongly interpreted as a move that creates two threats if we only simply add up the numbers of threats. Pattern B is another example, which the pattern will be interpreted as a four threats.

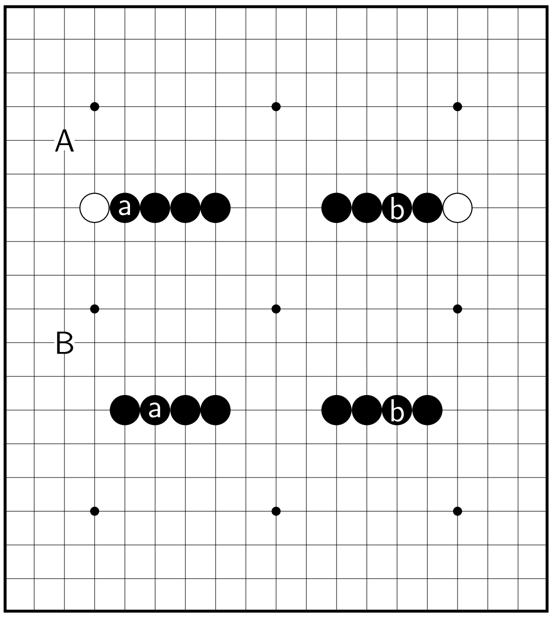

Figure 4.11. Errors when two stones align

A simple scheme may resolve the problem:

1. If both stones are not on the same line, simply add the two numbers of threats up.
2. If both stones are on the same line, and there are opposing stones between them, add the two numbers of threats together.
3. If both stones are on the same line, but without opposing stones between them, then
   (a) take only half of their threat sum into account, if their distance is less than or equal to 6.
   (b) subtract one from the sum of their threats, if their distance is between 7 and 11, inclusive.
   (c) take the sum of their threats, if their distance is 12 or more.

If both stones are not on the same line or there are opposing stones between them, the threats they create are independent, hence the sum of their threats are the threats created by this full move. The correctness of the sum of threats will only be affected if it will lead to a wrong interpretation of one threat and two threats. If the sum of the threats is greater than or equal to 3, like pattern B in Figure 4.11, it won't matter if it is wrong, since this kind of threat greater than 3 will end the game. Therefore, we only need to deal with the patterns which the sum of the threats is 2, because if it is wrongly computed as a two threats, and in reality it is a single threat, the opponent actually won't be forced to move in defense, and thus it may mislead the search.

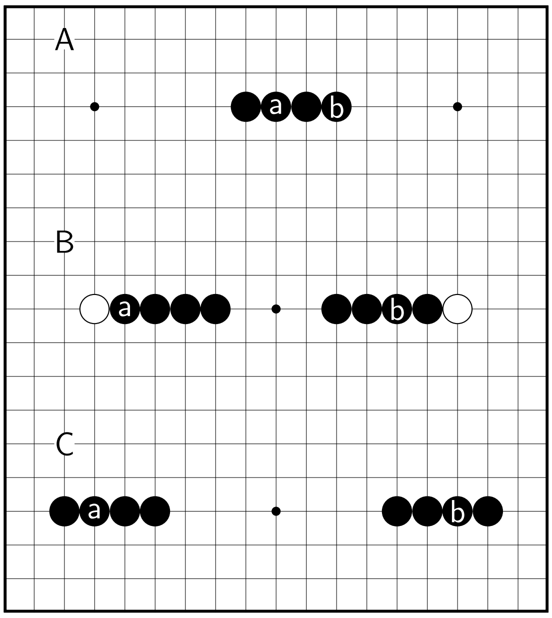

Figure 4.12. Example patterns for the correction scheme

Patterns A, B, and C in Figure 4.12 are examples of the above 3 cases (a), (b), and (c). Suppose the move consists of the stones a and b. In pattern A, the distance between a and b is 2, which is less than 6, and from the pattern table, both of them have two threats. By the correction scheme, only half of their threat sum is set to be the pattern's threat number , that is 2. For pattern B, the distance between a and b is 9, which is between 7 and 11, and from the table we can get that both of them are single threat, thus the sum of threats is 2. By the correction scheme, we need to subtract 1 from the sum, and thus the pattern's threat number is 1. Finally, the distance between a and b in pattern C is 12, and their individual numbers of threats are 2, making the threat sum 4. By the correction scheme, the threat sum is the pattern's threat number, which is 4.

4.4 Combining Search with Defensive Strategy

There is still a certain percentage of the positions that the static defensive strategy proposed in the previous chapter will fail. No matter how fine the parameters are tuned, due to inaccuracies in the model or any other reason, these kinds of positions may always exist.

Another drawback of only applying the defensive strategy is that, unless the opponent blunders badly, one can never win a game, even though there may exist a winning move.

Therefore, to address these problems, the combination of the defensive strategy and a threat-space search is proposed as a solution. Please see Figure 4.13 for a depiction.

A defensive move is always derived according to the defensive strategy, and it is verified by the threat-space search. If the verification fails, and the opponent has an immediate threat to win, another defensive move is chosen and verified again until it can defend the opponent's threat or the number of alternative moves exceeds a threshold value. If no effective defensive move can be found, then the last defensive move is set to be the move decided by

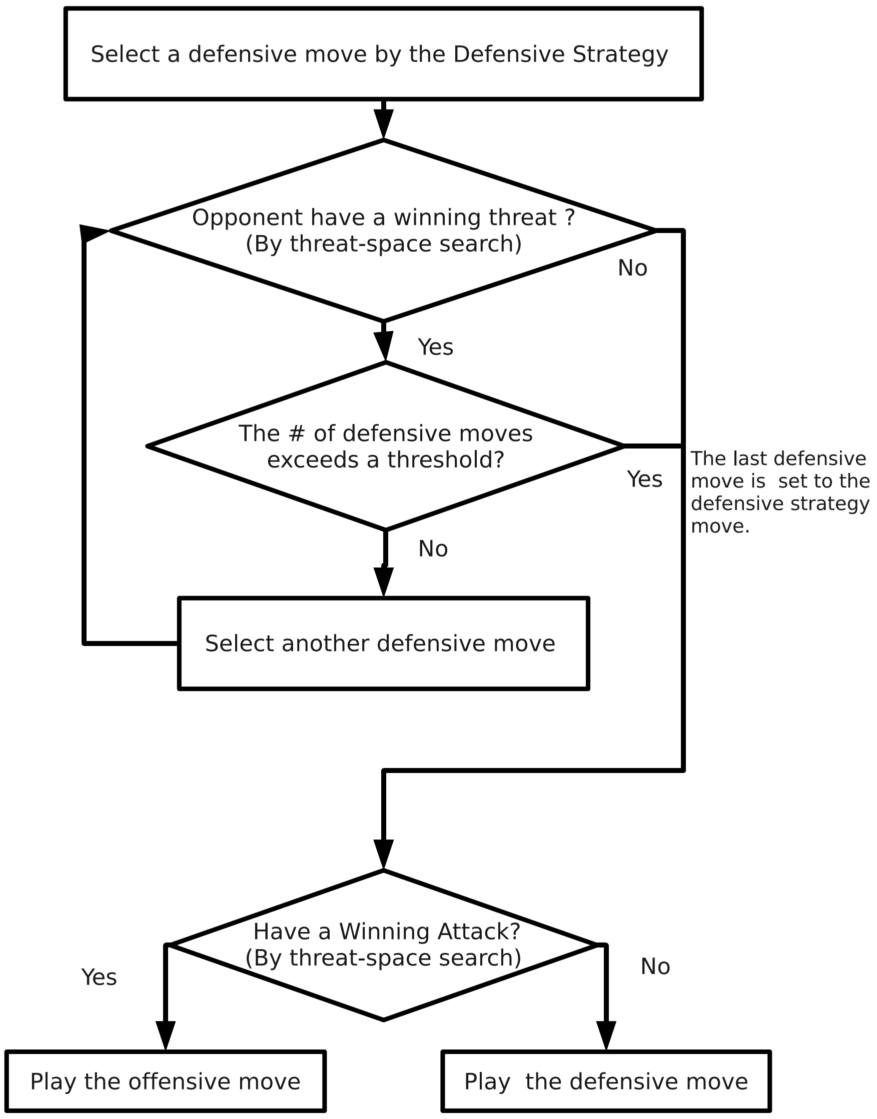

Figure 4.13. Defensive strategy combined with threat-space search

the defensive strategy.

Finally an offensive threat-space search is to be carried out. If there is a winning move, then the winning move is played, else the defensive move is played. Note that we still don't attempt to create any attacking opportunities or winning threats.

The only situation to attack is that a winning sequence is found by the threat-space search, whereas victory is sure to come. The threat-space search mostly acts as a verifier to make up for the weakness of the static defensive strategy.

The defensive moves are ordered in decreasing order according to the scores given by the evaluation scheme given in the previous chapter, and at most 50 defensive moves are considered.

Of course this is not an optimal arrangement of the two modules, since it would be better to do the winning-attack search first. But before starting the search for a winning attack sequence, a check is needed to be performed first to make sure that the opponent doesn't have a winning threat in the current position. This essentially splits the defensive part of the architecture into two parts, with the attacking module in the middle, and thus produces some complications. Since our purpose is to experiment the validity of the proposed strategy, and to verify the performance enhancement when it is combined with a threat-space search, we will avoid such complications and stick with this simple architecture shown in Figure 4.13.

4.5 Experimental Results

Using the ideas mentioned in this chapter, we developed a program to play against the following two programs:

- **X6**: The program was developed by Shih-Yuan Liou and Professor

Shi-Jim Yen. It was introduced in the previous chapter. Experiments are conducted with version 1.4.0f, with playing strength set to 9 (Kill-Defend Search depth was set to 11, and 100 seconds to timeout)

- **MeinStein**: The program was written by Theo van der Storm. Mr. van der Storm passed away in January 2009, and it was subsequently maintained by Jan Krabbenbos. It won a silver medal in the 12th Computer Olympiad, and another in the 14th Computer Olympiad. After the 14th Computer Olympiad, the source code was released to the public domain in memory of Mr. van der Storm. The program was written in Java, and incorporated techniques such as $\alpha\beta$-search and quiescence search.

Our program was run on a machine with AMD64 3000+ and 1GB of RAM under Ubuntu Linux 9.04, kernel version 2.26.1. The opponent programs were run on a machine with Intel Core2 Duo 1.66GHz and 1 GB RAM under Windows XP Professional Service Pack 2. Ten games were played with X6, 5 with black and 5 with white. Ten games were played against MeinStein also, but against five different settings, with both colors against each setting.

The results are listed in Table 4.1 and 4.2, where the Re-search entry is the number of defensive moves in the game that fail to defend and need to do another full defensive search. The number of candidates tried in each full defense search are shown in the next row. Only the moves that succeed to defend and are generated by the full defensive search are counted. The maximum number of alternative defensive moves tried by the full defensive search is 50. The alternative defensive move list is sorted according to the evaluation score given by the evaluation scheme presented in Chapter 3. In the result entry, D, W, and L, means we draw, win and lose, respectively.

It can be observed that our program can now perfectly draw against X6. In Table 4.1, only three out of ten games needs to do a re-search, and can

Table 4.1. Results of our defensive strategy combined with threat-space search against X6

Game	1	2	3	4	5	6	7	8	9	10
Result	D	D	D	D	D	D	D	D	D	D
Re-Search	0	0	0	1	0	0	0	1	2	0
Candidates	0	0	0	2	0	0	0	6	7,3	0
Game Length	121	119	123	121	119	123	124	126	124	120

Table 4.2. Results of our defensive strategy combined with threat-space search against MeinStein

Game	1	2	3	4	5	6	7	8	9	10
Result	W	L	W	L	L	W	W	W	L	W
search depth	5	5	4	4	6	6	5	5	5	5
quiescence	1	1	0	0	1	1	0	0	1	1
cutoff time	70	70	70	70	70	70	70	70	80	80
Re-Search	3	1	6	1	1	2	2	3	1	3
Candidates	6 ,6,4	1	6,6	5	1	6,8	2,6	6,6,4	1	6,6,4
Game Length	52	14	51	58	16	63	81	54	16	54

find an effective defensive move without more than 7 alternative moves.

The search depth row specifies the $\alpha\beta$-search depth, the quiescence specifies the depth of quiscience search, and the measure of cutoff time is in seconds of MeinStein. Our program won 60% of the games, while MeinStein only won 40%, showing that our program is slightly superior. Although almost every game needs a re-search, the effective defensive moves are still found by examining no more than 7 alternative candidates.

Therefore, from these two experiments, it can be seen that our defensive strategy can be effectively enhanced by combining with threat-space search, and this combination is able to compete with today's top programs. In Appendix B, we show some selected games in the experiments.

Chapter 5

Conclusions and Further Development

5.1 Conclusions

Connect6 is a relatively new game, and there is still a lot to be investigated and discovered. It has a huge complexity comparable to Go, making it interesting and challenging. Although techniques inspired from other games such as Go-Moku, or some standard techniques such as $\alpha\beta$-search work well in Connect6, there is still a large room for innovation and improvement especially in the realms of strategy.

In this thesis, we introduce a defensive strategy, with it a whole new evaluation scheme is proposed. The defensive strategy takes advantage of the property of locality in Connect6, and effectively reduces the huge branching factor in the game tree. The evaluation scheme is much more strategic sensitive compared to traditional evaluation methods, therefore is able to classify the value of a move into more detailed hierarchies. It is able to effectively reduce the opponent's attacking chances, and draw against some formidable opponents with just a simple greedy paradigm without any search algorithm.

Although the defensive strategy is effective, it is far from perfect. Threat-space search is combined to complement some of its weaknesses. Since threat-space search is used as a verifier, efficiency and accuracy have the utmost importance. We propose a method to use the sliding window algorithm to identify defensive moves and an enhancement to increase accuracy. A pattern table is implemented to increase the efficiency of the search. Furthermore, it can be applied to arbitrary board sizes. By integrating the threat-space search with the defensive strategy, it is demonstrated that the overall performance is effectively increased.

5.2 Further Development

The strategies and methods introduced in this thesis have high potential for further improvement.

- An attack strategy based on similar methods is yet to be explored. The proposed architecture can only win if victory is certain to come, i.e., it won't make any attempt to win.
- Consider a global evaluation based on these methods, making it applicable to search algorithm such as $\alpha\beta$-search. Investigate how to balance between attack and defence. Knowing when to attack and when to defend is the key to strategic understanding.
- Enhance threat-space search even further. Investigate the possibility of an efficient full threat-space search, and effectively mix single threats into the search sequence.

Appendix A

Kagami at the 14th Computer Olympiad

The Computer Olympiad is a multi-game event, and all the participants are computer programs. The Olympiad was proposed by David Levy, and he also organised the first Olympiad in London in 1989. Connect6 became a tournament item in 2006. The 14th Computer Olympiad was held in Pamplona, Spain, May 2009. In the Connect6 tournament each program must complete its moves for one game in 30 minutes.

Table A.1. 14th Computer Olympiad Connect6 Tournament Results

Standing	Program
1	Bit
2	MeinStein
3	Bit2
4	Kagami
5	Kavalan
6	Nomi6

Kagami, developed by the author, is a program based on the proposed defensive strategy. Threat-space search and a simple null-move scheme are also integrated. Kagami entered the 14th Olympiad with the purpose of testing the effectiveness of these techniques in tournament situations.

Six programs attended the tournament of Connect6 and Kagami finished fourth. Kagami is significantly faster than most of the programs, and even manages to draw against the silver medalist MeinStein and scores a win from the bronze medalist Bit2. Figures A.1 and A.2 are two of the games played in the tournament by Kagami.

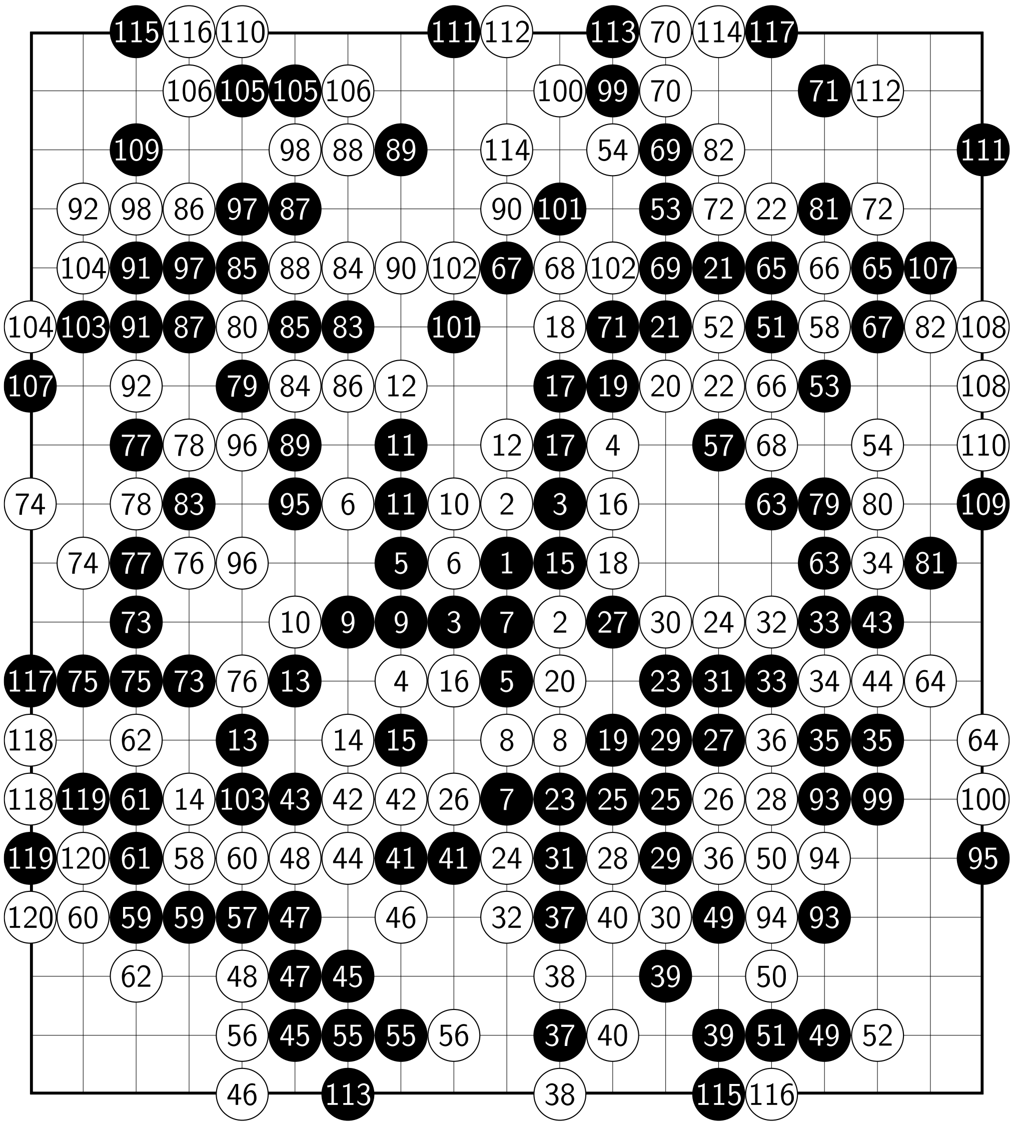

Figure A.1. Kagami (white) vs. MeinStein (black), game drawn.

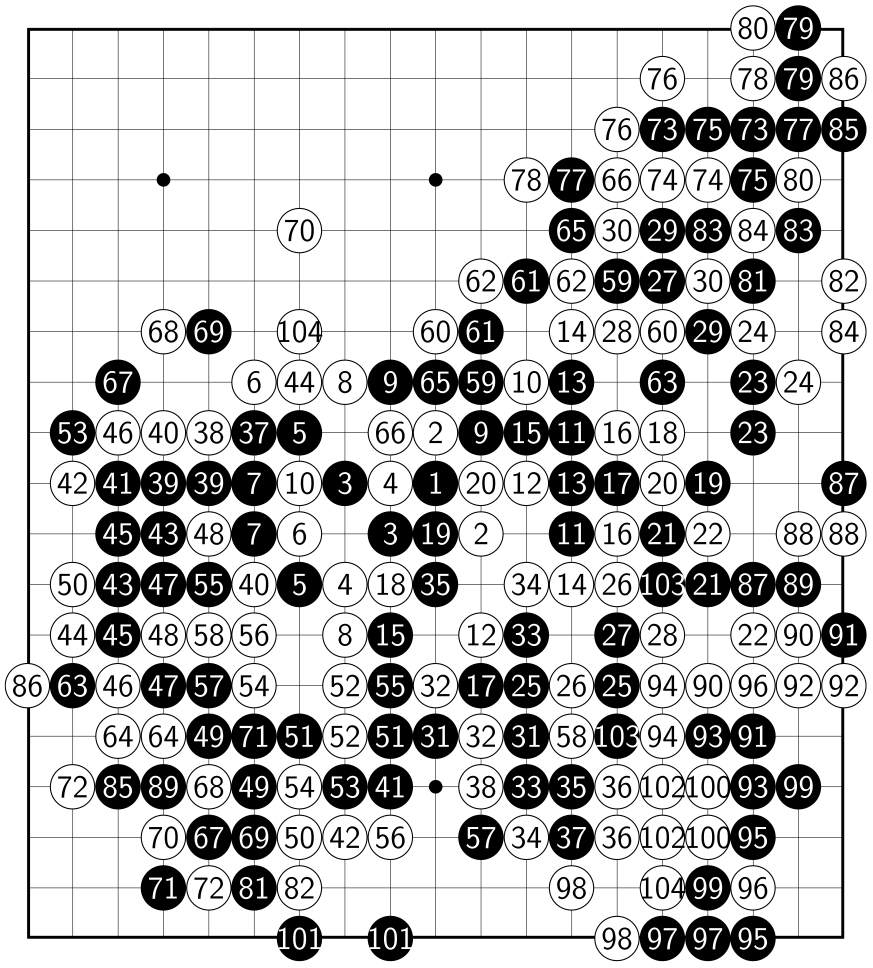
Figure A.2. Kagami (white) vs. Bit2 (black), white wins.

Appendix B

Selected Games

Here we present 8 selected games from the experiments of Chapters 3 and 4.

The first four games shown in Figures B.1 to B.4 are from the experiments of Chapter 3. The first two games are against NCTU6, and the next two games are against X6.

The last four games shown in Figures B.5 to B.8 are from the experiments of Chapter 4. The first two games are against X6, and the next two games are against MeinStein.

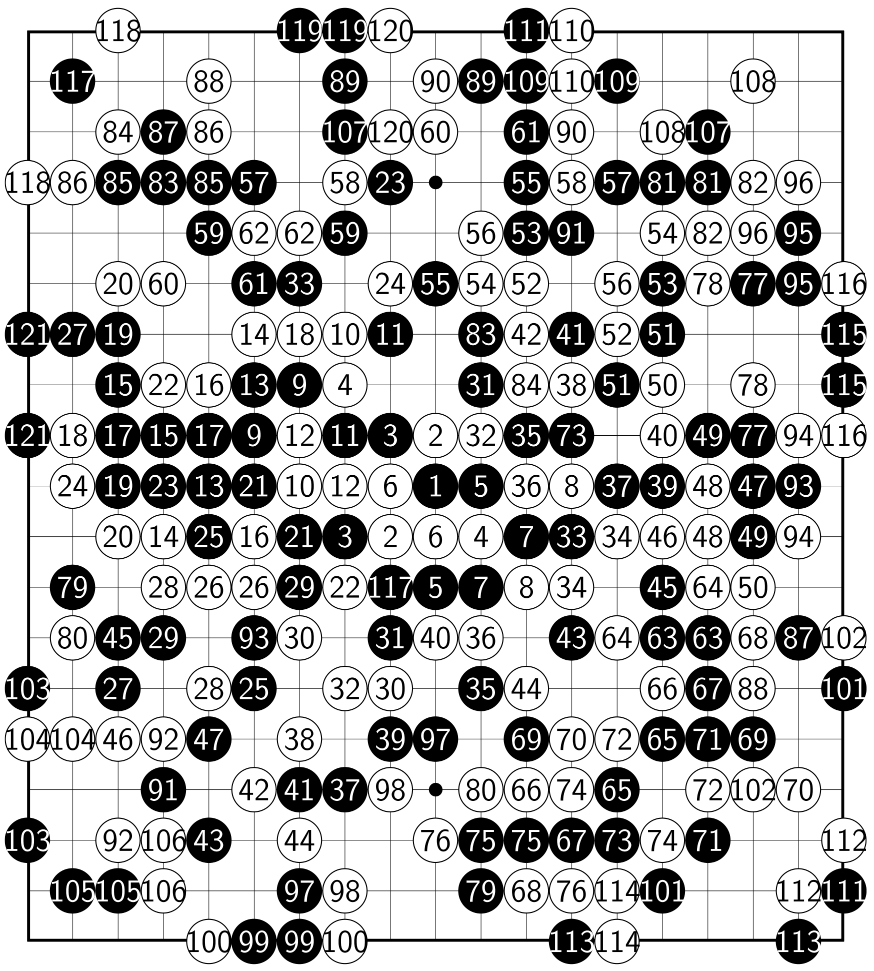
Figure B.1. Defensive Strategy (white) vs. NCTU6 (black), game drawn.

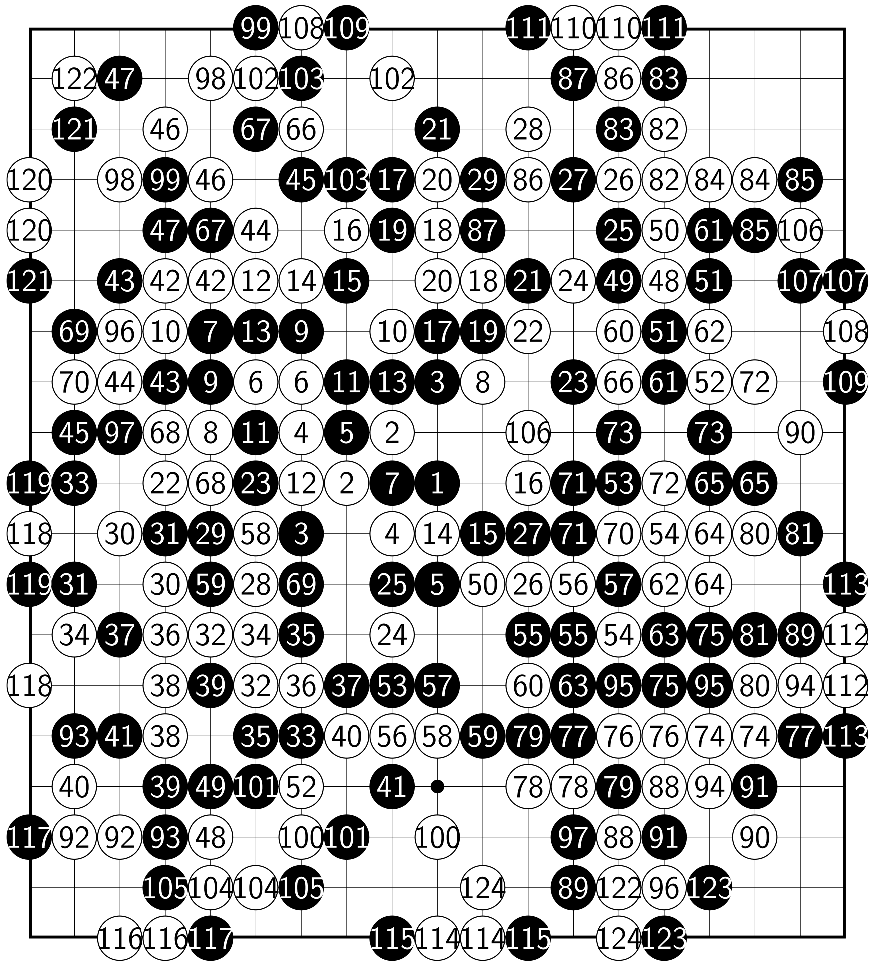
Figure B.2. Defensive Strategy (black) vs. NCTU6 (white), game drawn.

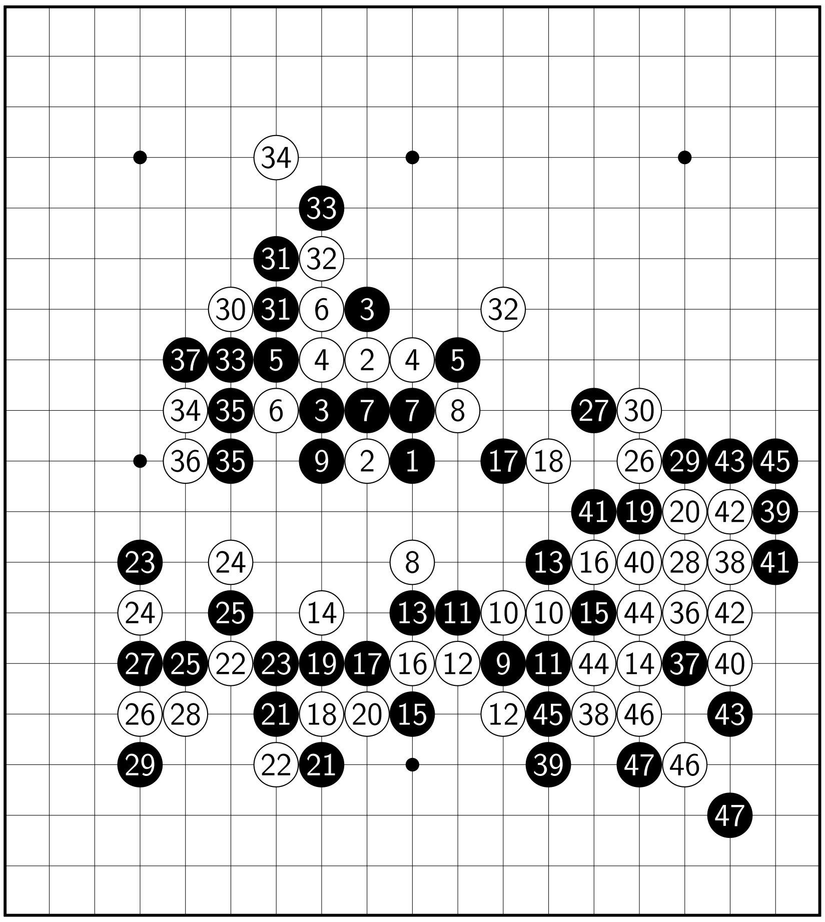
Figure B.3. Defensive Strategy (black) vs. X6 (white), white wins.

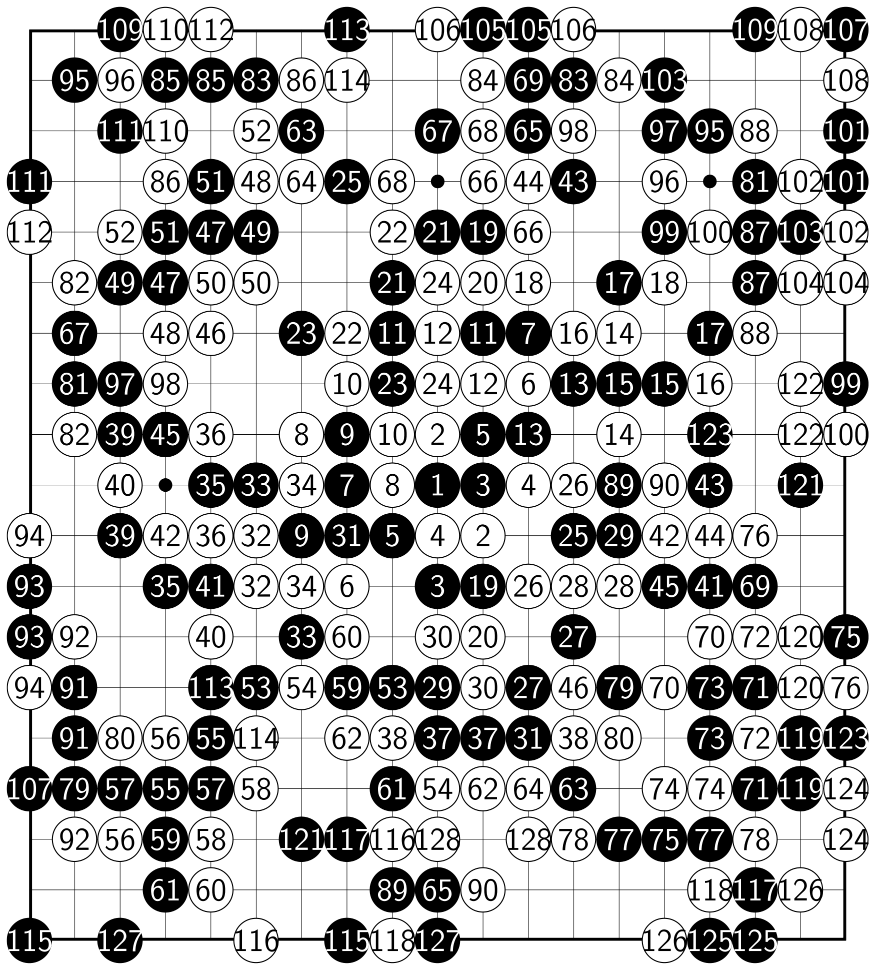
Figure B.4. Defensive Strategy (white) vs. X6 (black), game drawn.

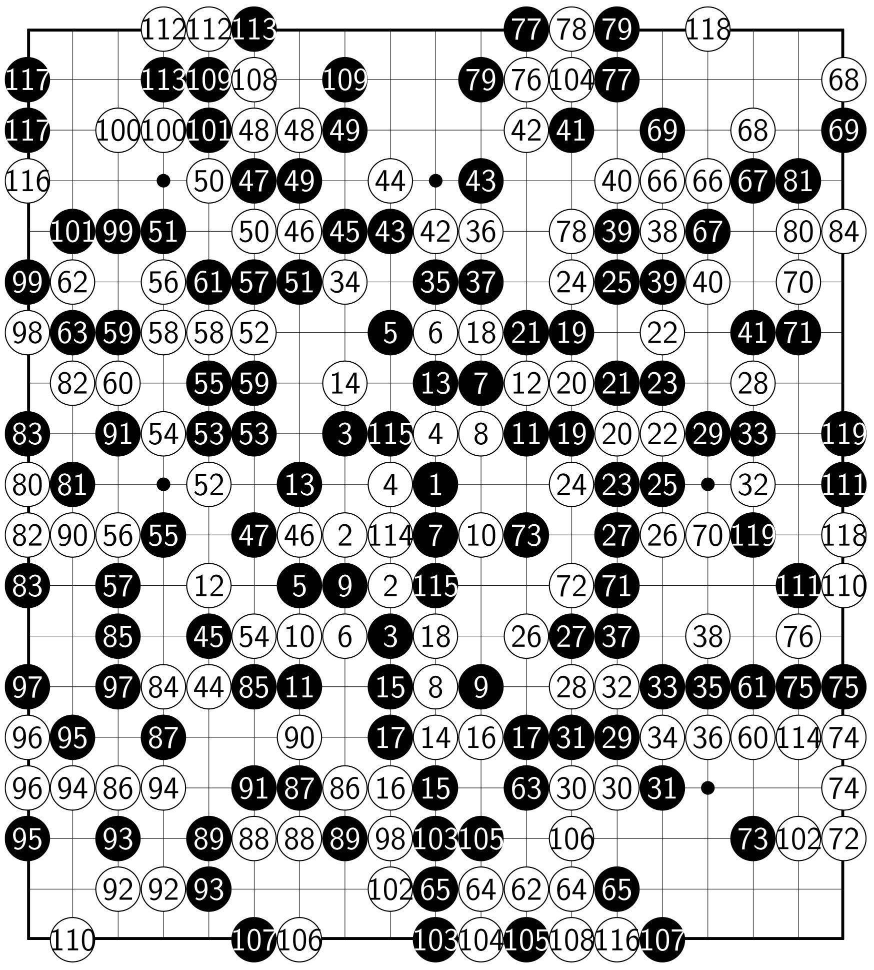

Figure B.5. Defensive Strategy combined with Threat Space Search (black) vs. X6 (white), game drawn.

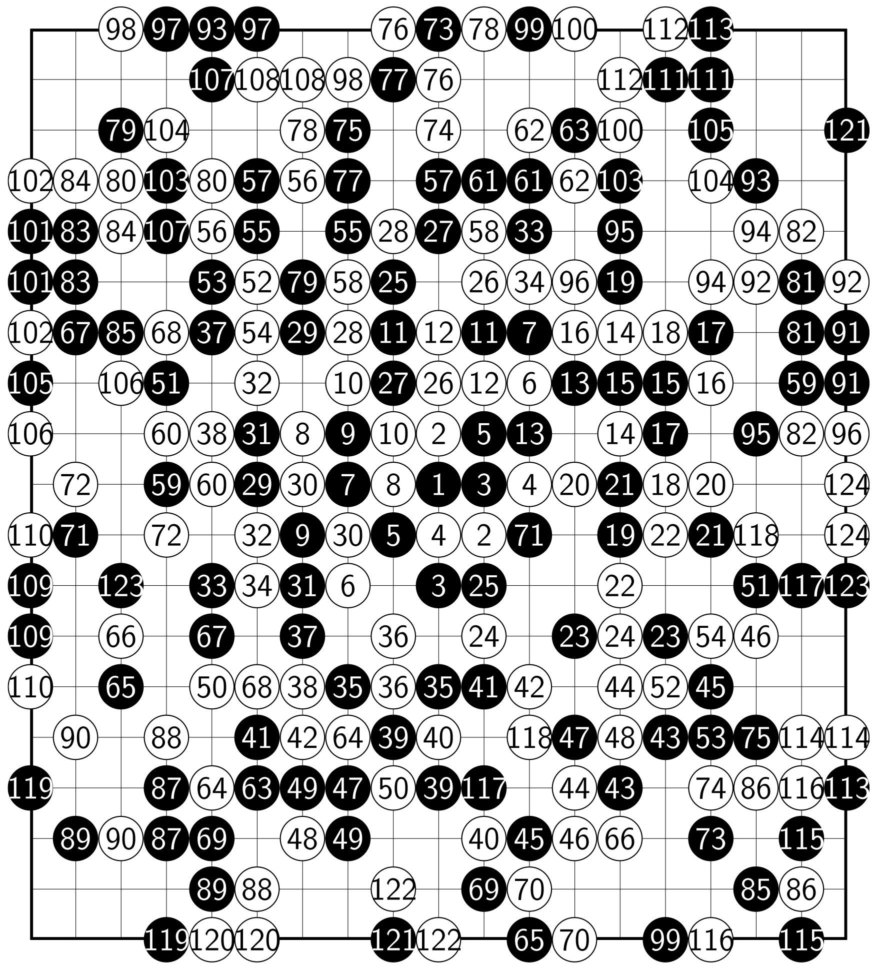

Figure B.6. Defensive Strategy combined with Threat Space Search (white) vs. X6 (black), game drawn.

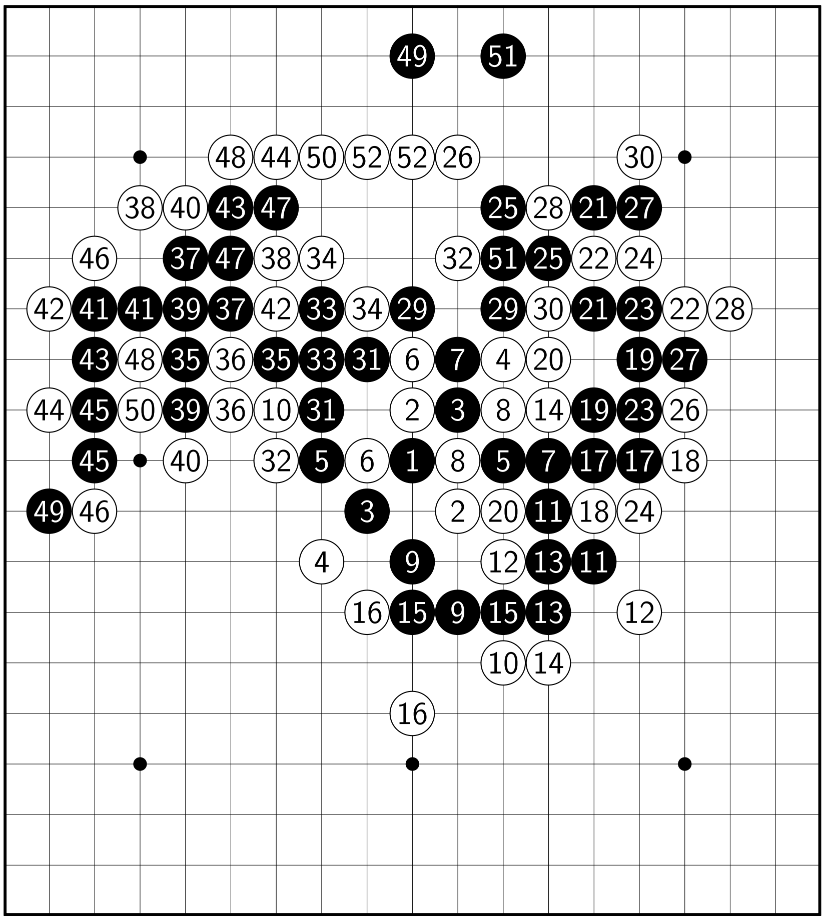

Figure B.7. Defensive Strategy combined with Threat Space Search (white) vs. MeinStein (black), white wins.

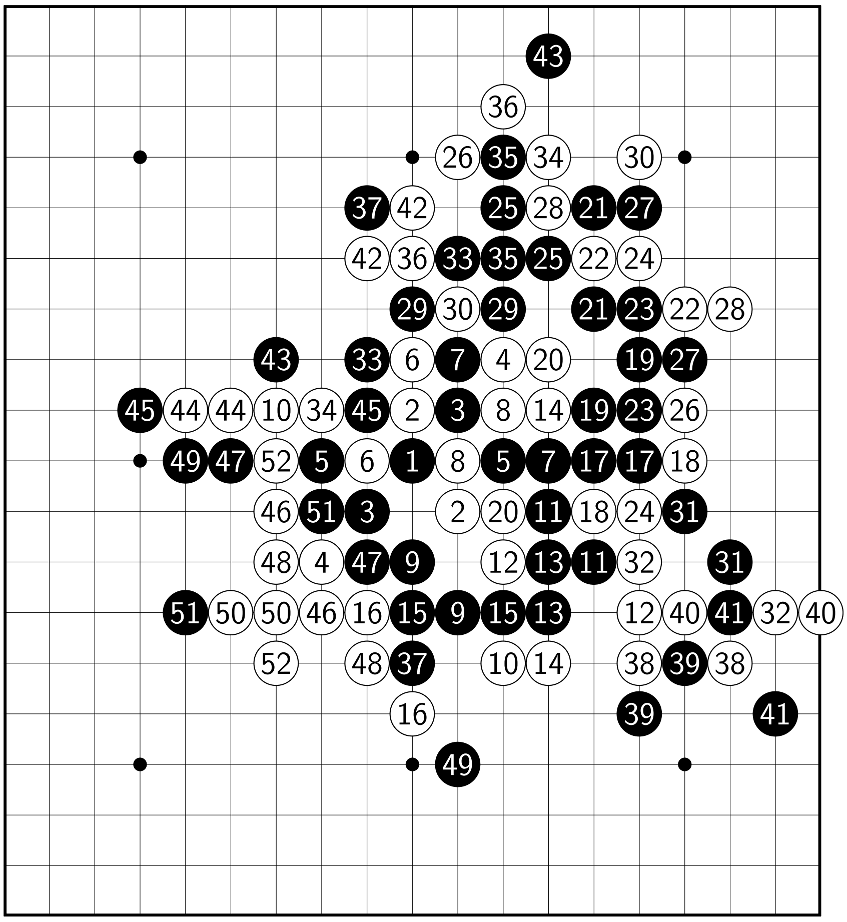

Figure B.8. Defensive Strategy combined with Threat Space Search (white) vs. MeinStein (black), white wins.

Bibliography

[1] J. MaCarthy, “Chess as the drosophila of a.i.” *Computers, Chess and Cognition*, pp. 227–237, 1990.

[2] H. van den Herik, J. Uiterwijk, and J. Rijswijck, “Games solved: Now and in the future.” *Artificial Intelligence*, vol. 134, pp. 277–311, 2002.

[3] I.-C. Wu and D.-Y. Huang, “A new family of k-in-a-row games,” in *the 11th Advances in Computer Games Conference (ACG’11), Taipei, Taiwan*, September 2005.

[4] I.-C. Wu, D.-Y. Huang, and H.-C. Chang, “Connect6,” *ICGA Journal*, vol. 28, no. 4, pp. 234–241, Decemeber 2005.

[5] L. Allis, H. van den Herik, and M. Hutjens, “Go-moku solved by new search techniques,” *Computational Intelligence*, vol. 12, pp. 7–23, 1996.

[6] L. Allis, “Searching for solutions in games and artificial intelligence,” Ph.D. dissertation, University of Limburg, Maastricht, 1994.

[7] D.-Y. Huang, “The study of artificial intelligence programming for gobang-like games,” Master’s thesis, National Chiao Tung University, June 2005.

[8] E. Berlekamp, J. Conway, and R. Guy, *Winning ways for your mathematical plays, Volume 2.* Academic Press, 1982.

[9] J. Uiterwijk and H. van den Herik, "The advantage of the initiative," *Information Sciences*, vol. 122(1), pp. 43–58, 2000.

[10] T. Zetters, "Problem s.10 proposed by r.k. guy and j.l. selfridge," *Amer. Math. Monlthly*, vol. 86, solution 87(1980), pp. 575–576, 1979.

[11] L. V. Allis and P. Schoo, "Qubic solved again," *Heruistic Programming in Artificial Intelligence 3: The Third Computer Olympiad*, pp. 192–204, 1992.

[12] A. W. Hales and R. Jewett, "Regularity and positional games," *Transactions of the American Mathematical Society*, vol. 106, pp. 222–229, 1963.

[13] S. W. Golomb and A. W. Hales, "Hypercube tic-tac-toe," *More Games of No Chance*, vol. 42, pp. 167–180, 2002.

[14] S.-H. Chiang, I.-C. Wu, and P.-H. Lin, "On drawn k-in-a-row games," in *the 12th Advances in Computer Games Conference (ACG12), Pamplona, Spain*, May 2009.

[15] "Connect6 homepage," *http://www.connect6.org*.

[16] S.-Y. Liou, "Design and implementation of computer connective 6 program x6," Master's thesis, National Dong Hwa University, July 2006.