

# 國立交通大學

資訊科學與工程研究所

博士論文

六子棋之區域證明搜尋

Relevance-Zone-Oriented Proof Search for Connect6

研究生：林秉宏

指導教授：吳毅成 教授

中華民國九十九年十一月

六子棋之區域證明搜尋  
Relevance-Zone-Oriented Proof Search for Connect6


研究生：林秉宏

Student : Ping-Hung Lin

指導教授：吳毅成

Advisor : I-Chen Wu

國立交通大學  
資訊科學與工程研究所  
博士論文



A Dissertation  
Submitted to Institute of Computer Science and Engineering  
College of Computer Science  
National Chiao Tung University  
in partial Fulfillment of the Requirements  
for the Degree of  
Doctor of Philosophy  
in  
Computer Science

November 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年十一月

# 六子棋之區域證明搜尋

研究生：林秉宏

指導教授：吳毅成 博士

國立交通大學資訊科學與工程研究所博士班

## 摘要

西元 2005 年，吳毅成教授提出了一系列新的 K 子棋遊戲，在這類遊戲中，六子棋特別引起高度的關注。本論文提出了一種新的迫著證明搜尋方式，稱之為區域證明搜尋 (RZOP)，以 Thomsen 所提出的 lambda 搜尋為基礎，此方法會建構出相關證明區域。區域證明搜尋是一種全新、通用而且優雅的方法。本論文已成功有效的解出許多六子棋盤面的勝敗，其中包含多個開局，例如米老鼠開局，在過去是很受歡迎的一種的開局。除了解題，本論文進一步改進區域證明搜尋的效率，稱之為區域內線段證明搜尋 (SRZOP)，此方法有效加速證明盤面勝敗所需花費的時間。根據實驗數據中 12 種開局的統計結果，區域內線段證明搜尋可加快 2.04 倍的時間。最後，附錄 F 展示作者和交大六號 (六子棋 AI 程式) 的相關比賽成果。例如在 2008 年第十三屆國際奧林匹亞電腦賽局競賽的六子棋組，交大六號輕量版獲得冠軍，作者也因此榮獲交大 98 年度 (春季) 重要學術獎；第二屆人腦對電腦六子棋大賽，交大六號更得到 8 勝 0 敗的好成績。

# Relevance-Zone-Oriented Proof Search for Connect6

Student : Ping-Hung Lin

Advisor : Dr. I-Chen Wu

Institute of Computer Science and Engineering  
National Chiao Tung University

## Abstract

Wu and Huang presented a new family of  $k$ -in-a-row games, among which Connect6 (a kind of six-in-a-row) attracted much attention. For Connect6 as well as the family of  $k$ -in-a-row games, this thesis proposes a new threat-based proof search method, named Relevance-Zone-Oriented Proof (RZOP) search, developed from the lambda search proposed by Thomsen. The proposed RZOP search is a novel, general and elegant method of constructing and promoting relevance zones. This thesis solved effectively and successfully many new Connect6 game positions, including several Connect6 openings, especially the Mickey-Mouse Opening, which used to be one of the popular openings before we solved it. In addition to solvability, this thesis further improves the RZOP method, named Segmented Relevance-Zone-Oriented Proof (SRZOP) search, which speeds up the time to solve Connect6 game positions. The experimental results show 2.04 speedups in total to solve 12 openings. Finally, this thesis demonstrates records of our Connect6 program, *NCTU6*, which won the gold in the 13<sup>th</sup> Computer Olympiads in 2008; and also won eight games and lost none against top Connect6 players in Taiwan in 2009.

## 致謝

經過了孜孜矻矻的多年，我終於在 2010 年 11 月拿到了博士學位。

感謝指導老師吳毅成教授多年來的提攜與照顧，不斷地鞭策我的研究和指點我為人處事的道理。老師在理論和實務方面的教學研究，也給予我學習的方向，我將秉持老師的教導，在人生的道路上，一步步努力前進。

除了影響我最深的老師之外，一直鼓勵、扶持和砥礪我，給予我持之以恆的動力，讓我能通過這佈滿大大小小石頭、充滿歡笑和眼淚的博士道路的人，就是我的太太。她要修習自己的學業，還要照顧我的生活起居，我由衷地感謝。

感謝論文口試委員朱正忠教授、林順喜教授、徐慰中教授、徐讚昇教授、許舜欽教授、陳穎平教授和顏士淨教授（以上按姓氏筆劃排列），對論文的改進方向，提出寶貴的意見，讓我的研究能更上一層樓。

培育了我多年的交大和我在 CYC Lab 的所有親朋好友們，給予我各式各樣的支持，在此衷心的感謝大家，包括陳隆彬學長、蘇瑞元學長、徐建智學長、hangten、coboy、阿杰、ricky、sarten、uncle、RC、BJ、益嘉、德中、宏軒、家茵、lida、小熊、QQting…等等，特別感謝 Mark 和草莓在我的口試時細心地幫我準備餐點。

最後，感謝我摯愛的爸爸、媽媽和尊敬的岳父、岳母。提供我麵包，讓我可以專心在論文研究；提供我避風港，讓我在遇到困難時，內心仍能充滿溫暖，不畏懼。沒有您們，這篇論文將無法完成。謹以此論文獻給我最摯愛的家人。

林秉宏

2010 年 10 月 29 日

# Contents

摘要.....	i
Abstract.....	ii
致謝.....	iii
List of Figures.....	vi
List of Tables.....	viii
Chapter 1 Introduction.....	1
1.1 Game Positions.....	2
1.2 Playing Strategies for Connect6.....	3
1.3 Winning Strategies for Connect6.....	5
1.4 Motivation.....	10
Chapter 2 Related Works.....	14
2.1 Search Trees.....	14
2.2 Null Move Heuristics for Connect6.....	17
2.3 Lambda Search for Connect6.....	20
Chapter 3 Relevance-Zone-Oriented Proof Search for Connect6.....	23
3.1 Relevance Zones.....	23
3.2 The Proposed Verifier $V_{C6}$ .....	28
3.2.1 Endgame Positions.....	28
3.2.2 Positions in Attacker's Turn.....	31
3.2.3 Positions in Defender's Turn.....	32
3.2.3.1. Three Threats or More.....	33
3.2.3.2. Two Threats.....	36
3.2.3.3. One Threat.....	40
3.2.3.4. No Threats.....	43
3.3 Conclusion for the Verifier $V_{C6}$ .....	45
Chapter 4 Segmented Relevance-Zone-Oriented Proof Search for Connect6.....	46
4.1 Irrelevant vs. Relevant Sequences of Squares.....	46
4.1.1 The Proposed Verifier $V_{C6-O1}$ .....	50
4.1.1.1. Endgame Positions.....	50
4.1.1.2. Positions in Attacker's Turn.....	51
4.1.1.3. Positions in Defender's Turn.....	52
4.1.1.3.1. Three Threats or More.....	52
4.1.1.3.2. Two Threats.....	54
4.1.1.3.3. One Threat.....	55

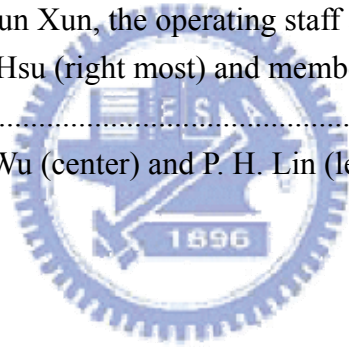
4.1.1.3.4.	No Threats .....	57
4.1.2	Conclusion for the Verifier $V_{C6-01}$ .....	59
4.2	Counter-threat Sequences of Squares.....	60
4.2.1	The Proposed Verifier $V_{C6-02}$ .....	63
4.2.1.1.	Endgame Positions .....	63
4.2.1.2.	Positions in Attacker's Turn.....	64
4.2.1.3.	Positions in Defender's Turn .....	64
4.2.1.3.1.	Three Threats or More.....	65
4.2.1.3.2.	Two Threats .....	65
4.2.1.3.3.	One Threat .....	66
4.2.1.3.4.	No Threats .....	67
4.2.2	Conclusion for the Verifier $V_{C6-02}$ .....	69
Chapter 5	Experiments.....	70
5.1	Assistant Programs .....	71
5.1.1	Solver .....	71
5.1.2	Verifier.....	72
5.1.3	Desktop Grids and Volunteer Computing for Connect6 .....	75
5.1.4	Job-Level Proof-Number Search for Connect6.....	78
5.2	Illustration of Solving Positions.....	80
5.3	Results .....	83
Chapter 6	Conclusions .....	90
References	.....	92
Appendix A	Sample Positions .....	98
Appendix B	Results of RZOP Benchmark .....	115
Appendix C	Results of SRZOP Benchmark.....	119
Appendix D	Verifiers for General Connect Games .....	125
Appendix E	Draw K-in-a-row Games.....	130
Appendix F	Author's Records.....	131
Vita	.....	138

## List of Figures

Figure 1. Threat patterns for Connect6. (a) One threat, (b) two threats and (c) three threats. ...	4
Figure 2. (a) Normal critical defense and (b) relaxed critical defense. ....	5
Figure 3. A sequence of VCDT winning strategy. ....	6
Figure 4. A sequence of VCST winning strategy. ....	7
Figure 5. A sequence of VCNT winning strategy. ....	8
Figure 6. Black's winning move in Conect(6,2,3). ....	9
Figure 7. (a) A position with Black winning. (b) A positon with White winning. ....	10
Figure 8. (a) A VCDT for the null move in Figure 7 (a). (b) A winning single-threat move 9 for the semi-null move 8. ....	11
Figure 9. (a) A winning single-threat move 8 for a null move in Figure 7 (b). (b) A winning non-threat move 8 for a semi-null move 7. ....	11
Figure 10. (a) A search tree and (b) a solution tree. ....	15
Figure 11. (a) Making squares of moves by inserting small boxes. (b) Combining the same edges from (a). ....	16
Figure 12. (a) A VCDT for a null move in Figure 6. (b) A VCDT for a semi-null move 2. ....	18
Figure 13. The proof search tree for solving <i>Connect(6,2,3)</i> . ....	19
Figure 14. A $\Lambda^3$ -tree. ....	21
Figure 15. A $\Lambda^3$ -strategy. ....	22
Figure 16. A sequence of zones $\langle Z_1, Z_2, Z_3 \rangle$ . ....	24
Figure 17. A sequence of relevance zones $\Psi = \langle Z_1, Z_2 \rangle$ for the winning position in Figure 12 (a). ....	25
Figure 18. (a) Relevance zones in a line and (b) in a board, upon winning with a win segment. ....	29
Figure 19. (a) Relevance zones in a line and (b) in a board, upon winning with three or more threats. ....	34
Figure 20. A winning positon with two threats for Black (Attacker) and the constructed $\Psi(P)$ . ....	37
Figure 21. A winning positon with two threats for Black (Attacker) and the constructed $\Psi(P)$ . ....	39
Figure 22. (a) A VCDT for the semi-null move 9. (b) A relaxed critical defense at 9. (c) The constructed zones for the semi-null move 9 in (a). ....	41
Figure 23. An example proof search tree for the Verifer $V_{C6}(P,S)$ . ....	45



Figure 24. An example proof search tree, where $\check{Z}_2 = Z_2 \setminus Z_1$ , for the Verifier $V_{C6-O1}(P,S)$ . .....	59
Figure 25. Two types of moves $M'_D(D12, G11)$ and $M''_D(D6, G6)$ .....	60
Figure 26. For Defender's first square $s$ , the dash line indicates the possible area for the second square $s'$ that may form counter-threat segments. ....	62
Figure 27. An example proof search tree, where $\check{Z}_2 = Z_2 \setminus Z_1$ , for the Verifier $V_{C6-O2}(P,S)$ . .....	69
Figure 28. (a) A proof search tree of <i>NCTU6-Verifier</i> and (b) the verifier of one higher order.	73
Figure 29. Desktop grid architecture.....	76
Figure 30. The proof search tree for the position in Figure 7 (a). .....	80
Figure 31. The proof search tree for the position in Figure 7 (b). .....	81
Figure 32. A sequence of $\Lambda^3$ -move starting from 7. ....	82
Figure 33. Six openings in which Black wins at 3. ....	86
Figure 34. 65 winning positions. ....	98
Figure 35. P. H. Lin, I-C. Wu and H.J. van den Herik. ....	133
Figure 36. L. Lee ( <i>BITSTRONGER</i> ) and P. H. Lin ( <i>NCTU6-LITE</i> ).....	134
Figure 37. The certificate of the 13th Computer Olympiad by NCTU. ....	134
Figure 38. Go Champion Chou Jun Xun, the operating staff and P. H. Lin. ....	135
Figure 39. Professor Shun-Chin Hsu (right most) and members of the Connect6 team lead by I-C. Wu. ....	136
Figure 40. Human players, I-C. Wu (center) and P. H. Lin (left most). ....	136



## List of Tables

Table 1. The solvability of verifiers for the three puzzles. “yes” means solved and “no” means unsolved. ....	84
Table 2. (a) The statistics of verifiers for the three puzzles in number of nodes. (b) Speedups compare to $V_{C6}$ . ....	84
Table 3. (a) The statistics of verifiers for the three puzzles in time (in seconds). (b) Speedups compare to $V_{C6}$ . ....	85
Table 4. The solvability of verifiers for 65 winning positions. ....	87
Table 5. (a) The statistics of verifiers for 65 winning positions in number of nodes. (b) Speedups compare to $V_{C6}$ . ....	87
Table 6. (a) The statistics of verifiers for 65 winning positions in time (in seconds). (b) Speedups compare to $V_{C6}$ . ....	87
Table 7. (a) The statistics of verifiers for 12 openings in number of nodes. (b) Speedups compare to $V_{C6}$ . ....	88
Table 8. (a) The statistics of verifiers for 12 openings in time (in seconds). (b) Speedups compare to $V_{C6}$ . ....	88
Table 9. The solvability of verifiers for 65 winning positions in Appendix A, where “yes” means solved and “no” means unsolved. ....	115
Table 10. The statistics of verifiers for 65 winning positions in Appendix A: (a) number of nodes and (b) times. ....	119
Table 11. The participants and the final standings of the Connect6 Tournament in the 13th Computer Olympiad (2008). ....	133

# Chapter 1 Introduction

A generalized family of  $k$ -in-a-row games, named  $Connect(m, n, k, p, q)$  [65][66], were introduced and presented by Wu *et al.* Two players, named *Black* and *White*, alternately place  $p$  stones on empty *squares*<sup>1</sup> of an  $m \times n$  board in each turn. Black plays first and places  $q$  stones initially. The player who first gets  $k$  consecutive stones of his own horizontally, vertically and diagonally wins. Both players tie the game when the board is filled up with neither player winning. Games in this family are also called *Connect* games<sup>2</sup> in this thesis. For example, *Tic-tac-toe* is  $Connect(3, 3, 3, 1, 1)$ , *Go-Moku* in the free style (a traditional five-in-a-row game) is  $Connect(15, 15, 5, 1, 1)$ , and *Connect6* played on the traditional Go board is  $Connect(19, 19, 6, 2, 1)$ . For simplicity, let  $Connect(k, p, q)$  denote the game  $Connect(\infty, \infty, k, p, q)$ , played on infinite boards. For example, when played on infinite boards, *Go-Moku* becomes  $Connect(5, 1, 1)$  and *Connect6* becomes  $Connect(6, 2, 1)$ .

Among these *Connect* games, *Connect6* attracted much attention due to three merits, *fairness*, *simplicity of rules* and *high game complexity* as described in [65][66]. Since *Connect6* was introduced, hundreds of thousands of *Connect6* games have been played on web sites, such as [littlegolem.net](http://littlegolem.net) [33] and [cycgame.com](http://cycgame.com) [51]. Since 2006, several *Connect6* open tournaments [50] for human players have been held, such as NCTU Open, ThinkNewIdea Open, Russian Open and World Open. *Connect6* has also been included as one of the computer game tournaments in Computer Olympiad [55] and Chinese Computer Games Contest [16], since 2006 and 2007 respectively.

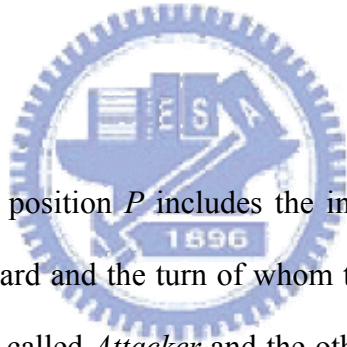
---

<sup>1</sup> Practically, stones are placed on empty intersections of Renju or Go boards. In this thesis, by squares, we mean intersections.

<sup>2</sup> The term of connect games defined in [22] covers the games such as Hex, Connect Four, etc. In this thesis, *Connect* are capitalized to indicate all the games in the family of  $Connect(m, n, k, p, q)$ .

In Connect6, a *segment* is defined to be a set of *six* consecutive squares horizontally, vertically or diagonally on the board; while in  $Connect(m,n,k,p,q)$ , a segment is a set of  $k$  consecutive squares. A segment is called an *empty segment* if all the squares on it are unoccupied yet. A segment is called an *active segment* of one player, if none of the squares are occupied by the opponent's stones. An active segment of one player is called a *win segment* of the player, if all the squares on it are occupied by the player. Obviously, one player wins if the player makes a win segment. From the definition of Connect games, a game ends when one makes some win segment or all the squares of the board are already occupied. According to this definition, it is impossible for both players to have win segments simultaneously.

## 1.1 Game Positions



In Connect games, a game position  $P$  includes the information of all the stones and their occupied squares on the board and the turn of whom to play. The player to be proved to win, either Black or White, is called *Attacker* and the other *Defender* in this thesis. Both input and output game positions are in the standard format, named SGF [40]. Let  $\sigma_A(s)$  denote the information of an Attacker stone placed on the unoccupied square  $s$ , and  $P + \sigma_A(s)$  denote the position after placing an Attacker stone on  $s$  in position  $P$  *without changing the turn*.  $\sigma_D(s)$  and  $P + \sigma_D(s)$ , are similarly defined for Defender. From the strategy stealing argument by Nash (cf. [7][65]), we obtain the following. If Attacker wins in  $P$ , Attacker wins in  $P + \sigma_A(s)$ , too; and if Attacker wins in  $P + \sigma_D(s)$ , Attacker wins in  $P$ , too.

In this thesis,  $P \oplus M$  denotes the position after one player makes move  $M$  and before the other makes the next move. In Connect6, let  $M_A(s_1, s_2)$  denote an Attacker move where two Attacker stones are placed on both unoccupied squares  $s_1$  and  $s_2$ .  $M_D(s_1, s_2)$  and  $P \oplus M_D(s_1, s_2)$

are similarly defined for Defender. Note that in contrast to  $P + \sigma_A(s_1) + \sigma_A(s_2)$ , the position  $P \oplus M_A(s_1, s_2)$  indicates changing the turn from Attacker to Defender.

In Connect6, one player, say Attacker, is allowed to make a *null move*,  $M_{A, \phi\phi}$ ; that is, to place no stones, and a *semi-null move*,  $M_{A, \phi}(s_1)$ ; that is, to place one stone only on square  $s_1$  in  $P$ . Thus, the position  $P \oplus M_A(s_1, s_2)$  is equivalent to  $(P \oplus M_{A, \phi}(s_1)) + \sigma_A(s_2)$  and  $(P \oplus M_{A, \phi\phi}) + \sigma_A(s_1) + \sigma_A(s_2)$ . From another viewpoint, null or semi-null moves are to place some *null stones* while placing normal stones. In  $Connect(m, n, k, p, q)$ , we place  $p$  null stones for a null move, while placing one to  $p-1$  null stones for semi-null moves.

## 1.2 Playing Strategies for Connect6

In Connect6 (other Connect games are similar), *threats* are the key to great reduction of the proof search tree. An active segment in which Attacker occupied four or five squares is called a *threat segment* of Attacker. The segment poses a threat and Defender has to block it, or Attacker wins by making a win segment in the next move.

A move is called a *single-threat move* if the player who makes the move has one and only one threat after the move, a *double-threat move* if two, a *triple-threat move* if three, and a *non-threat move* if none. In Connect6, one player clearly wins by a *triple-threat-or-more move* (a move with at least three threats). Examples of the line patterns with one, two and three threats are shown in Figure 1 (below).

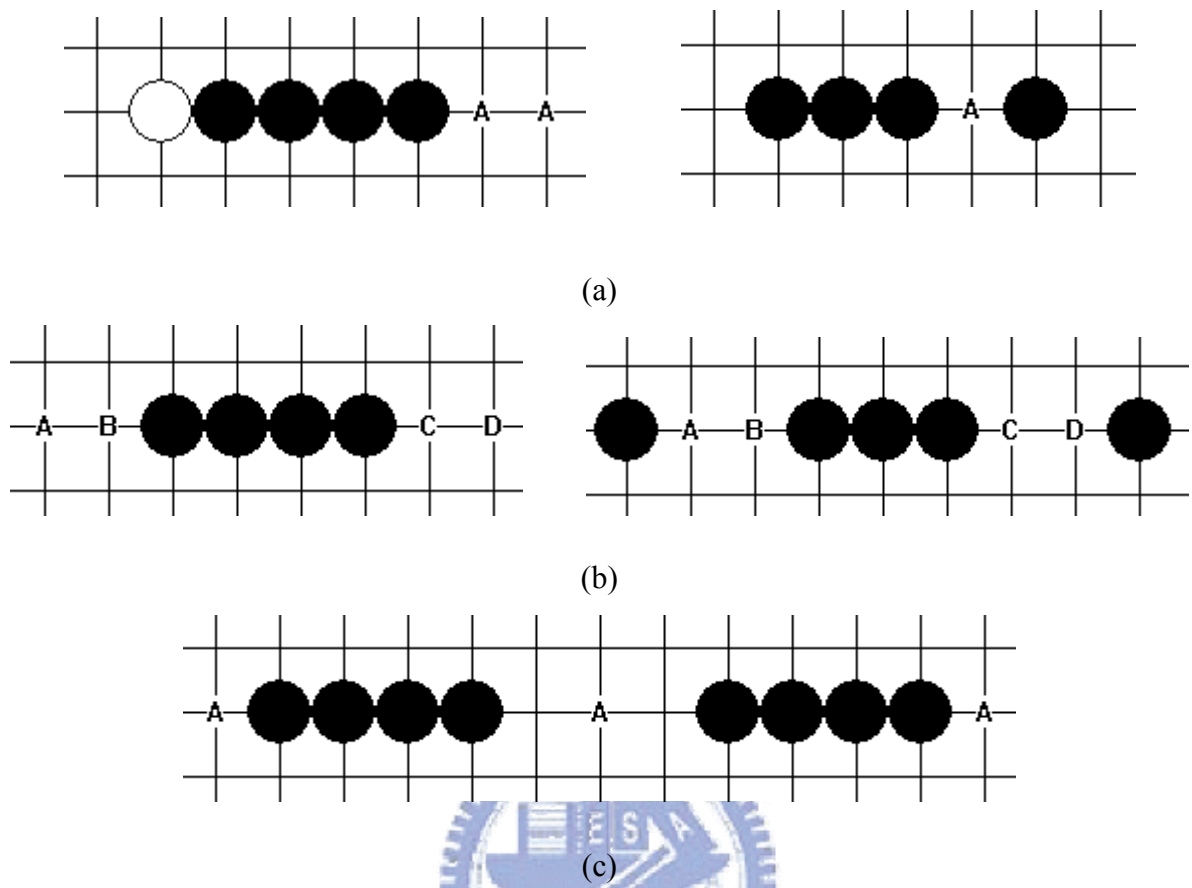
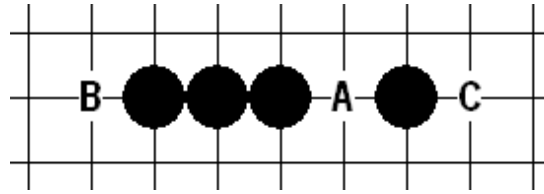
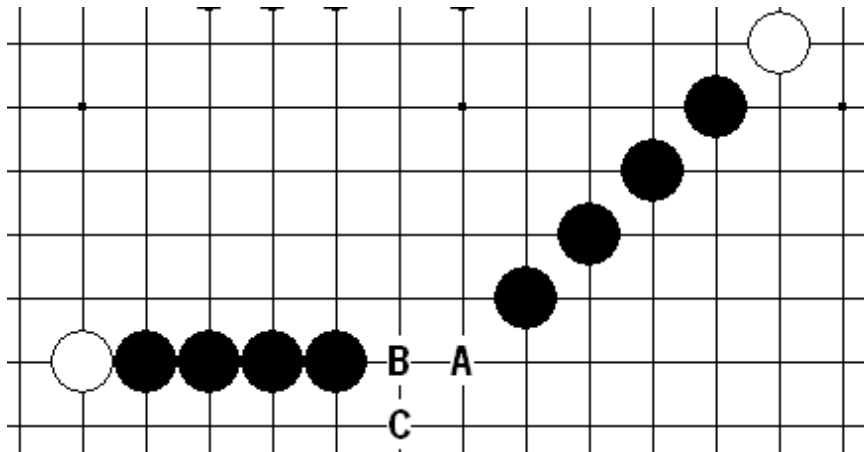


Figure 1. Threat patterns for Connect6. (a) One threat, (b) two threats and (c) three threats.

The defensive moves that block all the threats are called *critical defenses*, while removing any stones in the moves unblocks some threats. For example, White's semi-null moves  $M_{D,\phi}(A)$  and moves  $M_D(B,C)$  in both Figure 2 (a) and (b) are critical defenses, while moves  $M_D(A,B)$  are not because the threats are still blocked without  $B$ . (Note that null moves are also critical defenses in positions without any threats according to the above definition.) Critical defenses are said to be *normal* if the numbers of stones in the defenses are the same as the numbers of threats; and *relaxed*, otherwise. For example, in Figure 2, semi-null moves  $M_{D,\phi}(A)$  are normal, while moves  $M_D(B,C)$  are relaxed. In Connect6, relaxed critical defenses are not played frequently due to their inefficiency (using two stones to block only one threat).



(a)



(b)

Figure 2. (a) Normal critical defense and (b) relaxed critical defense.

### 1.3 Winning Strategies for Connect6

In [65][66], they showed a type of winning strategy, called *Victory by Continuous Double-Threat-or-more moves (VCDT)* in this thesis. It is similar to *Victory by Continuous Four (VCF)*, a common term for winning strategies in the Renju community [41]. More specifically, the type of VCDT strategy is to win by making continuously double-threat moves and ending with a triple-or-more-threat move or connecting up to six in all variations. For example, in Figure 3 (below), White's VCDT 12-18 (18 is a triple-threat move).

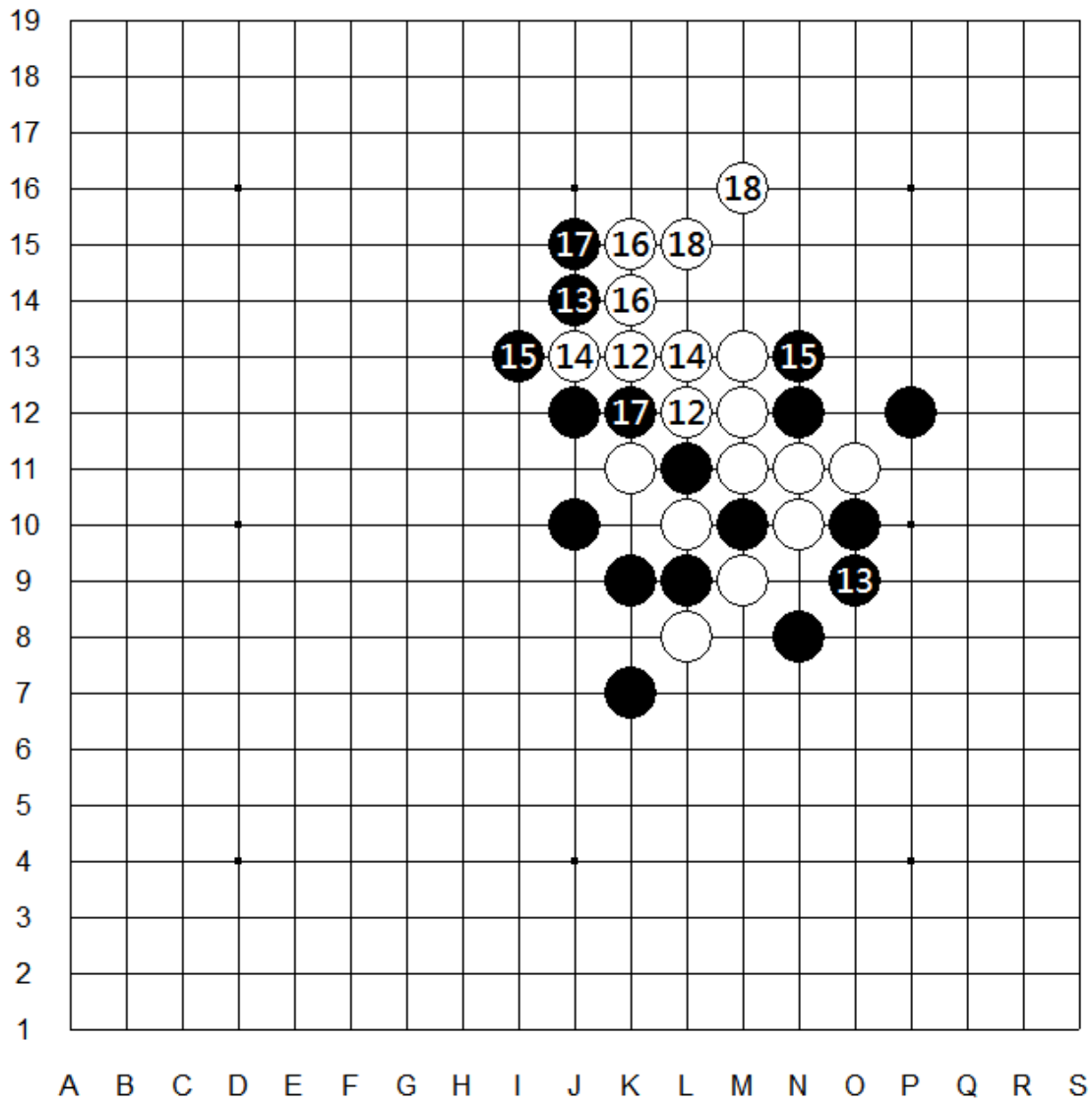


Figure 3. A sequence of VCDT winning strategy.

Soon after the introduction of Connect6, many human experts found another type of winning strategy in which additional single-threat moves are involved, i.e., single-threat and double-threat moves are mixed (before ending with a triple-or-more-threat move). This type of winning strategy is herein called *Victory by Continuous Single-Threat-or-more moves (VCST)*. For example, Lee [32], a Renju 3-dan player, found and claimed in late 2005 that White won starting from move 8 (both 8 and 10 are single-threat moves) in the game as shown in Figure 4 (below).



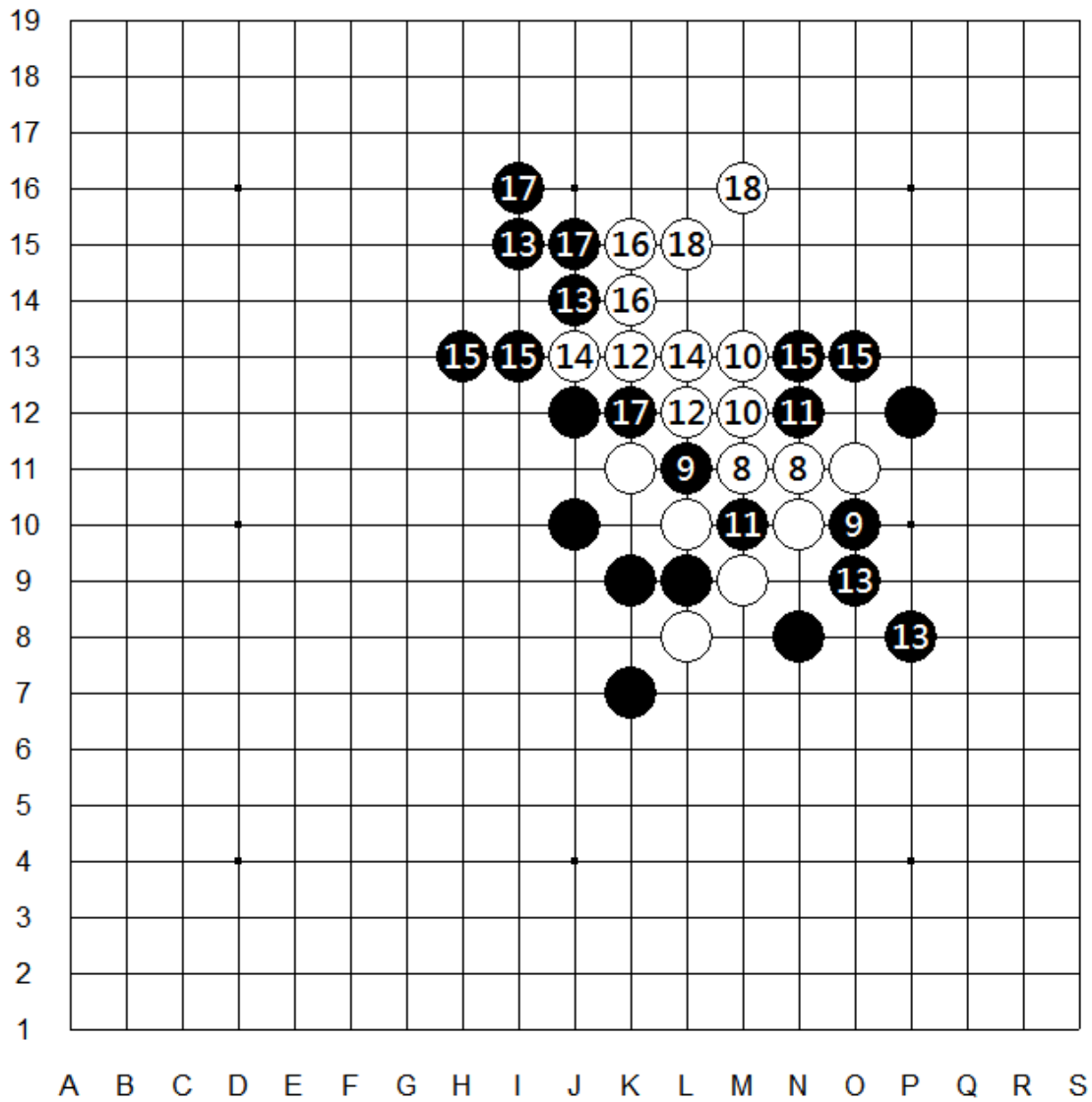


Figure 4. A sequence of VCST winning strategy.

Similarly, the type of winning strategy with additional non-threat moves involved is called *Victory by Continuous Non-Threat-or-more moves (VCNT)*. For example, Black won starting from move 1 (1 is a non-threat move) in *Connect(6,2,3)* as shown in Figure 5 (below).

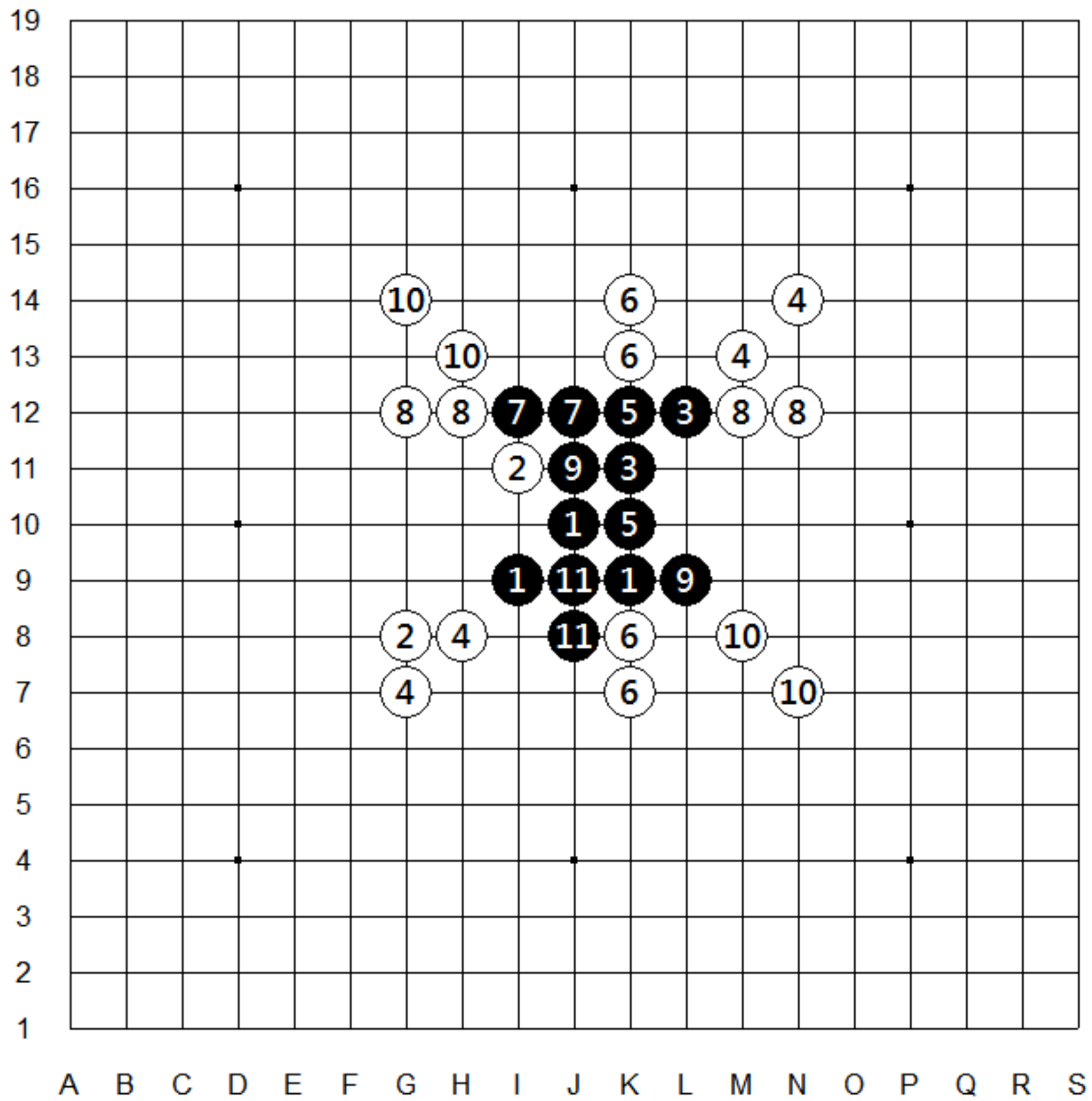


Figure 5. A sequence of VCNT winning strategy.

Although VCST was unknown then, Wu and Huang [65][66] were already able to solve a simple VCNT case, that Black wins  $Connect(6,2,3)$ . This clearly is a case of VCNT, since Black's first winning move, as shown in Figure 6 (below), must be a non-threat move.

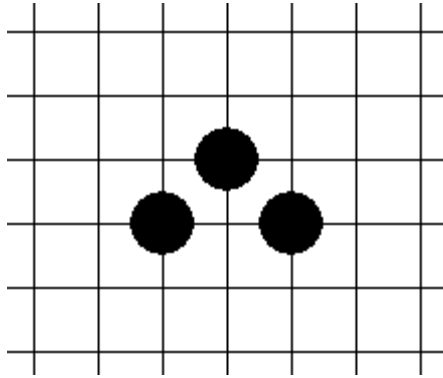
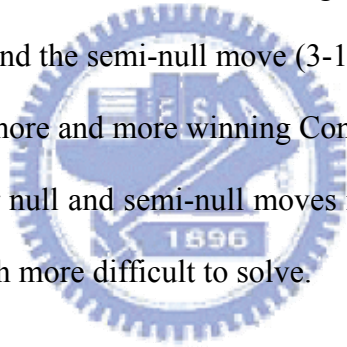


Figure 6. Black's winning move in Conect(6,2,3).

To solve it, they used a simple threat proof search method involving *null* or *semi-null* moves and *relevance zones*, as briefly described in Section 2.2. In the search method for solving the case *Connect(6,2,3)* with VCNT, both winning strategies for the null move (3-9 in Figure 12 (a) of Section 2.2) and the semi-null move (3-11 in Figure 12 (b) of Section 2.2) must be VCDT. However, with more and more winning Connect6 positions investigated, we found that winning strategies for null and semi-null moves may be VCSTs or even VCNTs, thus making these positions much more difficult to solve.



## 1.4 Motivation

Consider the two winning non-threat moves (proved in this thesis), moves 7 in Figure 7 (a) and 6 in Figure 7 (b), respectively.

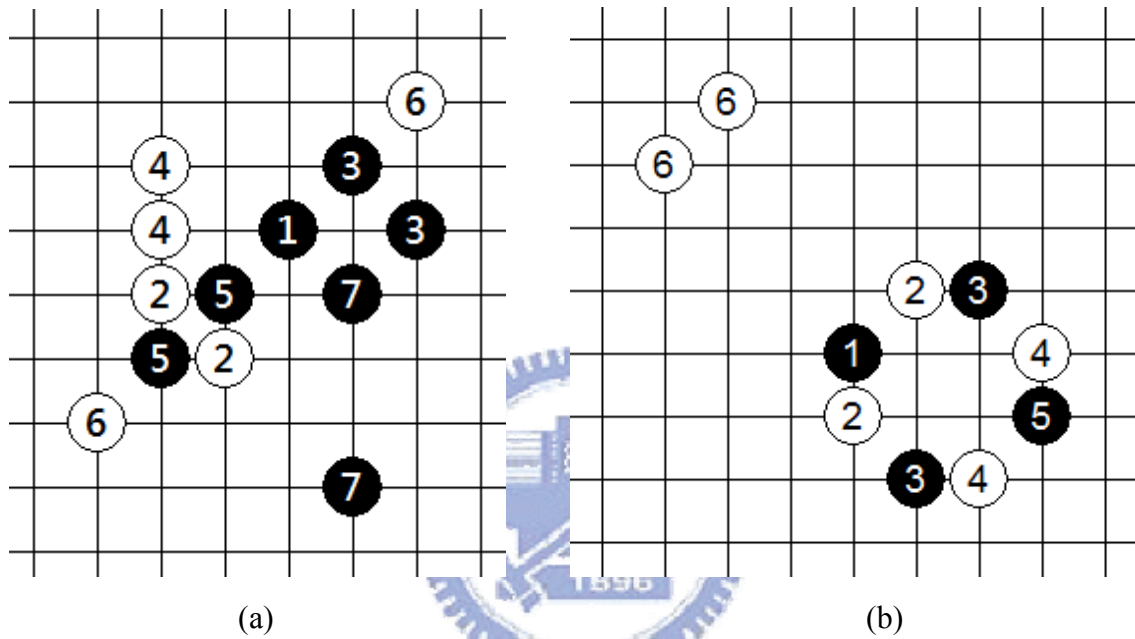


Figure 7. (a) A position with Black winning. (b) A position with White winning.

The former, found in 2006 [50], was the key used to help prove that Black wins at move 3 in Figure 7 (a); that is, the opening move 2 is solved. In this case, for the null move in Figure 7 (a), Black wins by a VCDT as shown in Figure 8 (a). However, for the semi-null move 8 in Figure 8 (b), Black has no double-threat moves to win by a VCDT, though Black wins by a VCST starting at 9 in Figure 8 (b).

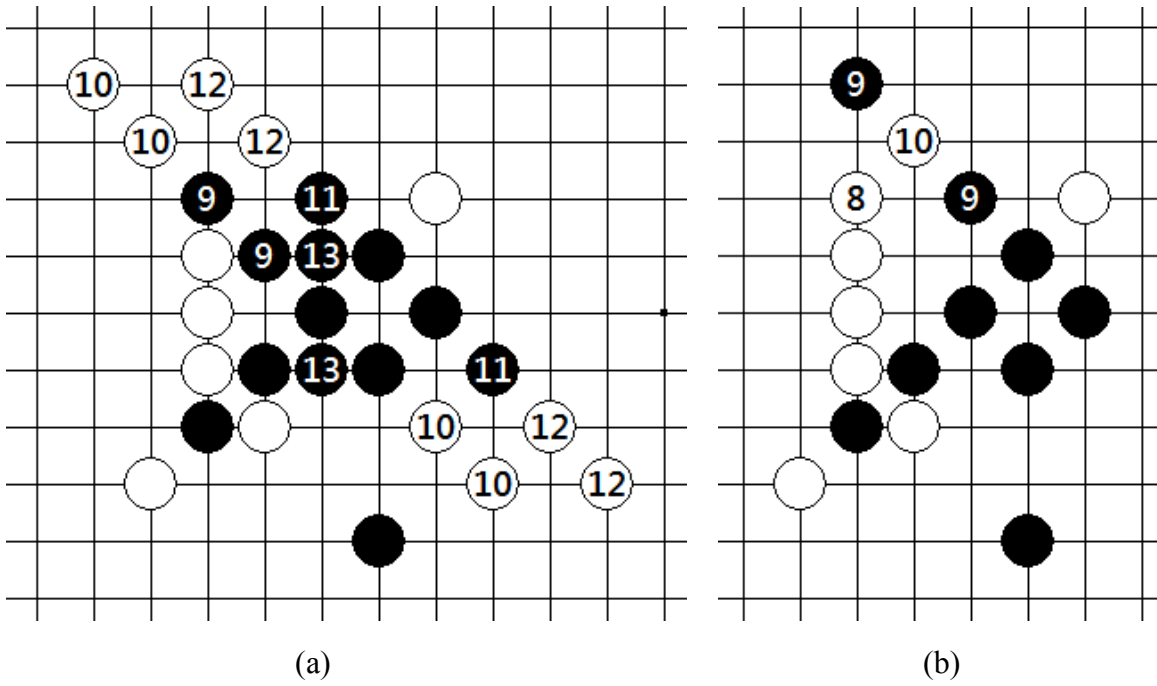


Figure 8. (a) A VCDT for the null move in Figure 7 (a). (b) A winning single-threat move 9 for the semi-null move 8.

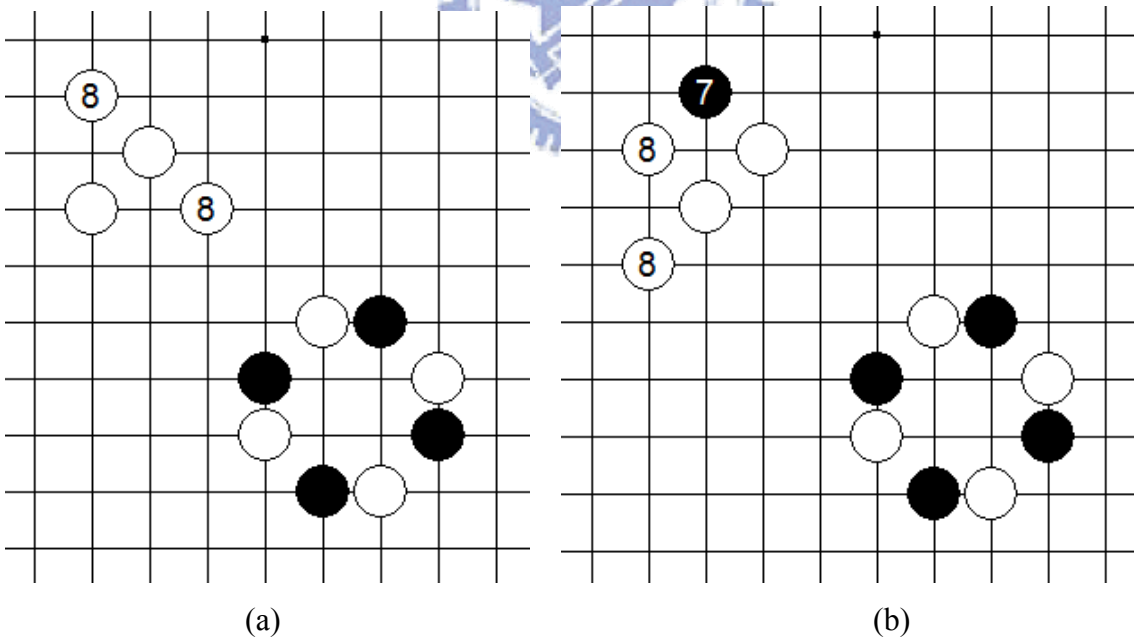


Figure 9. (a) A winning single-threat move 8 for a null move in Figure 7 (b). (b) A winning non-threat move 8 for a semi-null move 7.

The latter, the position in Figure 7 (b) found by Huang [25], was investigated to see whether the semi-null move 5 was safe enough, since the position at 5 was popular in the following sense. Among all the first-five-move positions of Connect6 games played by the players ranked above 1800 in [33] about 2% covered (or superset) the position according to the statistics discussed in [50]. The proof for this position is extremely complicated. Even for a null move by Black, White has no double-threat moves to win by a VCDT, but can actually win by a VCST starting at 8 as shown in Figure 9 (a). In addition, if a semi-null move is made at 7 in Figure 9 (b), White cannot win by a VCDT or even a VCST, thus making the position in Figure 7 (b) much more complicated to solve.

This thesis proposes a new threat-based proof search method in Chapter 3, named *Relevance-Zone-Oriented Proof (RZOP) search*, developed from the lambda search proposed by Thomsen [52]. The proposed RZOP method is also generalized to all Connect games in the Appendix D. In the past, many researchers [1][2][11][12][52] had proposed threat-based search methods. Lambda search is to formalize the search trees with null moves and to solve positions of games such as Go and Chess. In lambda search, null moves are involved with different orders of threat sequences, also called *lambda-trees*.

From the viewpoint of lambda search, a VCDT is a typical  $\lambda^1$ -tree with value 1 (cf. [52]). However, the definition of lambda search cannot be directly applied to Connect6 or Connect games with  $p \geq 2$ . For Connect games, this thesis modifies the definition of lambda search in Section 2.3, and replaces the notation  $\lambda^i$  by  $\Lambda^i$ . Under the new definition, a VCST is a  $\Lambda^2$ -tree with value 1, the winning strategy for the position in Figure 7 (a) is a  $\Lambda^3$ -tree with value 1, while that in Figure 7 (b) is a  $\Lambda^4$ -tree with value 1. The  $\Lambda$  search formalized in this thesis is able to solve  $\Lambda^l$ -trees to  $\Lambda^4$ -trees with value 1 for Connect6.

Related works are given in Chapter 2. Together with a proof number search [3][9][23][27][35][36][43][46][54][64], this thesis solved effectively and successfully many

new Connect6 game positions, including several Connect6 openings, especially the Mickey-Mouse Opening, as described in Section 5.3. This opening used to be one of the popular openings before we solved it.

Chapter 4 further presents an improved method, named *Segmented Relevance-Zone-Oriented Proof (SRZOP) search*, which speeds up the time to solve Connect6 game positions. Experiments are illustrated in Chapter 5, where the detail results are shown in Appendix A, B and C. Chapter 6 concludes this thesis. Appendix E explains draw k-in-a-row games. Appendix F demonstrates records of our Connect6 program *NCTU6*, which won the gold in the 11<sup>th</sup> and 13<sup>th</sup> Computer Olympiads [59][67] in 2006 and 2008, respectively; and also won eight games and lost none against top Connect6 players in Taiwan in 2009 [30].



## Chapter 2 Related Works

This chapter gives definitions and notation related to search trees and lambda search in Sections 2.1 and 2.2 respectively.

### 2.1 Search Trees

This thesis basically follows the definitions of search trees in [10][37]. A *search tree* is shown in Figure 10 (a) below, where rectangle and circle nodes indicate the positions in Attacker's and Defender's turns<sup>3</sup>, respectively. The value of a leaf is 1, if Attacker makes a win segment, and 0, otherwise. The value of a search tree is the minimax value of the tree. Attacker wins in the root position if the search tree has value 1 and all the internal circles expand all Defender's legal moves.

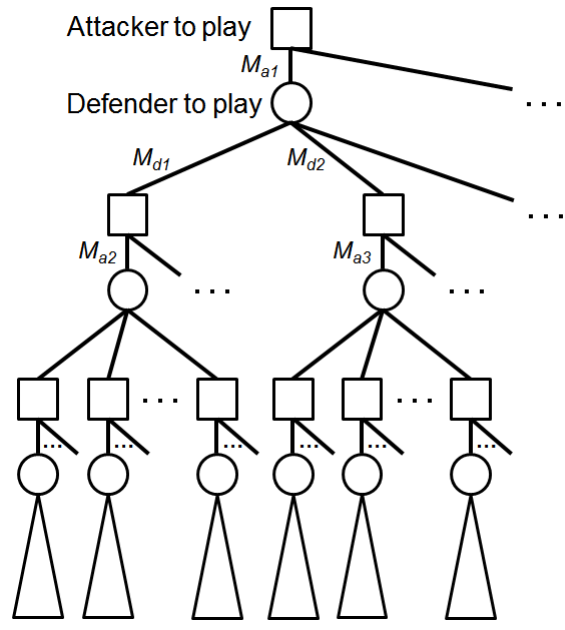
A *strategy*  $S$  of Attacker is viewed as a move-generating function of positions  $P$  that are in Attacker's turn. Naming,  $S(P)$  indicates the move that Attacker chooses to make according to the strategy  $S$ . In a *search tree following*  $S$ , each position  $P$  expands at most one move  $S(P)$ . A strategy  $S$  of Attacker is called a *winning strategy* for position  $P$ , if and only if the value of the search tree rooted at  $P$  is 1 following  $S$  and all Defender's legal moves are generated in the tree. Thus, we obtain Corollary 1 (below). A tree as shown in Figure 10 (b) is called a *solution tree* in [10][37].

**Corollary 1.** Attacker wins in a position  $P$  if and only if there exists at least one *winning strategy* of Attacker in  $P$ . ■

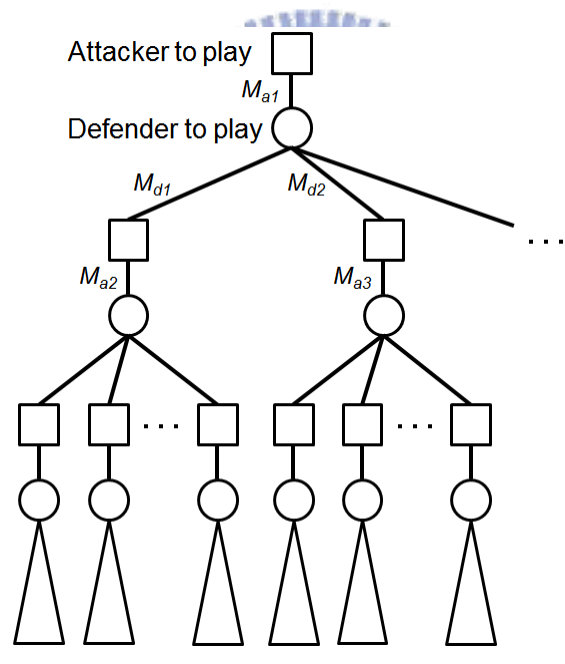
---

<sup>3</sup> When we say that a position  $P$  is in Attacker's (Defender's) turn, we mean that Attacker (Defender) is to move next in  $P$ .



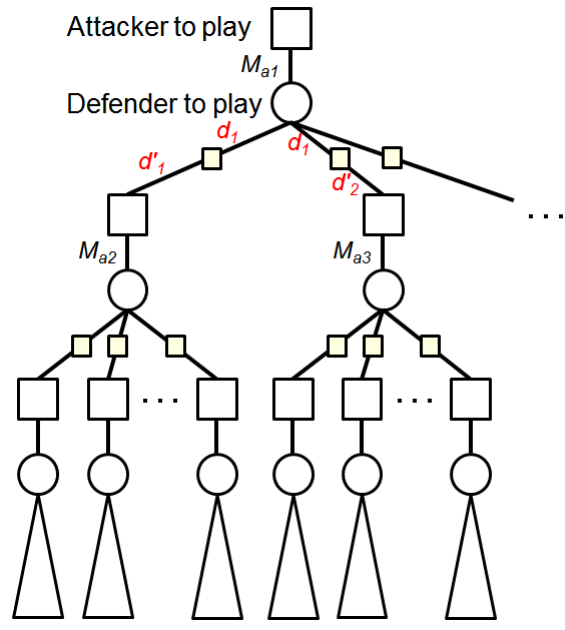


(a)

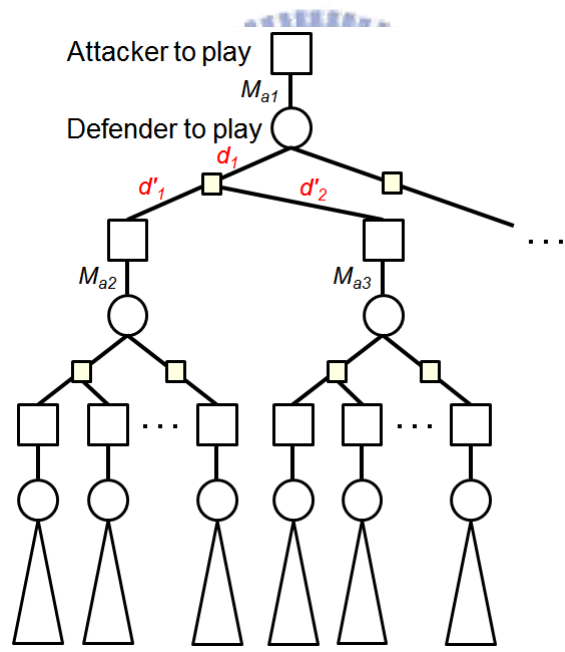


(b)

Figure 10. (a) A search tree and (b) a solution tree.



(a)



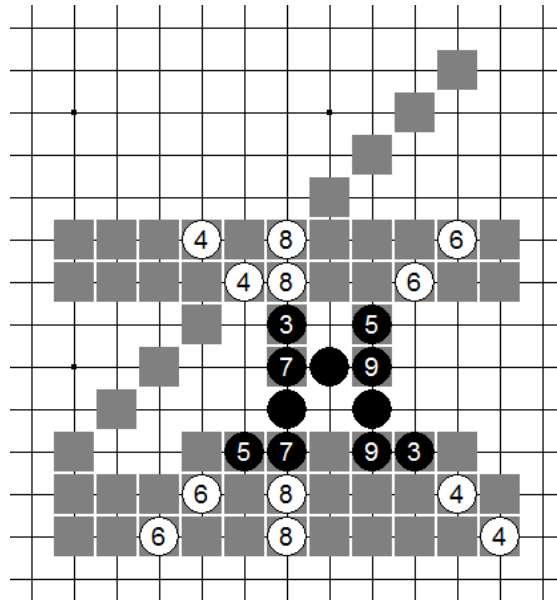
(b)

Figure 11. (a) Making squares of moves by inserting small boxes. (b) Combining the same edges from (a).

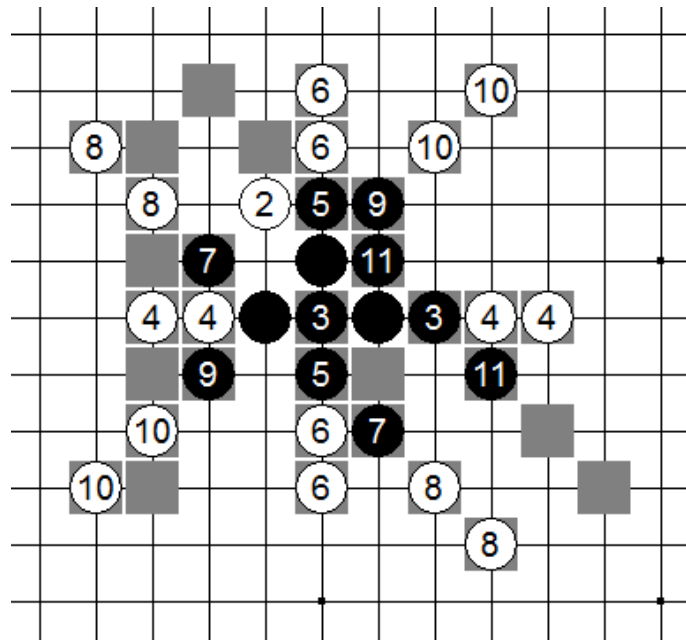
In order to investigate more closely squares of defensive moves, insert small rectangles onto the corresponding edges that are broken into two, marked  $s_1$  and  $s_2$  respectively, as shown in Figure 11 (a). Furthermore, the edges are combined with the same  $s_1$ , as shown in Figure 11 (b). Note that null stones are marked as  $\phi$  and the corresponding edges are indicated by *dashes*.

## 2.2 Null Move Heuristics for Connect6

To solve *Connect(6,2,3)*, Wu and Huang [65][66] used a simple threat proof search method involving *null* or *semi-null moves* and *relevance zones*, as briefly described in the following. Let White place no stones, called a null move in [65][66]. Obviously, Black wins by VCDT 3-9 as shown in Figure 12 (a) below. Then, a relevance zone  $Z$ , the area of gray squares in Figure 12 (a), can be derived to indicate that White must place at least one of the two stones inside this zone, or Black wins by simply *replaying* the same VCDT. Next, all squares  $s$  in  $Z$  are verified as follows. Let White place one stone on  $s$  only, called a semi-null move in [65][66]; for example, move 2 in Figure 12 (b). Again, Black is able to win by another VCDT 3-11. Thus, another relevance zone  $Z'$ , the gray area in Figure 12 (b), can be derived again to indicate that White must place another stone inside  $Z'$ , or Black wins by replaying the same VCDT. Finally, all  $s$  are verified such that Black wins over all moves placed at  $s$  and  $s'$ , where  $s'$  is in the  $Z'$  corresponding to the semi-null move at  $s$ . Hence, Black was proved to win.



(a)



(b)

Figure 12. (a) A VCDT for a null move in Figure 6. (b) A VCDT for a semi-null move 2.

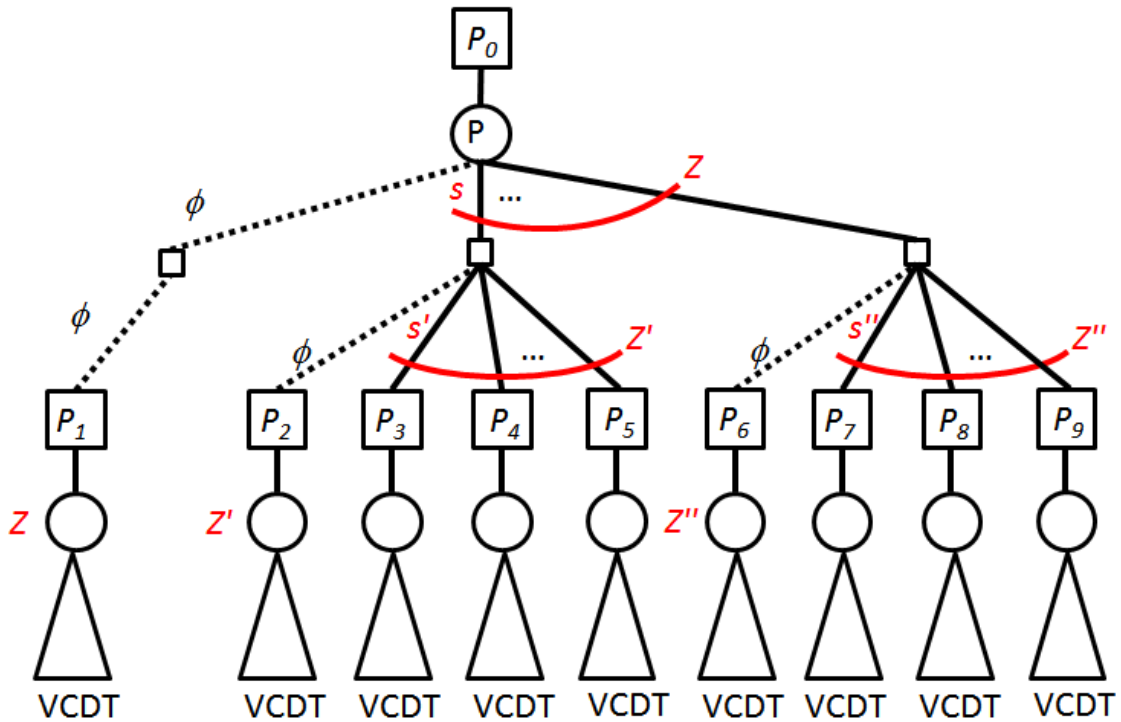


Figure 13. The proof search tree for solving  $Connect(6,2,3)$ .

A *verifier*  $V$  (for Attacker) is to verify whether Attacker wins in a position  $P$  by following a strategy  $S$ . Specifically, if  $V(P,S)$  returns the value 1, then Attacker wins in  $P$  and  $S$  is a winning strategy for  $P$ . A straightforward verifier is to verify it by traversing exhaustively the whole solution tree. Clearly, it is infeasible in most cases, especially in case of very large boards or even infinite boards. Fortunately, in Connect games, the traversal of the search tree for proof can be greatly reduced according to *threats*, as described in Chapter 1. The traversed search tree for proof by a verifier is called a *proof search tree*. The proof search tree for solving  $Connect(6,2,3)$  is shown in Figure 13.

## 2.3 Lambda Search for Connect6

In [52], Thomsen proposed using the lambda search to express how direct Attacker can achieve a goal. In Connect games, the goal is normally to make a win segment. The formalization of lambda search is modified for Connect games as follows.

**Definition 1.** In Connect games, a  $\Lambda^r$ -tree is a search tree which comprises all legal  $\Lambda^r$ -moves. If a  $\Lambda^r$ -move is an Attacker move, the following condition holds. For all subsequent null moves or semi-null moves  $M_D$  made by Defender, if  $M_D$  have exactly  $u$  null stones, where  $1 \leq u \leq p$ , there exists at least one subsequent  $\Lambda^i$ -tree with value 1, where  $0 \leq i \leq r - u$  or  $i = 0$  if  $r < u$ . If a  $\Lambda^r$ -move is a Defender move, the following condition holds. There exist no subsequent  $\Lambda^i$ -trees with value 1, where  $0 \leq i \leq r - 1$ . In a  $\Lambda^r$ -tree, a node is a leaf (without any children) if there are no  $\Lambda^r$ -moves following it. The value of a leaf is 1 if Defender is to move, and 0 if Attacker is to move. The value of a  $\Lambda^r$ -tree is either 1 (indicating that Attacker wins) or 0 (otherwise), derived using minimax calculation. The value of a  $\Lambda^0$ -tree (where Attacker to move) is simply 1 if Attacker makes a win segment in the next move. ■

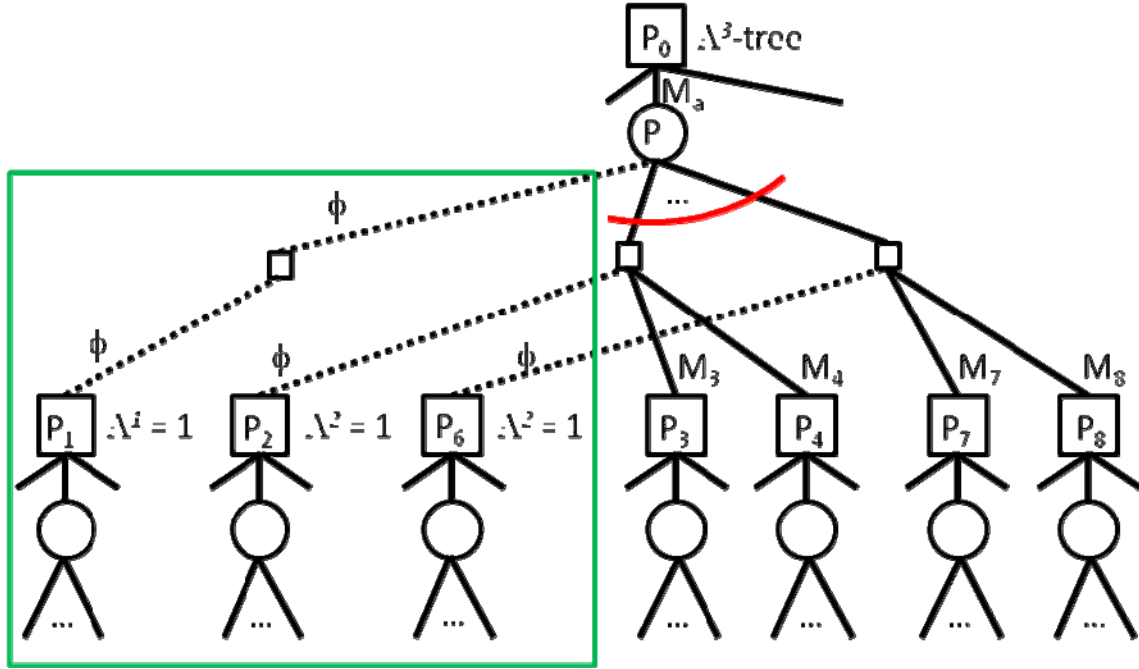


Figure 14. A  $\Lambda^3$ -tree.

In case of  $p = 1$ , the definition of  $\Lambda^r$  is the same as that of  $\lambda^r$  (the goal is to win) in [52]; that is, a  $\Lambda^r$ -tree is a  $\lambda^r$ -tree and a  $\Lambda^r$ -move is a  $\lambda^r$ -move, and *vice versa*. In case of  $p = 2$ , such as Connect6, a  $\Lambda^3$ -tree is illustrated in Figure 14 and move  $M_a$  in the tree is a  $\Lambda^3$ -move, since the values of  $\Lambda^1$ -tree and all  $\Lambda^2$ -trees in the left box are all 1. In addition, moves  $M_3$ ,  $M_4$ ,  $M_7$  and  $M_8$  are  $\Lambda^3$ -moves, if Attacker has no subsequent  $\Lambda^0$ -moves,  $\Lambda^1$ -moves or  $\Lambda^2$ -moves. By following the proof of Theorem 1 in [52], we derive the following theorem (whose proof is omitted).

**Theorem 1.** For a  $\Lambda^r$ -tree rooted in a position  $P$ , if a minimax search on it returns the value 1, Attacker wins in  $P$ . ■

**Definition 2.** A winning strategy is called a  $\Lambda^r$ -strategy for a position  $P$ , if the subsequent non-null moves following the strategy are all  $\Lambda^i$ -moves, where  $0 \leq i \leq r$ . ■

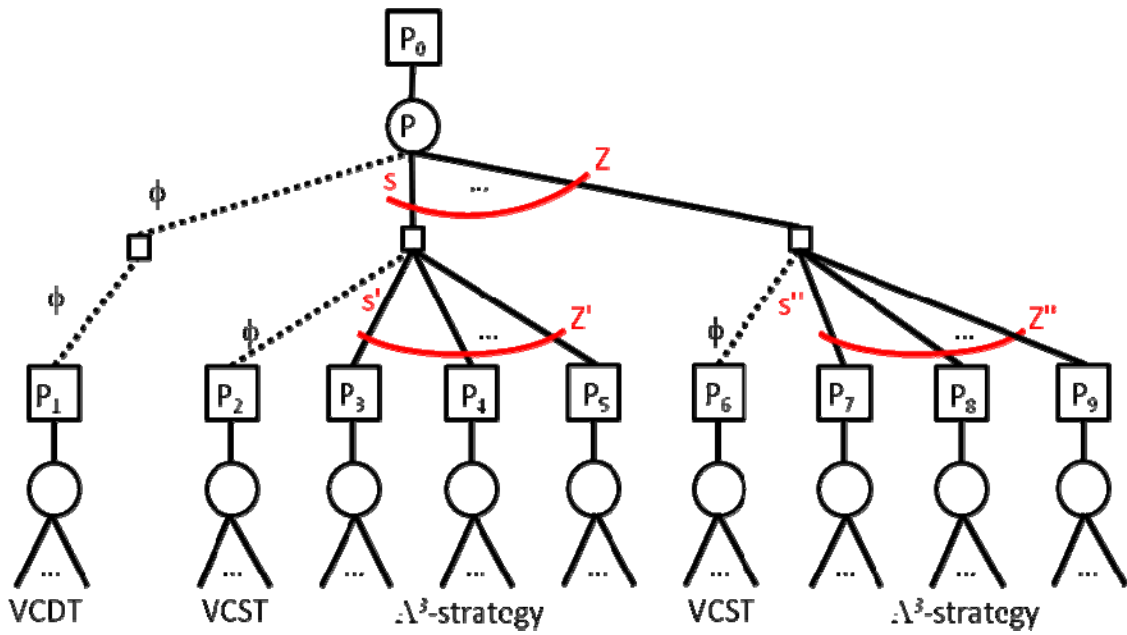


Figure 15. A  $\Lambda^3$ -strategy.

From the above definition, a VCDT is a  $\Lambda^1$ -strategy, while a VCST is a  $\Lambda^2$ -strategy. For example, there exists a  $\Lambda^2$ -strategy for winning position 7 in Figure 4 (Attacker is White), where moves 8 to 18 are all  $\Lambda^2$ -moves. VCNTs are  $\Lambda^3$ -strategies or strategies of higher orders, as illustrated in the following. In Figure 6, move  $M_{623}$  is a  $\Lambda^3$ -move, and the rest of Attacker moves are  $\Lambda^1$ -moves, so it is a  $\Lambda^3$ -strategy for *Connect6(6,2,3)*. In Figure 7 (a), move 7 is a  $\Lambda^3$ -move, and the rest of Attacker moves are  $\Lambda^1$ -moves or  $\Lambda^2$ -moves, so it is a  $\Lambda^3$ -strategy. Figure 15 shows a general  $\Lambda^3$ -strategy. However, it is more complicated in Figure 7 (b), where move 6 is a  $\Lambda^4$ -move. Section 5.2 shows that it is a  $\Lambda^4$ -strategy.

From Definition 2, a  $\Lambda^r$ -strategy,  $r \geq 1$ , also implies that for a move with  $u$  null stones Attacker has a  $\Lambda^{r-u}$ -strategy. For example, in the  $\Lambda^3$ -strategy in Figure 15, Attacker has a  $\Lambda^1$ -strategy for the null move and  $\Lambda^2$ -strategies for all the semi-null moves.



# Chapter 3 Relevance-Zone-Oriented Proof Search for Connect6

As seen in Section 2.3, the lambda search is a powerful method for proving the winning positions with different orders of threat sequences. The next important issue for lambda search is to construct relevance zones to reduce greatly the search space. In general, different applications construct relevance zones in different ways. In Connect games, it is critical to construct relevance zones in order to propagate relevance zones across different orders of threat sequences. For example, in Figure 15, the relevance zones derived in the VCDT ( $\Lambda^1$ -strategy) or VCSTs ( $\Lambda^2$ -strategies) can be used in the whole search tree ( $\Lambda^3$ -strategy). This chapter defines relevance zones and proposes the relevance-zone-oriented proof search for Connect6.

## 3.1 Relevance Zones

This section defines relevance zones, which are elegantly employed to solve Connect games. A set of squares on the board is called a *zone*. A sequence of zones with size  $r$ ,  $\Psi = \langle Z_1, Z_2, \dots, Z_r \rangle$ , is *incremental*, if the condition  $Z_1 \subseteq Z_2 \subseteq \dots \subseteq Z_r$  holds. In the rest of this thesis, sequences of zones with different sizes are all incremental and are thus not explicitly specified. In addition, these zones usually indicate the squares to be chosen for stones to be placed on, so only unoccupied (or empty) squares are of interest.

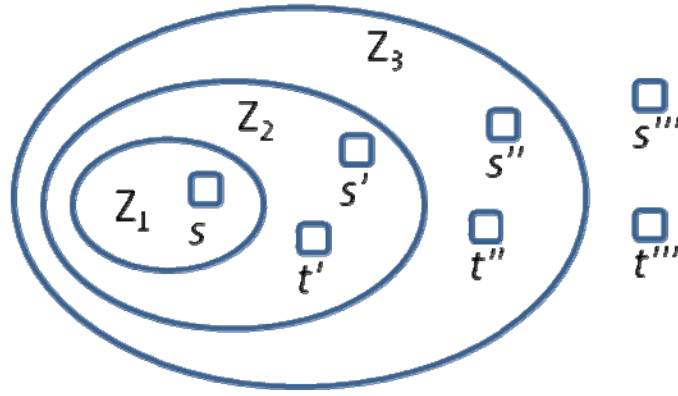


Figure 16. A sequence of zones  $\langle Z_1, Z_2, Z_3 \rangle$ .

In a position  $P$ , its *unoccupied zone*, denoted by  $Z_{un}(P)$ , is the zone that comprises all the unoccupied squares. That is,  $Z_{un}(P) = Z_{board} \setminus Z_P$ , where  $Z_{board}$  is the zone for the whole board and  $Z_P$  is the set of all occupied squares in  $P$ . Let  $\neg_P(Z)$  denote  $Z_{un}(P) \setminus Z$  and indicate the set of unoccupied squares *outside*  $Z$ . Consider a sequence of zones  $\Psi = \langle Z_1, Z_2, \dots, Z_r \rangle$  in  $P$ . A sequence of unoccupied squares  $\varphi = \langle s_{I_1}, s_{I_2}, \dots, s_{I_{r'}} \rangle$ , where  $r' \leq r$ , is said to be *outside*  $\Psi$  or *irrelevant* to  $\Psi$ , if all  $s_{I_i} \notin Z_i$  or  $s_{I_i} \in \neg_P(Z_i)$ . Let  $\varphi \in \neg_P(\Psi)$  denote the relation that  $\varphi$  is *irrelevant* to  $\Psi$  in  $P$ . Implicitly,  $\neg_P(\Psi)$  denotes  $\langle \neg_P(Z_1), \neg_P(Z_2), \dots, \neg_P(Z_r) \rangle$ . For example, in Figure 16,  $\langle s', s'', s''' \rangle$ ,  $\langle s', s'' \rangle$ ,  $\langle s'', s''' \rangle$ ,  $\langle s' \rangle$ ,  $\langle s''' \rangle$  and even the empty sequence  $\langle \rangle$  are all irrelevant to  $\langle Z_1, Z_2, Z_3 \rangle$ , while  $\langle s \rangle$ ,  $\langle s', t' \rangle$ ,  $\langle s', s'', t' \rangle$ ,  $\langle s', s'', s''', t' \rangle$ ,  $\langle s'', s' \rangle$ ,  $\langle s, s', s'' \rangle$  are not. For simplicity, let  $\sigma_A(\varphi)$  denote  $\sigma_A(s_{I_1}) + \sigma_A(s_{I_2}) + \dots + \sigma_A(s_{I_{r'}}) = \sum_{1 \leq i \leq r'} \sigma_A(s_{I_i})$ . Similarly,  $\sigma_D(\varphi) = \sum_{1 \leq i \leq r'} \sigma_D(s_{I_i})$ .

**Definition 3.** A sequence of zones  $\Psi$  is called a *sequence of relevance zones* for Attacker in a position  $P$ , if and only if Attacker wins in  $P + \sigma_D(\varphi)$  for all irrelevant  $\varphi$ ; that is,  $\varphi \in \neg_P(\Psi)$ . Let  $RZ(P)$  denote the set of all the sequences of relevance zones for Attacker in  $P$ . (Use the notation  $RZ(P)$  instead of  $RZ_A(P)$ , since only relevance zones for Attacker are discussed in this thesis). ■

From Definition 3, if  $RZ(P)$  is not empty, there must exist some  $\Psi$  in  $RZ(P)$ . This implies that Attacker wins in  $P$  by choosing the empty sequence of squares  $\langle \rangle$  for  $\varphi$ , since  $\varphi$  is irrelevant to  $\Psi$  as described above. Thus, Corollary 2 is obtained.

**Corollary 2.** If there exists at least one sequence of zones  $\Psi$  in  $RZ(P)$ , then Attacker wins in  $P$ . ■

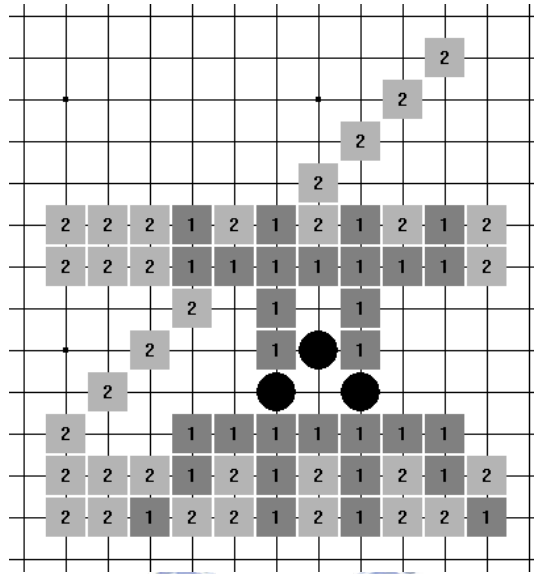


Figure 17. A sequence of relevance zones  $\Psi = \langle Z_1, Z_2 \rangle$  for the winning position in Figure 12 (a).

For the winning sequence in Figure 12 (a), Figure 17 illustrates relevance zones  $\Psi = \langle Z_1, Z_2 \rangle$ , where  $Z_1$  is the set of empty squares marked with a small “1”, and  $Z_2$  marked “1” and “2”. Note that in the rest of this thesis, a sequence of zones is shown in this manner. Interestingly,  $Z_2$  is the same as  $Z$  in Figure 12 (a). From observation, Black still wins over all irrelevant  $\varphi \in \neg_P(\Psi)$ . That is, if White places one in  $\neg_P(Z_1)$  and the other in  $\neg_P(Z_2)$ , Black still wins by replaying the winning sequence in Figure 12 (a). The result is slightly stronger than that in [65][66].

Lemma 1 shows an important property that appending extra  $Z_{board}$  to a sequence of

relevance zones is still in  $RZ(P)$ . Note that we use  $Z_{board}$ , instead of  $Z_{un}(P)$ , in order to be independent of the position  $P$ , for simplicity. For example, in Figure 17,  $\langle Z_1, Z_2, Z_{board} \rangle$  is also in  $RZ(P)$ .

**Lemma 1.** Assume that  $\Psi = \langle Z_1, Z_2, \dots, Z_r \rangle$  is in  $RZ(P)$ . Then,  $\Psi' = \langle Z_1, Z_2, \dots, Z_r, Z_{board} \rangle$  is also in  $RZ(P)$ .

**Proof.** Consider all irrelevant  $\varphi \in \neg_P(\Psi')$ . For this lemma, it suffices to prove that Attacker wins in  $P + \sigma_D(\varphi)$ . Since  $\neg_P(Z_{board})$  is empty,  $\varphi$  must not have the  $(r + 1)$ -st item. From the definition, we also obtain  $\varphi \in \neg_P(\Psi)$ . Since  $\Psi$  is assumed to be in  $RZ(P)$ , Attacker wins in  $P + \sigma_D(\varphi)$  due to  $\varphi \in \neg_P(\Psi)$ . ■

From Lemma 1, two sequences of relevance zones with different sizes can be adjusted to those with the same size by appending extra  $Z_{board}$  or removing  $Z_{board}$  at the end. For simplicity of discussion, this thesis uses some more notation for operations on sequences of zones with the same size in  $P$ , say  $\Psi = \langle Z_1, Z_2, \dots, Z_r \rangle$  and  $\Psi' = \langle Z'_1, Z'_2, \dots, Z'_r \rangle$ , as follows.

- Let  $\Psi \subseteq \Psi'$  indicate that  $\Psi$  is contained in  $\Psi'$  *pair wise*; that is,  $Z_i \subseteq Z'_i$  over all  $1 \leq i \leq r$ .
- Let  $\Psi \cup \Psi' = \langle Z_1 \cup Z'_1, Z_2 \cup Z'_2, \dots, Z_r \cup Z'_r \rangle$ .
- Let  $\Psi \cup Z = \langle Z_1 \cup Z, Z_2 \cup Z, \dots, Z_r \cup Z \rangle$  and  $\Psi \setminus Z = \langle Z_1 \setminus Z, Z_2 \setminus Z, \dots, Z_r \setminus Z \rangle$ , where  $Z$  is a zone.
- Let  $\Psi \ll 1$  denote  $\langle Z_2, Z_3, \dots, Z_r, Z_{board} \rangle$  and indicate promotion of the zones in  $\Psi$  (that is, shifting zones to the left by 1) with extra  $Z_{board}$ . Similarly, let  $\Psi \ll 2$  denote  $(\Psi \ll 1) \ll 1$ , and  $\Psi \ll i$  denote  $(\Psi \ll (i-1)) \ll 1$ , where  $i \geq 2$ .

From the above notation and definitions, more properties are shown in Lemma 2 and Lemma 3 as follows.

**Lemma 2.** Assume that  $\Psi$  is in  $RZ(P)$  and  $\Psi \subseteq \Psi'$ . Then,  $\Psi'$  is also in  $RZ(P)$ .

**Proof.** Let  $\Psi = \langle Z_1, Z_2, \dots, Z_r \rangle$  and  $\Psi' = \langle Z'_1, Z'_2, \dots, Z'_r \rangle$ . Consider all irrelevant  $\varphi \in \neg_P(\Psi')$ . It suffices to prove that Attacker wins in  $P + \sigma_D(\varphi)$ . Since  $\Psi \subseteq \Psi'$ , the condition  $\varphi \in \neg_P(\Psi')$  also implies  $\varphi \in \neg_P(\Psi)$ . Since  $\Psi$  is in  $RZ(P)$ , Attacker wins in  $P + \sigma_D(\varphi)$  due to  $\varphi \in \neg_P(\Psi)$ . ■

Lemma 3 (below) shows important properties that are employed to improve the verifiers in Section 3.2.

**Lemma 3.** Assume that  $\Psi = \langle Z_1, Z_2, \dots, Z_r \rangle$  is in  $RZ(P)$ . The following two properties are satisfied.

1. Assume that  $\neg_P(Z_1)$  is not empty. Let the unoccupied square be  $s \in \neg_P(Z_1)$ . Then,  $\Psi \ll 1$  is in  $RZ(P + \sigma_D(s))$ .
2. Let  $\varphi$  be a sequence of unoccupied squares  $\langle s_1, s_2, \dots, s_{r'} \rangle$  in  $\neg_P(\Psi)$ , where  $r' \leq r$ . Then,  $\Psi \ll r'$  is in  $RZ(P + \sigma_D(\varphi))$ .

**Proof.** It suffices to prove the first property, since the first implies the second by induction.

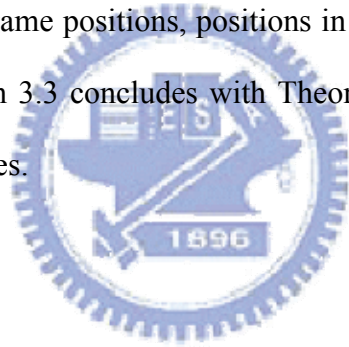
Let  $\Psi' = \Psi \ll 1$  and consider all irrelevant  $\varphi' = \langle s_2, \dots, s_{r'} \rangle \in \neg_P(\Psi')$ , where  $r' \leq r$ . For the first property, it suffices to prove that Attacker wins in  $(P + \sigma_D(s)) + \sigma_D(\varphi')$ . Let  $\varphi = \langle s, s_2, \dots, s_{r'} \rangle$ . Then, the condition  $\varphi \in \neg_P(\Psi)$  holds due to  $s \in \neg_P(Z_1)$ . Since  $\Psi$  is in  $RZ(P)$ , Attacker wins in  $P + \sigma_D(\varphi)$  due to  $\varphi \in \neg_P(\Psi)$ ; that is, Attacker wins in  $(P + \sigma_D(s)) + \sigma_D(\varphi')$  ( $= P + \sigma_D(\varphi)$ ). ■

## 3.2 The Proposed Verifier $V_{C6}$

For solving positions in Connect6, this section investigates a verifier  $V(P,S)$  that also construct recursively a sequence of zones  $\Psi(P) = \langle Z_1(P), Z_2(P), \dots, Z_r(P) \rangle$  with the following property.

**Property RZV:** In the case that  $V(P,S)$  returns the value 1, the sequence of zones  $\Psi(P)$  constructed by  $V(P,S)$  is in  $RZ(P)$ .

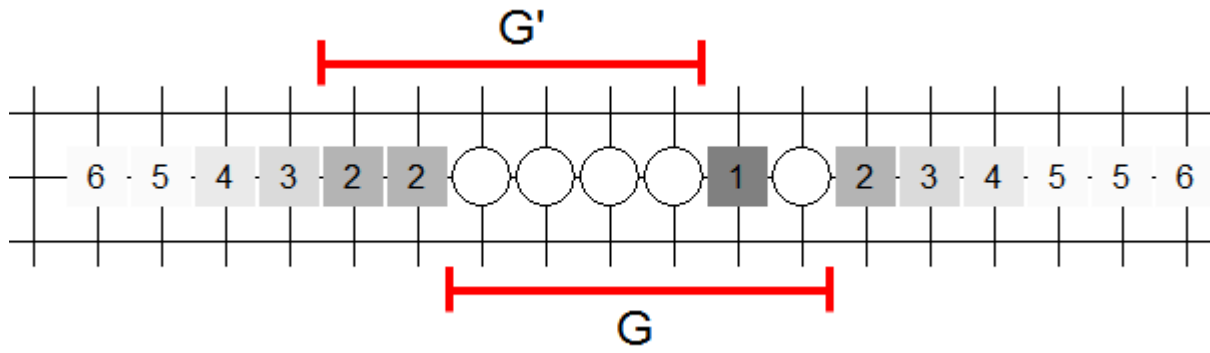
This section presents such a verifier, named  $V_{C6}(P,S)$ , with a new proof search method for Connect6. This method will be generalized to all Connect games in Appendix D. The verifier  $V_{C6}(P,S)$  is described in Subsections 3.2.1, 3.2.2 and 3.2.3 respectively for three distinct kinds of  $P$ , namely endgame positions, positions in Attacker's turn and positions in Defender's turn. Finally, Section 3.3 concludes with Theorem 2, showing that the verifier satisfies Property RZV in all cases.



### 3.2.1 Endgame Positions

If Attacker does not win in the endgame position  $P$ , the verifier simply returns the value 0. If Attacker wins in  $P$  (i.e., Attacker has a win segment in  $P$ ), the verifier returns 1 and constructs  $\Psi(P)$  in the following operation.

**EP-1.** For each active segment  $G$  of Defender containing *exactly*  $i$  unoccupied squares, these squares in  $G$  are all added into  $Z_i(P)$  or higher-order zones; that is,  $Z_j(P)$  for all  $j \geq i$ . In other words, for each active segment  $G$  of Defender containing at most  $i$  unoccupied squares, add all of these squares in  $G$  into  $Z_i(P)$ .



(a)

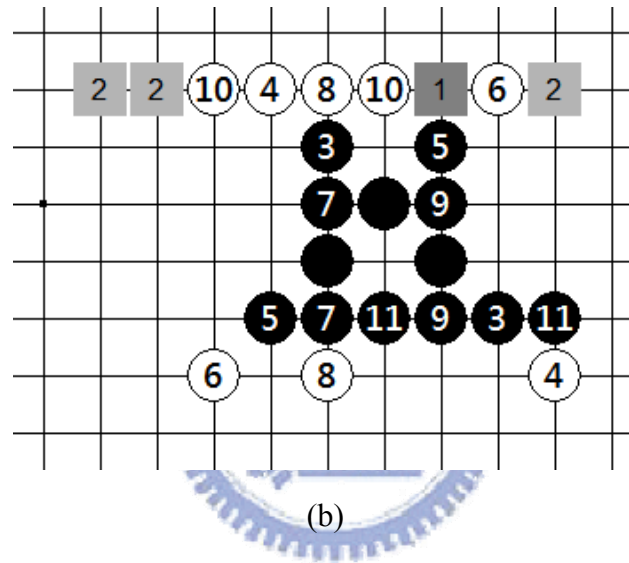


Figure 18. (a) Relevance zones in a line and (b) in a board, upon winning with a win segment.

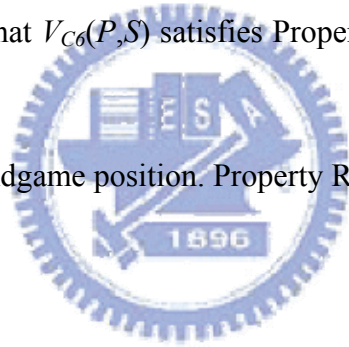
Let us illustrate the above operation by the line shown in Figure 18 (a), where Defender is White. Following the operation, the square marked with “1” is in  $Z_1$ , those marked with “1” or “2” are in  $Z_2$ , and so on. For example, segment  $G$  has only one unoccupied square that is in  $Z_1$  or higher-order zones, while segment  $G'$  has two unoccupied squares that are in  $Z_2$  or higher-order zones. It is observed that placing one white stone on the square in  $Z_1$  forms a *counter win segment* (e.g.,  $G$ ) or an *inversion* that may prevent Attacker from winning. Note that if Defender has an inversion, this position  $P$  is

unreachable since neither can have win segments simultaneously (as described in the previous section), who wins first is thus unknown. On the other hand, Attacker still wins if one white stone is placed in square  $s_1$ , where  $s_1 \notin Z_1$ . Similarly, Attacker still wins if one white stone is placed on  $s_1$ , where  $s_1 \notin Z_1$ , and the other on  $s_2$ , where  $s_2 \notin Z_2$ . The above can be generalized to higher orders, and to all lines (or segments) on a board. An example of constructing zones  $\langle Z_1, Z_2 \rangle$  on a board is illustrated in Figure 18 (b). Note that move 10 in the figure is simply one of all defenses and is chosen for illustration. In addition, since move 9 clearly wins already, Subsection 3.2.3 will describe how to speed up the establishment of relevance zones.

From the above observation, it can be derived that the constructed  $\Psi(P)$  in operation EP-1 is in  $RZ(P)$ . This implies that  $V_{C6}(P,S)$  satisfies Property RZV in the case of endgame  $P$ , as shown in Lemma 4.

**Lemma 4.** Assume  $P$  to be an endgame position. Property RZV is satisfied for  $V_{C6}(P,S)$ .

**Proof.** Omitted. ■



In Connect6, all  $Z_i(P)$  with  $i \geq 6$ , are nearly the same as  $Z_{un}(P)$ , except for those unoccupied squares covered by none of active segments of Defender. For example, if an unoccupied square is surrounded by Attacker's squares, it is clearly covered by none of active segments of Defender and is not included in these  $Z_i(P)$ . However, such squares are normally not many, especially when board sizes are large and only a small number of stones are in positions. Practically, we simply ignore all  $Z_i(P)$  with  $i \geq 6$  or use  $Z_{un}(P)$  whenever needed.



### 3.2.2 Positions in Attacker's Turn

In such positions, Attacker simply follows strategy  $S$  to make the move  $S(P)$  in  $P$ . Let  $P_A$  denote  $P \oplus S(P)$ . This verifier first performs  $V_{C6}(P_A, S)$  recursively. If  $V_{C6}(P_A, S)$  returns the value 0, this verifier  $V_{C6}(P, S)$  also returns 0. On the other hand, if  $V_{C6}(P_A, S)$  returns 1, this verifier  $V_{C6}(P, S)$  returns 1, too; and constructs  $\Psi(P)$  in the following operation.

**AT-1.** Let  $\Psi(P) = \Psi(P_A) \cup Z_S$ , where  $Z_S = \{s \mid s \in S(P)\}$ .

Intuitively, placing any stones on the squares in  $Z_S$  by Defender in advance may block attacks and prevent Attacker from winning. In this sense, the squares in  $Z_S$  are relevant and are therefore contained in all  $Z_i(P)$  (or  $\Psi(P)$ ).

In fact, the above operation AT-1 also implies the property,  $\neg_P \Psi(P) = \neg_{P_A} \Psi(P_A)$ , for the following reason. From the operation, the condition  $Z_i(P) = Z_i(P_A) \cup Z_S$  holds for all  $i$ . In addition, since  $P_A = P \oplus S(P)$ , it is clear that  $Z_{un}(P_A) = Z_{un}(P) \setminus Z_S$  or  $Z_{un}(P) = Z_{un}(P_A) \cup Z_S$ . Thus, for all  $i$ , we derive  $\neg_P Z_i(P) = Z_{un}(P) \setminus Z_i(P) = (Z_{un}(P_A) \cup Z_S) \setminus (Z_i(P_A) \cup Z_S) = Z_{un}(P_A) \setminus Z_i(P_A) = \neg_{P_A} Z_i(P_A)$ . From this property, Lemma 5 (below) shows that this verifier  $V_{C6}(P, S)$  satisfies Property RZV if  $V_{C6}(P_A, S)$  satisfies Property RZV.

**Lemma 5.** Assume a position  $P$  in Attacker's turn. From the above, assume that  $V_{C6}(P_A, S)$  satisfies Property RZV, where  $P_A = P \oplus S(P)$ . This verifier  $V_{C6}(P, S)$  satisfies Property RZV.

**Proof.** Assume that this verifier  $V_{C6}(P, S)$  returns the value 1. For this lemma (this verifier satisfies Property RZV), it suffices to prove that the constructed  $\Psi(P)$  is in  $RZ(P)$ . From the above operation,  $V_{C6}(P_A, S)$  must also return 1. Since  $V_{C6}(P_A, S)$  satisfies Property RZV from the lemma,  $\Psi(P_A)$  is in  $RZ(P_A)$ .

Consider all irrelevant  $\phi$ , where  $\phi \in \neg_P \Psi(P)$ . It suffices to prove that Attacker wins in  $P + \sigma_D(\phi)$ . Since the property  $\neg_P \Psi(P) = \neg_{P_A} \Psi(P_A)$  is satisfied as described above, the

condition  $\varphi \in \neg_{PA} \Psi(P_A)$  holds too. Since  $\Psi(P_A)$  is in  $RZ(P_A)$  from above, Attacker wins in  $P_A + \sigma_D(\varphi)$  due to  $\varphi \in \neg_{PA} \Psi(P_A)$ . Since Attacker wins in  $P_A + \sigma_D(\varphi) = (P + \sigma_D(\varphi)) \oplus S(P)$ , Attacker wins in  $P + \sigma_D(\varphi)$  by choosing the move  $S(P)$ . ■

### 3.2.3 Positions in Defender's Turn

For positions in Defender's turn, Lemma 6 shows a very important property used in this section as well as the Appendix.

**Lemma 6.** Assume a position  $P$  in Defender's turn. For a given sequence of zones  $\Psi$ , assume that for all Defender moves  $M_D$  there exists some  $\Psi_D$  such that  $\Psi_D \subseteq \Psi$  and  $\Psi_D$  is in  $RZ(P \oplus M_D)$ . Then  $\Psi$  is in  $RZ(P)$ .

**Proof.** Consider all irrelevant  $\varphi \in \neg_P \Psi$ . For this lemma, it suffices to prove that Attacker wins in  $P + \sigma_D(\varphi)$ .

Now, consider all Defender moves  $M_D$  in  $P + \sigma_D(\varphi)$ . From this lemma, there exists some  $\Psi_D$  such that  $\Psi_D \subseteq \Psi$  and  $\Psi_D$  is in  $RZ(P \oplus M_D)$ . Since  $\Psi_D \subseteq \Psi$ , the condition  $\varphi \in \neg_P \Psi$  implies  $\varphi \in \neg_P \Psi_D$ . Since squares in  $M_D$  and  $\sigma_D(\varphi)$  are mutually exclusive,  $\varphi \in \neg_P \Psi_D$  also implies  $\varphi \in \neg_{P \oplus M_D} \Psi_D$ . Since  $\Psi_D$  is in  $RZ(P \oplus M_D)$  from above, Attacker wins in  $(P \oplus M_D) + \sigma_D(\varphi)$  due to  $\varphi \in \neg_{P \oplus M_D} \Psi_D$ . Since  $(P \oplus M_D) + \sigma_D(\varphi) = (P + \sigma_D(\varphi)) \oplus M_D$ , Attacker also wins in  $(P + \sigma_D(\varphi)) \oplus M_D$ . From the above, since Attacker wins in  $(P + \sigma_D(\varphi)) \oplus M_D$  over all Defender moves  $M_D$ , Attacker wins in  $P + \sigma_D(\varphi)$ . ■

A straightforward verifier is to verify whether Attacker wins for all Defender moves, as follows. The verifier  $V_{C\delta}(P, S)$  returns the value 1, if the recursive  $V_{C\delta}(P \oplus M_D, S)$  returns 1 for all Defender moves  $M_D$ ; otherwise, it returns 0. In the case that this verifier  $V_{C\delta}(P, S)$  returns 1, the zones  $\Psi(P)$  are constructed in the following operation.

**DT-1.** Initialize all zones in  $\Psi(P)$  to be empty. Then, for all Defender moves  $M_D$ , let  $\Psi(P) = \Psi(P) \cup \Psi(P \oplus M_D)$ .

From the above operation, the condition  $\Psi(P \oplus M_D) \subseteq \Psi(P)$  clearly holds for all  $M_D$ . Assume that all the recursive  $V_{C6}(P \oplus M_D, S)$  satisfy Property RZV. Then, all  $\Psi(P \oplus M_D)$  are in  $RZ(P \oplus M_D)$  for all Defender moves  $M_D$ . From Lemma 6, we obtain that  $\Psi(P)$  is in  $RZ(P)$ ; and therefore, the verifier satisfies Property RZV. By induction, the above straightforward verifier satisfies Property RZV in all cases.

However, the above straightforward verifier is apparently inefficient, since it searches exhaustively all Defender moves, even when Attacker moves have some threats. The situation is even worse in the case that the board size is very large or infinite. In this subsection, an efficient and elegant verifier is devised to reduce the search space by making use of both threats and relevance zones. In Connect6, the position  $P$  (in Defender's turn) can be classified into the following four cases. The number of Attacker threats in  $P$  is (1) three or more, (2) two, (3) one and (4) zero. The four cases are discussed respectively in the following four subsections.

### 3.2.3.1. Three Threats or More

In this case, Attacker is sure to win by simply following the strategy,  $S_{3T}$ , as follows. For each Defender move, since the move must leave some threat segments unblocked, Attacker wins simply by making a win segment from the unblocked one. Since the strategy is a sure win, the verifier returns the value 1 and constructs the zones (initialized to be empty) in the following operations.

- T3-1.** Add all unoccupied squares  $s$  on threat segments into all  $Z_i(P)$ .
- T3-2.** For each active segment  $G$  of Defender containing *exactly*  $i + 2$  unoccupied squares, all these squares in  $G$  are added into all  $Z_j(P)$  or higher-order zones. In other words, for each active segment  $G$  of Defender containing at most  $i + 2$  unoccupied squares, add all these squares in  $G$  into  $Z_i(P)$ .

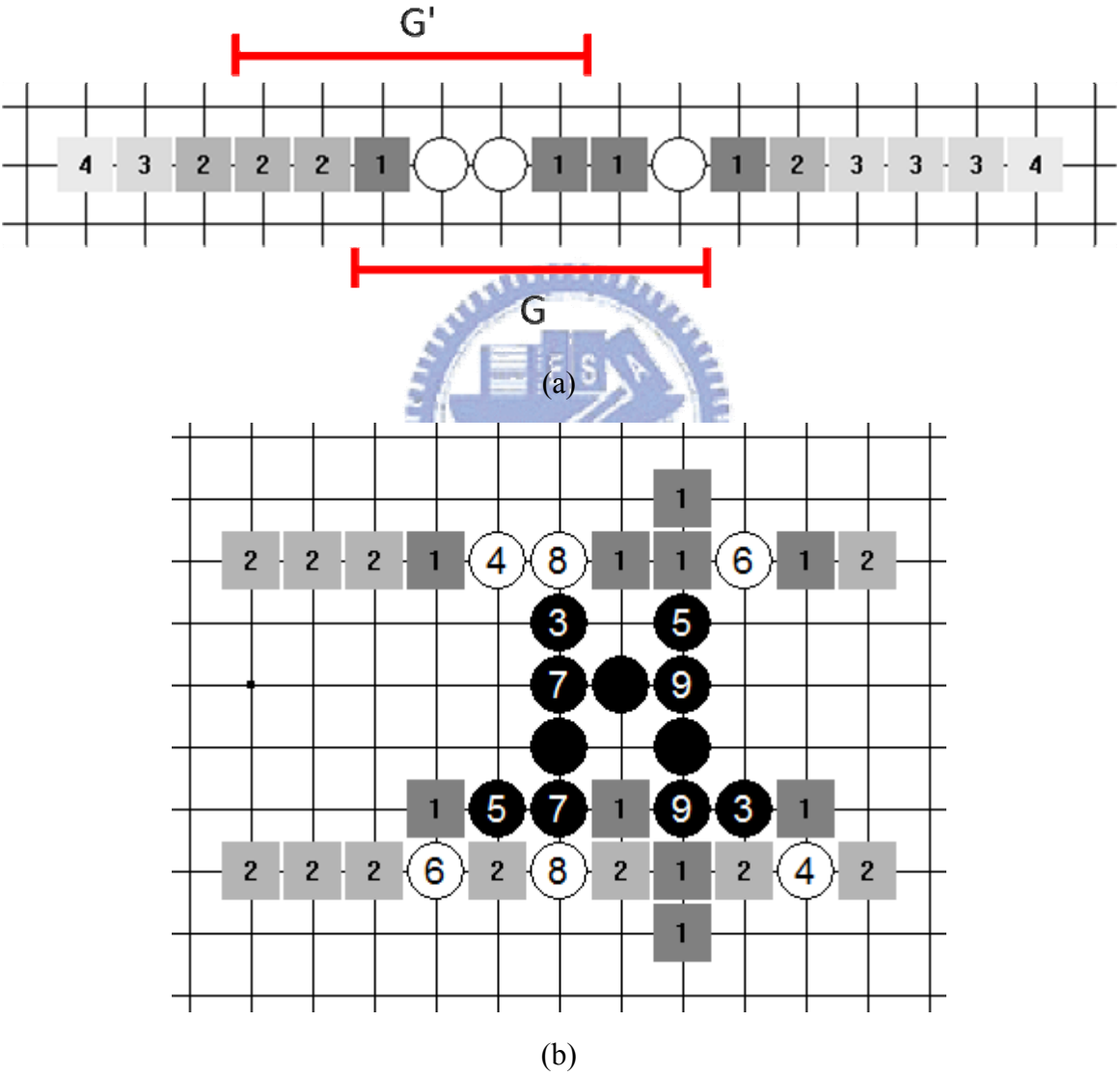


Figure 19. (a) Relevance zones in a line and (b) in a board, upon winning with three or more threats.

Let us illustrate the above operations by the line shown in Figure 19 (a), where Defender is White. Zones in the line are marked in a way similar to that in Figure 18 (a). It is observed that placing one white stone in  $G$  or  $Z_1$  results in a *counter threat segment* or an *inversion* that *may* threaten Attacker to defend in some of his earlier moves and prevent Attacker from winning. On the other hand, Attacker still wins if one white stone is placed on other squares  $s_1$ , where  $s_1 \notin Z_1$ . Similarly, Attacker still wins if one white stone is placed on  $s_1$ , where  $s_1 \notin Z_1$ , and the other on  $s_2$ , where  $s_2 \notin Z_2$ . The above can be generalized to higher orders, and to all lines (or segments) on the board. An example of constructing two zones  $\langle Z_1, Z_2 \rangle$  on a board is illustrated in Figure 19 (b). Lemma 7 shows that in this case the verifier satisfies Property RZV; that is,  $\Psi(P)$  is in  $RZ(P)$ .

**Lemma 7.** Assume that Defender is to move and Attacker has three or more threats in  $P$ . The verifier described above satisfies Property RZV.

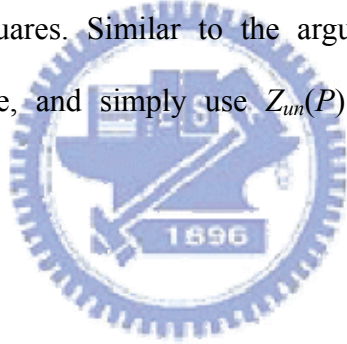
**Proof.** For this lemma, it suffices to prove that the constructed  $\Psi(P)$  is in  $RZ(P)$ . Consider all Defender moves  $M_D$ . Attacker simply follows a strategy  $S_{3T}$  to connect six from an unblocked threat segment. Let  $P_D = P \oplus M_D$  and  $P_6 = P_D \oplus S_{3T}(P_D)$ . From Lemma 4 and Lemma 5,  $\Psi(P_6)$  and  $\Psi(P_D)$  are in  $RZ(P_6)$  and  $RZ(P_D)$ , respectively.

To prove that  $\Psi(P)$  is in  $RZ(P)$ , it suffices to prove from Lemma 6 that  $\Psi(P_D) \subseteq \Psi(P)$ , since  $\Psi(P_D)$  is already in  $RZ(P_D)$ . From Subsection 3.2.2,  $\Psi(P_D) = \Psi(P_6) \cup Z_S$ , where  $Z_S = \{s \mid s \in S_{3T}(P_D)\}$ . From operation T3-1, all squares in  $Z_S$  are added into  $\Psi(P)$ . Thus, it suffices to prove that  $\Psi(P_6) \subseteq \Psi(P)$ .

Since Attacker connects six in  $P_6$ , operation EP-1 (in Subsection 3.2.1) is employed to construct zones  $\Psi(P_6)$ . The operation is restated as follows. For each active segment  $G$  of Defender containing at most  $i$  unoccupied squares in  $P_6$ , all the squares in  $G$  are added into  $Z_i(P_6)$ . Since one move has at most two squares, at most two occupied squares in  $G$  were occupied by move  $M_D$ . Therefore,  $G$  contains at most  $2 + i$  unoccupied squares back in  $P$

(before making move  $M_D$ ). From operation T3-2, all these unoccupied squares are also added into  $Z_i(P)$ . For example, let both lines in Figure 18 (a) and Figure 19 (a) be respectively in positions  $P_6$  and  $P$ , where move  $M_D$  is placed on the two leftmost squares marked “1” in segment  $G$  in Figure 19 (a). Thus, the two squares marked “2” in segment  $G'$  in Figure 18 (a) are also added into  $Z_2(P)$  in Figure 19 (a). From the above observation, we can derive  $\Psi(P_6) \subseteq \Psi(P)$ . ■

Since all active segments  $G$  of Defender contains at most 6 (= 4 + 2) unoccupied squares in Connect6, all these squares in  $G$  are added into all  $Z_i(P)$  from operation T3-2, where  $i \geq 4$ . Thus, these  $Z_i(P)$  are nearly the same as  $Z_{un}(P)$ , except for the unoccupied squares not covered by any active segments of Defender, e.g., the unoccupied squares surrounded by all Attacker squares. Similar to the argument in Subsection 3.2.2, we construct zones with size three, and simply use  $Z_{un}(P)$  for those higher-order zones, whenever needed.



### 3.2.3.2. Two Threats

When Attacker has two threats in  $P$ , Defender must defend by blocking the two threats. In this case, the verifier performs the following operations.

- T2-1.** For each Defender move  $M_D$  that blocks the two threats, perform the following.
- a.** Return the value 0 if the recursive  $V_{C6}(P_D, S)$  returns the value 0, where  $P_D = P \oplus M_D$ .
  - b.** Let  $\Psi(P) = \Psi(P) \cup \Psi(P_D)$ .
- T2-2.** Continue to construct zones by both operations T3-1 and T3-2, and return 1.

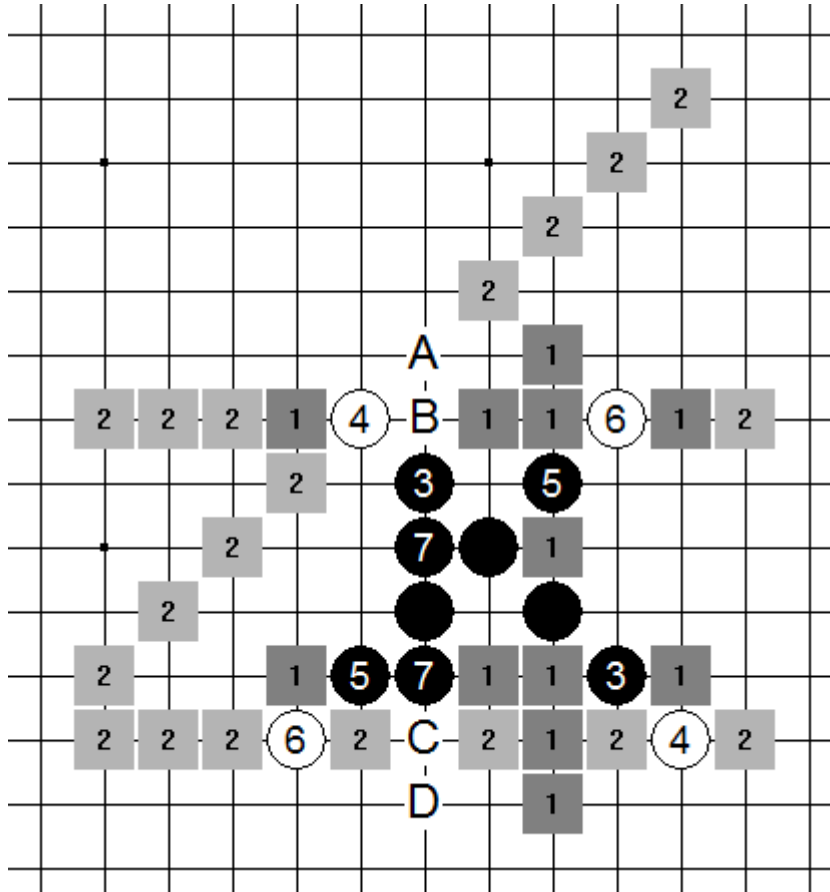


Figure 20. A winning position with two threats for Black (Attacker) and the constructed  $\Psi(P)$ .

For example, for position  $P$  in Figure 20 (the grandparent of the position in Figure 19 (b)) where Black has two threats, White has three defensive moves at (B,C), (A,C) and (B,D). Obviously, since Black still wins for each of the three moves, Black wins in  $P$ . From the above operations, this verifier returns the value 1 and constructs  $\Psi(P)$  as shown in Figure 20. Lemma 8 (below) shows that this verifier satisfies Property RZV if the verifier satisfies Property RZV for all the defensive moves, too. From this lemma,  $\Psi(P)$  in Figure 20 is in  $RZ(P)$ .

**Lemma 8.** From the above, assume that Defender is to move and Attacker has two threats in  $P$ . Assume that all the recursive  $V_{C6}(P_D, S)$  in operation T2-1 satisfy Property RZV. Then, the verifier  $V_{C6}(P, S)$  satisfies Property RZV too.

**Proof.** Assume that this verifier  $V_{C6}(P, S)$  returns 1. For this lemma (this verifier satisfies Property RZV), it suffices to prove that the constructed  $\Psi(P)$  is in  $RZ(P)$ . Since  $V_{C6}(P, S)$  returns 1, all the recursive  $V_{C6}(P_D, S)$  in operation T2-1 must return 1. Since these  $V_{C6}(P_D, S)$  satisfy Property RZV from this lemma, all constructed  $\Psi(P_D)$  are in  $RZ(P_D)$ .

To prove  $\Psi(P) \in RZ(P)$ , it suffices to prove from Lemma 6 the following. For all Defender moves  $M_D$  there exists some  $\Psi_D$  such that  $\Psi_D$  is in  $RZ(P \oplus M_D)$  and  $\Psi_D \subseteq \Psi(P)$ .

All Defender moves  $M_D$  are classified into the following cases.

1. All Defender moves  $M_D$  that block both threats. From the above,  $\Psi(P_D)$  are in  $RZ(P_D)$ . In addition, since these  $\Psi(P_D)$  are merged into  $\Psi(P)$  in operation T2-1.b, we obtain  $\Psi(P_D) \subseteq \Psi(P)$ . Thus,  $\Psi(P_D)$  is the  $\Psi_D$ .
2. All Defender moves  $M_D$  that leave some threat segment unblocked. Attacker wins by connecting six on the segment, like strategy  $S_{3T}$ . Since operation T2-2 follows those steps in T3-1 and T3-2, we simply follow the proof of Lemma 7 to prove that there exists some  $\Psi_D$  such that  $\Psi_D \subseteq \Psi(P)$  and  $\Psi_D$  is in  $RZ(P_D)$ . ■



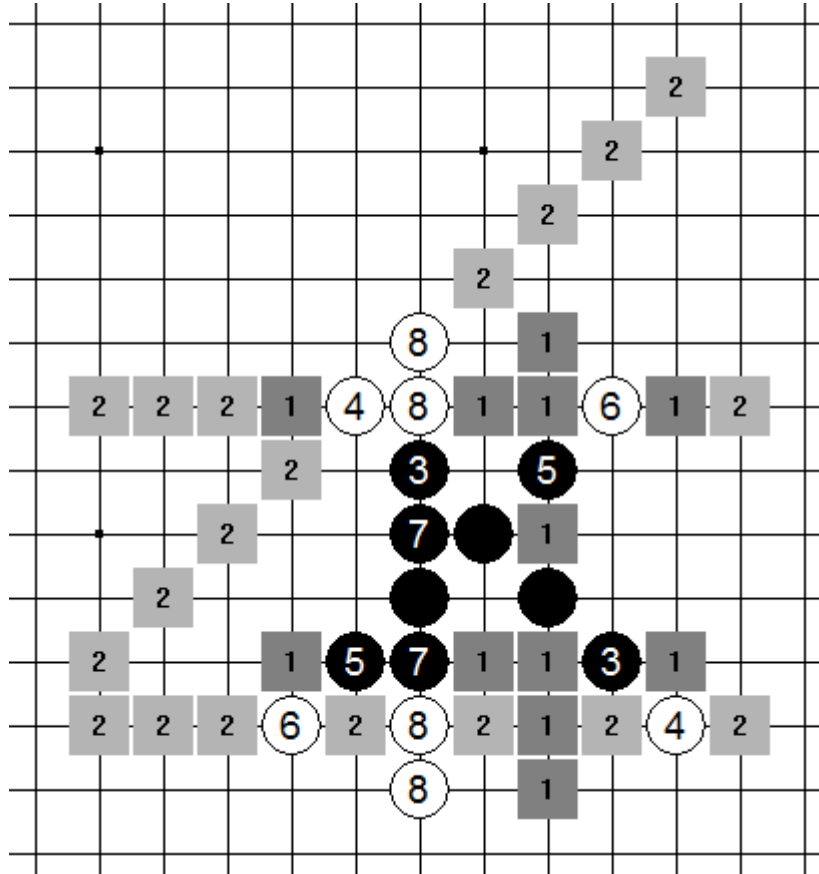


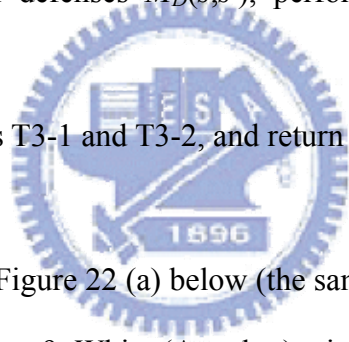
Figure 21. A winning position with two threats for Black (Attacker) and the constructed  $\Psi(P)$ .

Assume that the subsequent winning moves of Attacker are the same for all the defensive moves. Then, we can optimize the construction of zones by combining these defensive moves together. For example, in Figure 20, the three defensive moves, (B,C), (A,C) and (B,D) can be combined into a macro move (A, B, C, D) as shown in Figure 21. Since the subsequent winning sequences of Attacker are the same, the sizes of relevance zones are relatively smaller and the threat-based search is also greatly reduced. However, note that the segment containing both *A* and *B* (the same for *C* and *D*) in Figure 20 should be considered to have one white stone only for zone construction. Since the winning sequences in Figure 12 (a) are the same for all defensive moves, the relevance zones are constructed as shown in Figure 17.

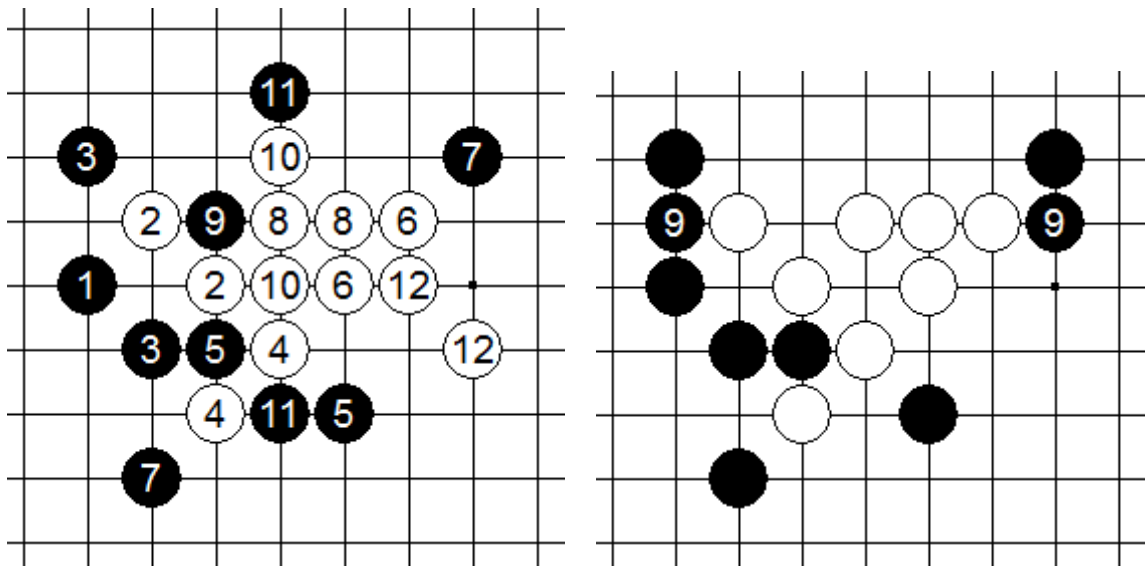
### 3.2.3.3. One Threat

When Attacker has one threat, Defender must defend by blocking the threat. In this case, the verifier performs the following operations.

- T1-1.** For each normal critical defense (defined in Section 1.2),  $M_{D,\phi}(s)$  where square  $s$  blocks the threat, perform the operation of semi-null-move proof search as follows.
  - a.** Return the value 0, if the recursive  $V_{C6}(P_s, S)$  returns 0 where  $P_s = P \oplus M_{D,\phi}(s)$ .
  - b.** Let  $\Psi(P) = \Psi(P) \cup (\Psi(P_s) \ll 1)$ .
  - c.** For each defensive move  $M_D(s, s')$ , where  $s' \in Z_I(P_s)$ , perform both operations T2-1.a and T2-1.b.
- T1-2.** For all relaxed critical defenses  $M_D(s, s')$ , perform both operations T2-1.a and T2-1.b.
- T1-3.** Perform both operations T3-1 and T3-2, and return 1.

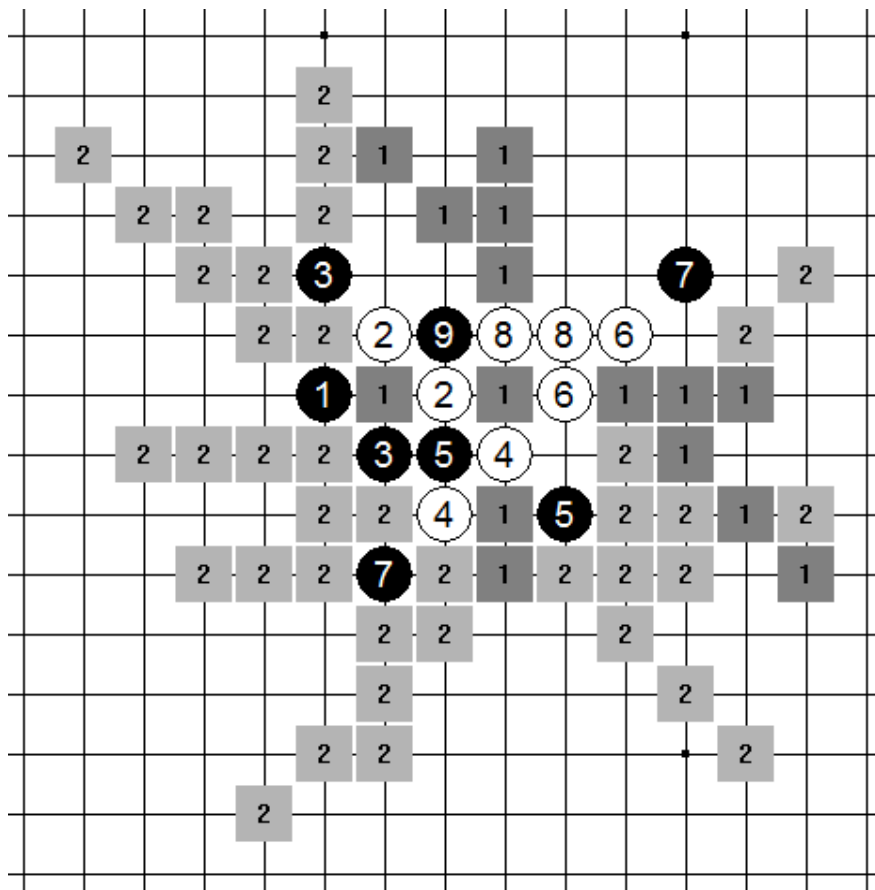


Consider a position  $P$ , 8 in Figure 22 (a) below (the same as 8 in Figure 4), and another  $P_s$ , with a semi-null move added at 9. White (Attacker) wins in  $P_s$  by the winning sequence in Figure 22 (a). The above operations construct the zones  $\Psi(P_s) = \langle Z_I(P_s), Z_2(P_s), Z_3(P_s) \rangle$ , with the first two zones shown in Figure 22 (c). According to operation T1-1.b, both zones  $Z_2(P_s)$  and  $Z_3(P_s)$  are shifted and merged into  $Z_I(P)$  and  $Z_2(P)$ , respectively. For all defensive moves  $M_D(s, s')$ , where  $s' \in Z_I(P_s)$ , operation T1-1.c follows both T2-1.a and T2-1.b to construct zones and verify whether  $V_{C6}(P \oplus M_D(s, s'), S)$  return 1. In addition, operation T1-2 also performs the same for all relaxed critical defenses, such as the one in Figure 22 (b). From Figure 22 (c), since the number of squares in  $Z_I(P_s)$  is only 15, the number of recursive  $V_{C6}$  is relatively small, even in very large or infinite boards.



(a)

(b)



(c)

Figure 22. (a) A VCDT for the semi-null move 9. (b) A relaxed critical defense at 9. (c) The constructed zones for the semi-null move 9 in (a).

Lemma 9 shows that the verifier satisfies Property RZV if all the recursive  $V_{C6}$  satisfy Property RZV.

**Lemma 9.** From the above, assume that Defender is to move and Attacker has one threat in  $P$ . Assume that all the recursive  $V_{C6}$  in both operations T1-1 and T1-2 satisfy Property RZV. Then, the verifier  $V_{C6}(P,S)$  satisfies Property RZV too.

**Proof.** Assume that this verifier  $V_{C6}(P,S)$  returns 1. For this lemma, it suffices to prove that the constructed  $\Psi(P)$  is in  $RZ(P)$ . Since  $V_{C6}(P,S)$  returns 1, all the recursive  $V_{C6}$  in both operations T1-1 and T1-2 must also return 1. Since all the recursive  $V_{C6}$  satisfy Property RZV from this lemma, all  $\Psi(P_s)$  constructed from T1-1.a are in  $RZ(P_s)$  and all  $\Psi(P_D)$  from T1-1.c and T1-2 are in  $RZ(P_D)$ .

To prove  $\Psi(P) \in RZ(P)$ , it suffices to prove from Lemma 6 the following. For all Defender moves  $M_D$ , there exists some  $\Psi_D$  such that  $\Psi_D$  is in  $RZ(P \oplus M_D)$  and  $\Psi_D \subseteq \Psi(P)$ . All Defender moves  $M_D$  are classified into the following cases.

1. All Defender moves  $M_D(s,s')$  where  $s$  blocks the threat as described in T1-1. Let  $P_s = P \oplus M_{D,\phi}(s)$ . Furthermore, this case is separated into the following two subcases.
  - a.  $s' \in Z_I(P_s)$ . Let  $P_D$  denote  $P \oplus M_D(s,s')$ . The zones  $\Psi(P_D)$  is constructed in operation T1-1.c, and is in  $RZ(P_D)$  according to the first paragraph of this proof. Since  $\Psi(P_D)$  is merged into  $\Psi(P)$  in T1-1.c, we obtain  $\Psi(P_D) \subseteq \Psi(P)$ . Thus,  $\Psi(P_D)$  is the  $\Psi_D$ .
  - b.  $s' \in \neg_{P_s}(Z_I(P_s))$ . From the above,  $\Psi(P_s)$  is in  $RZ(P_s)$ . Since  $s' \in \neg_{P_s}(Z_I(P_s))$ , Lemma 3 shows that  $\Psi(P_s) \ll 1$  is in  $RZ(P_s + \sigma_D(s'))$ , meaning  $RZ(P \oplus M_D(s,s'))$ . From operation T1-1.b,  $(\Psi(P_s) \ll 1) \subseteq \Psi(P)$ . Thus,  $\Psi(P_s) \ll 1$  is  $\Psi_D$ .
2. All Defender moves  $M_D(s,s')$  in operation T1-2 are relaxed critical defenses. The proof is similar to that in Case 1.a and therefore omitted.

All Defender moves  $M_D(s,s')$  that do not block the threat. Attacker wins by connecting

six on some unblocked threat segments, like strategy  $S_{3T}$ . Find  $\Psi_D$  by following the proof of Lemma 7. ■

### 3.2.3.4. No Threats

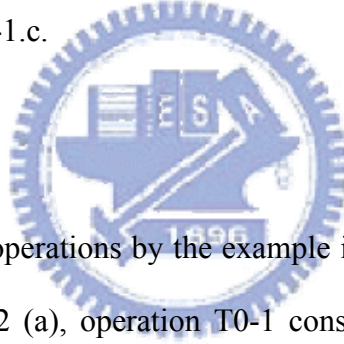
When Attacker has no threats, it becomes more complicated since Defender has much more freedom to move. In this case, the verifier makes use of the constructed relevance zones to minimize the search space in the following operations.

**T0-1.** Return the value 0 if  $V_{C6}(P_\phi, S)$  returns 0, where  $P_\phi = P \oplus M_{D, \phi}$ .

**T0-2.** Let  $\Psi(P) = \Psi(P_\phi) \ll 2$ .

**T0-3.** For each square  $s$  in  $Z_2(P_\phi)$ , perform the semi-null move proof search, as in operations T1-1.a to T1-1.c.

**T0-4.** Return 1.



Let us illustrate the above operations by the example in Figure 6 and Figure 12. From the winning moves in Figure 12 (a), operation T0-1 constructs relevance zones  $\Psi(P_\phi) = \langle Z_1(P_\phi), Z_2(P_\phi), Z_3(P_\phi) \rangle$ , with only the first two zones shown in Figure 17. Similarly, zone  $Z_2(P_\phi)$  is the same as  $Z$  in Figure 12 (a). According to operation T0-2, zone  $Z_3(P_\phi)$  is shifted and merged into  $Z_1(P)$ . Then, in operation T0-3, one square  $s$  in  $Z_2(P_\phi)$  is chosen to perform the semi-null move proof search. In the case that 2 in Figure 12 (b) is chosen, the semi-null move proof search in T0-3 constructs the relevance zones  $\Psi(P_s) = \langle Z_1(P_s), Z_2(P_s), Z_3(P_s) \rangle$ , where  $P_s = P \oplus M_{D, \phi}(s)$ . Zone  $Z_1(P_s)$  is actually the same as  $Z'$  in Figure 12 (b). After verifying that White wins for all  $s \in Z_2(P_\phi)$  and all  $s' \in Z_1(P_s)$ , the verifier confirms that White wins in  $P$ , as shown in Lemma 10 (below). For the position in Figure 6, the number of the recursive  $V_{C6}$  in T0-1 to T0-3 is 2656, relatively small when compared with the number of legal moves.

**Lemma 10.** Assume that Defender is to move and Attacker has no threats in  $P$ . From the above, assume that all recursive  $V_{C6}$  in both operations T0-1 and T0-3 satisfy Property RZV. Then, the verifier  $V_{C6}(P,S)$  also satisfies Property RZV.

**Proof.** Assume that this verifier  $V_{C6}(P,S)$  returns 1. For this lemma, it suffices to prove that the constructed  $\Psi(P)$  is in  $RZ(P)$ . Since  $V_{C6}(P,S)$  returns 1, all the recursive  $V_{C6}$  in both operations T0-1 and T0-3 must also return 1. Since these recursive  $V_{C6}$ , say for position  $P'$ , satisfy Property RZV from this lemma, the constructed zones  $\Psi(P')$  are in  $RZ(P')$ .

To prove  $\Psi(P) \in RZ(P)$ , it suffices to prove from Lemma 6 the following: For all Defender moves  $M_D$ , there exists some  $\Psi_D$  such that  $\Psi_D$  is in  $RZ(P \oplus M_D)$  and  $\Psi_D \subseteq \Psi(P)$ .

All Defender moves  $M_D$  are classified into the following cases:

1. All Defender moves  $M_D(s,s')$  where  $s \in \neg_{P_\phi}(Z_2(P_\phi))$  and  $s' \in \neg_{P_\phi}(Z_2(P_\phi))$ . From the first paragraph in this proof,  $\Psi(P_\phi)$  is in  $RZ(P_\phi)$ . Since  $s \in \neg_{P_\phi}(Z_2(P_\phi))$  and  $s' \in \neg_{P_\phi}(Z_2(P_\phi))$ ,  $\Psi(P_\phi) \ll 2$  is in  $RZ(P_\phi + \sigma_D(s) + \sigma_D(s'))$  from Lemma 3. Since  $P_\phi + \sigma_D(s) + \sigma_D(s') = P \oplus M_D(s,s')$ ,  $\Psi(P_\phi) \ll 2$  is also in  $RZ(P \oplus M_D(s,s'))$ . In addition,  $(\Psi(P_\phi) \ll 2) \subseteq \Psi(P)$  from operation T0-2. Thus,  $\Psi(P_\phi) \ll 2$  is  $\Psi_D$ .
2. All Defender moves  $M_D(s,s')$  where  $s \in Z_2(P_\phi)$ . By following the proof for Case 1 (including Subcases 1.a and 1.b) in Lemma 9, we obtain that there exists some  $\Psi$  in  $P \oplus M_D(s,s')$  for all  $s'$  such that  $\Psi \subseteq \Psi(P)$ . The details are omitted. ■

### 3.3 Conclusion for the Verifier $V_{C6}$

Theorem 2 (below) concludes that the verifier  $V_{C6}(P,S)$  in all cases satisfy Property RZV. Therefore, if  $V_{C6}(P,S)$  returns the value 1, the constructed  $\Psi(P)$  is in  $RZ(P)$ , and Attacker wins in  $P$  from Corollary 2.

**Theorem 2.** The verifier  $V_{C6}(P,S)$  satisfies Property RZV in all cases.

**Proof.** By induction, the verifier  $V_{C6}(P,S)$  satisfies Property RZV in all cases from Lemma 4 to Lemma 10. ■

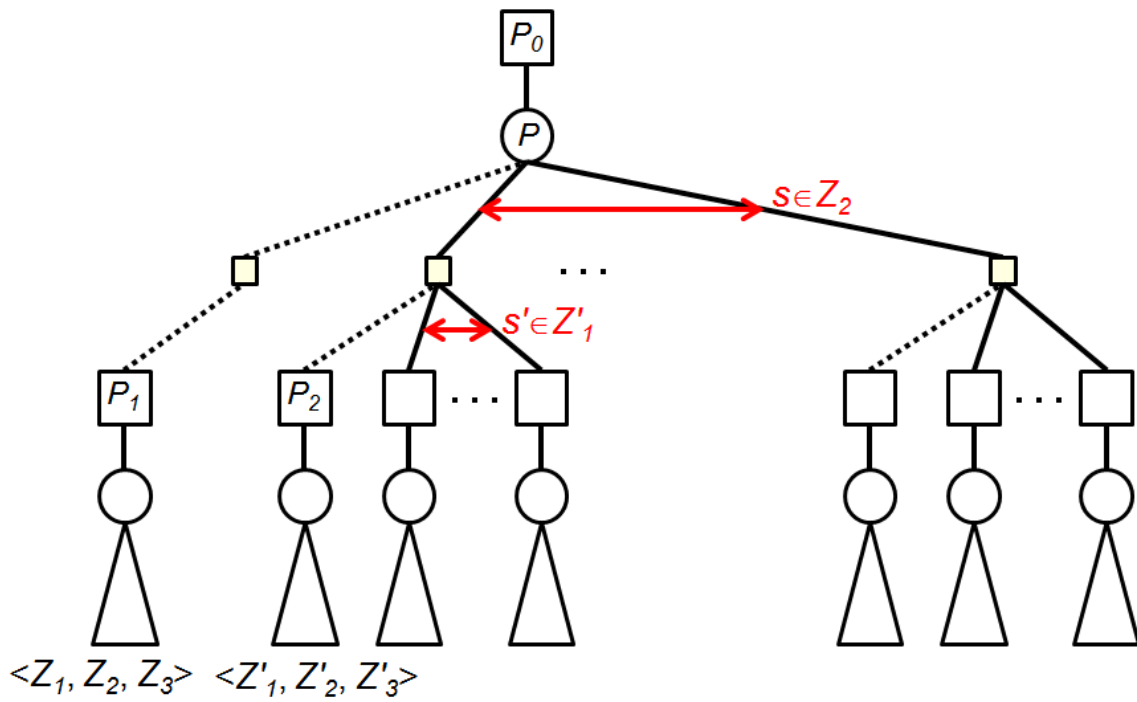


Figure 23. An example proof search tree for the Verifier  $V_{C6}(P,S)$ .

Figure 23 shows an example proof search tree that gives an overview for the Verifier  $V_{C6}(P,S)$ .

# Chapter 4 Segmented Relevance-Zone-Oriented Proof Search for Connect6

As seen in Chapter 3, the RZOP search is a powerful method for proving the winning positions with different orders of threat sequences and constructs relevance zones to reduce greatly the search space. However, we observed some issues when solving Connect6 positions with the RZOP search. First issue is that in the RZOP search we found sequences of squares that are not defined to be irrelevant as shown in Section 4.1. In Section 4.2, we observed the second issue in which counter-threat segments (Defender's threat segments) are the key point that influence whether Attacker can win by simply replaying or not. Section 4.3 presents that there are interesting moves Attacker can win by replaying. Section 4.4 presents two experimental verifiers modified from Sections 4.1 and 4.2 respectively. Section 4.5 proposes an advanced improvement which does not implement yet due to the memory limitation in current *NCTU6* program. Finally, we present our SRZOP verifiers respectively in Subsections 4.1.1, 4.2.1, 4.3.1, 4.4.1 and 4.4.3.

## 4.1 Irrelevant vs. Relevant Sequences of Squares

In Figure 16 of Chapter 3,  $\langle s', s'' \rangle$  is irrelevant to  $\langle Z_1, Z_2, Z_3 \rangle$ , while  $\langle s'', s' \rangle$  is not defined to be irrelevant. However, consider the following situation. Assume  $\Psi = \langle Z_1, Z_2, Z_3 \rangle$  is in  $RZ(P)$ . Then, according to the definition, Attacker wins in  $P + \sigma_D(s') + \sigma_D(s'')$  for irrelevant  $\langle s', s'' \rangle$ . However, it is clear that Attacker wins in  $P + \sigma_D(s'') + \sigma_D(s')$  too. In this sense,  $\langle s'', s' \rangle$  should be defined to be irrelevant too, in this sense. Therefore, we define *relevant* and *not relevant* sequences of squares as follows.



Consider a sequence of zones  $\Psi = \langle Z_1, Z_2, \dots, Z_r \rangle$  in  $P$ . A sequence of unoccupied squares  $\varphi = \langle s_1, s_2, \dots, s_{r'} \rangle$ , where  $r' \leq r$ , is said to be *relevant* to  $\Psi$ , if there exists no permutation  $\varphi'$  from  $\varphi$  such that  $\varphi'$  is irrelevant to  $\Psi$ . Otherwise,  $\varphi$  is said to be *not relevant* to  $\Psi$ . Let  $\varphi \in_P(\Psi)$  denote the relation that  $\varphi$  is *relevant* to  $\Psi$  in  $P$  and  $\varphi \in_{\neg P}(\Psi)$  denote the relation that  $\varphi$  is *not relevant* to  $\Psi$  in  $P$ . Implicitly,  $\neg_P(\Psi)$  denotes  $\langle \neg_P(Z_1), \neg_P(Z_2), \dots, \neg_P(Z_r) \rangle$ . With this new definition, in Figure 16,  $\langle s \rangle$ ,  $\langle s', t' \rangle$ ,  $\langle s', s'', t'' \rangle$ ,  $\langle s', s'', s''', t''' \rangle$ ,  $\langle s, s', s'' \rangle$  are all relevant to  $\langle Z_1, Z_2, Z_3 \rangle$ , while  $\langle s', s'', s''' \rangle$ ,  $\langle s', s'' \rangle$ ,  $\langle s'', s' \rangle$ ,  $\langle s'', s''' \rangle$ ,  $\langle s' \rangle$ ,  $\langle s''' \rangle$  and  $\langle \rangle$  are not.

**Definition 4.** A sequence of zones  $\Psi$  is called a *sequence of relevance zones* for Attacker in a position  $P$ , if and only if Attacker wins in  $P + \sigma_D(\varphi)$  for all not relevant  $\varphi$ ; that is,  $\varphi \in_{\neg P}(\Psi)$ . Let  $RZ(P)$  denote the set of all the sequences of relevance zones for Attacker in  $P$ .

■

From Definition 4, if  $RZ(P)$  is not empty, there must exist some  $\Psi$  in  $RZ(P)$ . This implies that Attacker wins in  $P$  by choosing the empty sequence of squares  $\langle \rangle$  for  $\varphi$ , since  $\varphi$  is not relevant to  $\Psi$  as described above. Thus, Corollary 3 is obtained.

**Corollary 3.** If there exists at least one sequence of zones  $\Psi$  in  $RZ(P)$ , then Attacker wins in  $P$ . ■

The following lemma shows an important property of the SRZOP search.

**Lemma 11.** Assume that  $\Psi = \langle Z_1, Z_2, \dots, Z_r \rangle$  is in  $RZ(P)$ . If  $\varphi$  is not relevant to  $\Psi$ , then Attacker wins in  $P + \sigma_D(\varphi)$ .

**Proof.** Since  $\varphi \in_{\neg P}(\Psi)$ , by definition, there exists some permutation  $\varphi'$  such that  $\varphi'$  is irrelevant to  $\Psi$ . Let  $\varphi = \langle s_1, s_2, \dots, s_{r'} \rangle$  and  $\varphi' = \langle s'_1, s'_2, \dots, s'_{r'} \rangle$ , where  $r' \leq r$ . Since  $\varphi'$  is irrelevant to  $\Psi$ , Attacker wins in  $P + \sigma_D(\varphi') = P + \sigma_D(s'_1) + \sigma_D(s'_2) + \dots + \sigma_D(s'_{r'})$ . Since  $\varphi'$  is a permutation of  $\varphi$ ,  $P + \sigma_D(s'_1) + \sigma_D(s'_2) + \dots + \sigma_D(s'_{r'}) = P + \sigma_D(s_1) + \sigma_D(s_2) + \dots + \sigma_D(s_{r'}) = P + \sigma_D(\varphi)$ . Therefore, Attacker wins in  $P + \sigma_D(\varphi)$  ( $= P + \sigma_D(\varphi')$ ). ■

With Lemma 11, the proofs of Lemma 1, Lemma 2 and Lemma 3 still hold by considering all not relevant  $\varphi$  as shown in Lemma 12, Lemma 13 and Lemma 14, respectively.

**Lemma 12.** Assume that  $\Psi = \langle Z_1, Z_2, \dots, Z_r \rangle$  is in  $RZ(P)$ . Then,  $\Psi' = \langle Z_1, Z_2, \dots, Z_r, Z_{board} \rangle$  is also in  $RZ(P)$ .

**Proof.** Consider all not relevant  $\varphi \in \neg_P(\Psi')$ . For this lemma, it suffices to prove that Attacker wins in  $P + \sigma_D(\varphi)$ . Since  $\neg_P(Z_{board})$  is empty,  $\varphi$  must not have the  $(r + 1)$ -st item. From the definition, we also obtain  $\varphi \in \neg_P(\Psi)$ . Since  $\Psi$  is assumed to be in  $RZ(P)$ , Attacker wins in  $P + \sigma_D(\varphi)$  due to  $\varphi \in \neg_P(\Psi)$ . ■

**Lemma 13.** Assume that  $\Psi$  is in  $RZ(P)$  and  $\Psi \subseteq \Psi'$ . Then,  $\Psi'$  is also in  $RZ(P)$ .

**Proof.** Let  $\Psi = \langle Z_1, Z_2, \dots, Z_r \rangle$  and  $\Psi' = \langle Z'_1, Z'_2, \dots, Z'_r \rangle$ . Consider all not relevant  $\varphi \in \neg_P(\Psi')$ . It suffices to prove that Attacker wins in  $P + \sigma_D(\varphi)$ . Since  $\Psi \subseteq \Psi'$ , the condition  $\varphi \in \neg_P(\Psi')$  also implies  $\varphi \in \neg_P(\Psi)$ . Since  $\Psi$  is in  $RZ(P)$ , Attacker wins in  $P + \sigma_D(\varphi)$  due to  $\varphi \in \neg_P(\Psi)$ . ■

**Lemma 14.** Assume that  $\Psi = \langle Z_1, Z_2, \dots, Z_r \rangle$  is in  $RZ(P)$ . The following two properties are satisfied.

1. Assume that  $\neg_P(Z_1)$  is not empty. Let the unoccupied square be  $s \in \neg_P(Z_1)$ . Then,  $\Psi \ll 1$  is in  $RZ(P + \sigma_D(s))$ .
2. Let  $\varphi$  be a sequence of unoccupied squares  $\langle s_1, s_2, \dots, s_{r'} \rangle$  in  $\neg_P(\Psi)$ , where  $r' \leq r$ . Then,  $\Psi \ll r'$  is in  $RZ(P + \sigma_D(\varphi))$ .

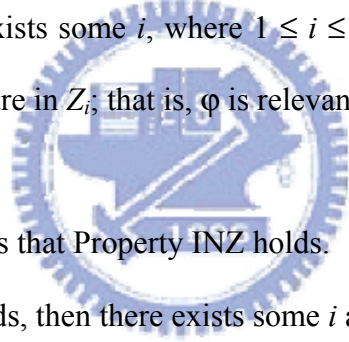
**Proof.** It suffices to prove the first property, since the first implies the second by induction.

Let  $\Psi' = \Psi \ll 1$  and consider all not relevant  $\varphi' = \langle s_2, \dots, s_{r'} \rangle \in \neg_P(\Psi')$ , where  $r' \leq r$ . For the

first property, it suffices to prove that Attacker wins in  $(P + \sigma_D(s)) + \sigma_D(\varphi')$ . Let  $\varphi = \langle s_1, s_2, \dots, s_{r'} \rangle$ . Then, the condition  $\varphi \in \neg_P(\Psi)$  holds due to  $s_1 \in \neg_P(Z_1)$ . Since  $\Psi$  is in  $RZ(P)$ , Attacker wins in  $P + \sigma_D(\varphi)$  due to  $\varphi \in \neg_P(\Psi)$ ; that is, Attacker wins in  $(P + \sigma_D(s)) + \sigma_D(\varphi)$  ( $= P + \sigma_D(\varphi)$ ). ■

Consider  $\Psi = \langle Z_1, Z_2, \dots, Z_r \rangle$  and a sequence of unoccupied squares  $\varphi = \langle s_1, s_2, \dots, s_{r'} \rangle$ , where  $r' \leq r$ . A sequence of unoccupied squares  $\varphi' = \langle s_{b_1}, s_{b_2}, \dots, s_{b_i} \rangle$  is said to a *subsequence* of  $\varphi$ , if  $1 \leq b_1 < b_2 < \dots < b_i \leq r'$ . The definition of relevant  $\varphi$  is equivalent to the following property which means placing  $i$  stones inside  $Z_i$ .

**Property INZ:** Let  $\Psi = \langle Z_1, Z_2, \dots, Z_r \rangle$  and  $\varphi = \langle s_1, s_2, \dots, s_{r'} \rangle$ , where  $r' \leq r$ . Assume that  $\varphi$  is relevant to  $\Psi$ , then there exists some  $i$ , where  $1 \leq i \leq r'$ , and some subsequence  $\varphi'$  of size  $i$  such that all squares in  $\varphi'$  are in  $Z_i$ ; that is,  $\varphi$  is relevant to  $\Psi$ .



The following lemma shows that Property INZ holds.

**Lemma 15.** If Property INZ holds, then there exists some  $i$  and some subsequence  $\varphi'$  of size  $i$  such that all squares in  $\varphi'$  are in  $Z_i$ ; that is,  $\varphi$  is relevant to  $\Psi$ . Otherwise,  $\varphi$  is not relevant to  $\Psi$ .

**Proof.** Assume by contradictory that Property INZ does not hold and  $\varphi$  is relevant to  $\Psi$ . This implies that for all  $1 \leq i \leq r'$ , there does not exist some subsequence with size  $i$ . Let us investigate each  $i$  as follows.

(a) There does not exist some subsequence  $\varphi'$  of size  $r'$  such that all squares in  $\varphi'$  are in  $Z_{r'}$ . Therefore, there exists at least one square, said  $s'_{r'}$ , not in  $Z_{r'}$ , that is  $s'_{r'} \in \neg_P(Z_{r'})$ . Let  $s_{r'} = s'_{r'}$ .

(b) Similarly, there does not exist some subsequence  $\varphi'$  of size  $r' - 1$  such that all squares in  $\varphi'$  are in  $Z_{r'-1}$ . Therefore, there exist at least two squares not in  $Z_{r'}$ . In addition to  $s_{r'}$ , let another square  $s_{r'-1} \in \neg_P(Z_{r'-1})$ .

(c) Therefore, there exist at least  $i$  squares, where  $1 \leq i \leq r'$ , not in  $Z_i$ . In addition to  $s_{r'}$ ,  $s_{r'-1}, \dots, s_{i+1}$ , let another square  $s_i \in \neg_P(Z_i)$ .

From above, let  $\varphi = \langle s_1, s_2, \dots, s_{r'} \rangle$ . Since  $s_1 \in \neg_P(Z_1)$ ,  $s_2 \in \neg_P(Z_2)$ ,  $s_3 \in \neg_P(Z_3)$ ,  $\dots$ ,  $s_{r'} \in \neg_P(Z_{r'})$ ,  $\varphi$  is not relevant to  $\Psi$ . It contradicts the assumption. ■

In additional, we obtain the following lemma.

**Lemma 16.** Assume that  $\Psi$  is in  $RZ(P)$ , where  $\Psi = \langle Z_1, Z_2, \dots, Z_{r'} \rangle$  and  $\varphi = \langle s_1, s_2, \dots, s_{r'} \rangle$ , where  $r' \leq r$ . If Property INZ does not hold, then Attacker wins in  $P + \sigma_D(\varphi)$ .

**Proof.** From Lemma 11 and Lemma 15, this lemma is trivial and therefore omitted. ■

#### 4.1.1 The Proposed Verifier $V_{C6-01}$

This subsection presents a verifier, named  $V_{C6-01}(P, S)$ , with a new proof search method for Connect6. The verifier  $V_{C6-01}(P, S)$  is described in Subsections 4.1.1.1, 4.1.1.2 and 4.1.1.3 respectively for three distinct kinds of  $P$ , namely endgame positions, positions in Attacker's turn and positions in Defender's turn. Finally, Subsection 4.1.2 concludes with Theorem 3, showing that the verifier satisfies Property RZV in all cases.

##### 4.1.1.1. Endgame Positions

If Attacker does not win in the endgame position  $P$ , the verifier simply returns the value 0. If Attacker wins in  $P$  (i.e., Attacker has a win segment in  $P$ ), the verifier returns 1 and constructs  $\Psi(P)$  in the following operation.

**O1-EP-1.** For each active segment  $G$  of Defender containing *exactly*  $i$  unoccupied squares, these squares in  $G$  are all added into  $Z_i(P)$  or higher-order zones; that is,  $Z_j(P)$  for all  $j \geq i$ . In other words, for each active segment  $G$  of Defender containing at most  $i$  unoccupied squares, add all of these squares in  $G$  into  $Z_i(P)$ .

**Lemma 17.** Assume  $P$  to be an endgame position. Property RZV is satisfied for  $V_{C6-OI}(P,S)$ .

**Proof.** Omitted. ■

#### 4.1.1.2. Positions in Attacker's Turn

In such positions, Attacker simply follows strategy  $S$  to make the move  $S(P)$  in  $P$ . Let  $P_A$  denote  $P \oplus S(P)$ . This verifier first performs  $V_{C6-OI}(P_A,S)$  recursively. If  $V_{C6-OI}(P_A,S)$  returns the value 0, this verifier  $V_{C6-OI}(P,S)$  also returns 0. On the other hand, if  $V_{C6-OI}(P_A,S)$  returns 1, this verifier  $V_{C6-OI}(P,S)$  returns 1, too; and constructs  $\Psi(P)$  in the following operation.

**O1-AT-1.** Let  $\Psi(P) = \Psi(P_A) \cup Z_S$ , where  $Z_S = \{s \mid s \in S(P)\}$ .

**Lemma 18.** Assume a position  $P$  in Attacker's turn. From the above, assume that  $V_{C6-OI}(P_A,S)$  satisfies Property RZV, where  $P_A = P \oplus S(P)$ . This verifier  $V_{C6-OI}(P,S)$  satisfies Property RZV.

**Proof.** Assume that this verifier  $V_{C6-OI}(P,S)$  returns the value 1. For this lemma (this verifier satisfies Property RZV), it suffices to prove that the constructed  $\Psi(P)$  is in  $RZ(P)$ . From the above operation,  $V_{C6-OI}(P_A,S)$  must also return 1. Since  $V_{C6-OI}(P_A,S)$  satisfies Property RZV from the lemma,  $\Psi(P_A)$  is in  $RZ(P_A)$ .

Consider all not relevant  $\varphi$ , where  $\varphi \in \neg_P \Psi(P)$ . It suffices to prove that Attacker wins in  $P + \sigma_D(\varphi)$ . Since the property  $\neg_P \Psi(P) = \neg_{P_A} \Psi(P_A)$  is satisfied as described above, the condition  $\varphi \in \neg_{P_A} \Psi(P_A)$  holds too. Since  $\Psi(P_A)$  is in  $RZ(P_A)$  from above, Attacker wins in  $P_A + \sigma_D(\varphi)$  due to  $\varphi \in \neg_{P_A} \Psi(P_A)$ . Since Attacker wins in  $P_A + \sigma_D(\varphi) = (P + \sigma_D(\varphi)) \oplus S(P)$ , Attacker wins in  $P + \sigma_D(\varphi)$  by choosing the move  $S(P)$ . ■

### 4.1.1.3. Positions in Defender's Turn

For positions in Defender's turn, the following lemma shows a very important property used in this subsection.

**Lemma 19.** Assume a position  $P$  in Defender's turn. For a given sequence of zones  $\Psi$ , assume that for all Defender moves  $M_D$  there exists some  $\Psi_D$  such that  $\Psi_D \subseteq \Psi$  and  $\Psi_D$  is in  $RZ(P \oplus M_D)$ . Then  $\Psi$  is in  $RZ(P)$ .

**Proof.** Consider all not relevant  $\varphi \in \neg_P \Psi$ . For this lemma, it suffices to prove that Attacker wins in  $P + \sigma_D(\varphi)$ .

Now, consider all Defender moves  $M_D$  in  $P + \sigma_D(\varphi)$ . From this lemma, there exists some  $\Psi_D$  such that  $\Psi_D \subseteq \Psi$  and  $\Psi_D$  is in  $RZ(P \oplus M_D)$ . Since  $\Psi_D \subseteq \Psi$ , the condition  $\varphi \in \neg_P \Psi$  implies  $\varphi \in \neg_P \Psi_D$ . Since squares in  $M_D$  and  $\sigma_D(\varphi)$  are mutually exclusive,  $\varphi \in \neg_P \Psi_D$  also implies  $\varphi \in \neg_{P \oplus M_D} \Psi_D$ . Since  $\Psi_D$  is in  $RZ(P \oplus M_D)$  from above, Attacker wins in  $(P \oplus M_D) + \sigma_D(\varphi)$  due to  $\varphi \in \neg_{P \oplus M_D} \Psi_D$ . Since  $(P \oplus M_D) + \sigma_D(\varphi) = (P + \sigma_D(\varphi)) \oplus M_D$ , Attacker also wins in  $(P + \sigma_D(\varphi)) \oplus M_D$ . From the above, since Attacker wins in  $(P + \sigma_D(\varphi)) \oplus M_D$  over all Defender moves  $M_D$ , Attacker wins in  $P + \sigma_D(\varphi)$ . ■

In Connect6, the position  $P$  (in Defender's turn) can be classified into the following four cases. The number of Attacker threats in  $P$  is (1) three or more, (2) two, (3) one and (4) zero. The four cases are discussed respectively in the following four subsections.

#### 4.1.1.3.1. Three Threats or More

In this case, Attacker is sure to win by simply following the strategy,  $S_{3T}$ , as follows. For each Defender move, since the move must leave some threat segments unblocked, Attacker wins simply by making a win segment from the unblocked one. Since the strategy

is a sure win, the verifier returns the value 1 and constructs the zones (initialized to be empty) in the following operations.

**O1-T3-1.** Add all unoccupied squares  $s$  on threat segments into all  $Z_i(P)$ .

**O1-T3-2.** For each active segment  $G$  of Defender containing *exactly*  $i + 2$  unoccupied squares, all these squares in  $G$  are added into all  $Z_i(P)$  or higher-order zones. In other words, for each active segment  $G$  of Defender containing at most  $i + 2$  unoccupied squares, add all these squares in  $G$  into  $Z_i(P)$ .

**Lemma 20.** Assume that Defender is to move and Attacker has three or more threats in  $P$ . The verifier described above satisfies Property RZV.

**Proof.** For this lemma, it suffices to prove that the constructed  $\Psi(P)$  is in  $RZ(P)$ . Consider all Defender moves  $M_D$ . Attacker simply follows a strategy  $S_{3T}$  to connect six from an unblocked threat segment. Let  $P_D = P \oplus M_D$  and  $P_6 = P_D \oplus S_{3T}(P_D)$ . From Lemma 17 and Lemma 18,  $\Psi(P_6)$  and  $\Psi(P_D)$  are in  $RZ(P_6)$  and  $RZ(P_D)$ , respectively.

To prove that  $\Psi(P)$  is in  $RZ(P)$ , it suffices to prove from Lemma 19 that  $\Psi(P_D) \subseteq \Psi(P)$ , since  $\Psi(P_D)$  is already in  $RZ(P_D)$ . From Subsection 4.1.1.2,  $\Psi(P_D) = \Psi(P_6) \cup Z_S$ , where  $Z_S = \{s \mid s \in S_{3T}(P_D)\}$ . From operation O1-T3-1, all squares in  $Z_S$  are added into  $\Psi(P)$ . Thus, it suffices to prove that  $\Psi(P_6) \subseteq \Psi(P)$ .

Since Attacker connects six in  $P_6$ , operation O1-EP-1 (in Subsection 4.1.1.1) is employed to construct zones  $\Psi(P_6)$ . The operation is restated as follows. For each active segment  $G$  of Defender containing at most  $i$  unoccupied squares in  $P_6$ , all the squares in  $G$  are added into  $Z_i(P_6)$ . Since one move has at most two squares, at most two occupied squares in  $G$  were occupied by move  $M_D$ . Therefore,  $G$  contains at most  $2 + i$  unoccupied

squares back in  $P$  (before making move  $M_D$ ). From operation O1-T3-2, all these unoccupied squares are also added into  $Z_i(P)$ . For example, let both lines in Figure 18 (a) and Figure 19 (a) (in Section 3.2) be respectively in positions  $P_6$  and  $P$ , where move  $M_D$  is placed on the two leftmost squares marked “1” in segment  $G$  in Figure 19 (a). Thus, the two squares marked “2” in segment  $G'$  in Figure 18 (a) are also added into  $Z_2(P)$  in Figure 19 (a). From the above observation, we can derive  $\Psi(P_6) \subseteq \Psi(P)$ . ■

#### 4.1.1.3.2. Two Threats

When Attacker has two threats in  $P$ , Defender must defend by blocking the two threats. In this case, the verifier performs the following operations.

- O1-T2-1.** For each Defender move  $M_D$  that blocks the two threats, perform the following.
- a. Return the value 0 if the recursive  $V_{C6-O1}(P_D, S)$  returns the value 0, where  $P_D = P \oplus M_D$ .
  - b. Let  $\Psi(P) = \Psi(P) \cup \Psi(P_D)$ .
- O1-T2-2.** Continue to construct zones by both operations O1-T3-1 and O1-T3-2, and return 1.

**Lemma 21.** From the above, assume that Defender is to move and Attacker has two threats in  $P$ . Assume that all the recursive  $V_{C6-O1}(P_D, S)$  in operation O1-T2-1 satisfy Property RZV. Then, the verifier  $V_{C6-O1}(P, S)$  satisfies Property RZV too.

**Proof.** Assume that this verifier  $V_{C6-O1}(P, S)$  returns 1. For this lemma (this verifier satisfies Property RZV), it suffices to prove that the constructed  $\Psi(P)$  is in  $RZ(P)$ . Since  $V_{C6-O1}(P, S)$  returns 1, all the recursive  $V_{C6-O1}(P_D, S)$  in operation O1-T2-1 must return 1. Since these  $V_{C6-O1}(P_D, S)$  satisfy Property RZV from this lemma, all constructed  $\Psi(P_D)$  are in  $RZ(P_D)$ .

To prove  $\Psi(P) \in RZ(P)$ , it suffices to prove from Lemma 19 the following. For all



Defender moves  $M_D$  there exists some  $\Psi_D$  such that  $\Psi_D$  is in  $RZ(P \oplus M_D)$  and  $\Psi_D \subseteq \Psi(P)$ .

All Defender moves  $M_D$  are classified into the following cases.

1. All Defender moves  $M_D$  that block both threats. From the above,  $\Psi(P_D)$  are in  $RZ(P_D)$ . In addition, since these  $\Psi(P_D)$  are merged into  $\Psi(P)$  in operation O1-T2-1.b, we obtain  $\Psi(P_D) \subseteq \Psi(P)$ . Thus,  $\Psi(P_D)$  is the  $\Psi_D$ .
2. All Defender moves  $M_D$  that leave some threat segment unblocked. Attacker wins by connecting six on the segment, like strategy  $S_{3T}$ . Since operation O1-T2-2 follows those steps in O1-T3-1 and O1-T3-2, we simply follow the proof of Lemma 20 to prove that there exists some  $\Psi_D$  such that  $\Psi_D \subseteq \Psi(P)$  and  $\Psi_D$  is in  $RZ(P_D)$ . ■

#### 4.1.1.3.3. One Threat

When Attacker has one threat, Defender must defend by blocking the threat. In this case, the verifier performs the following operations.

- O1-T1-1.** For each normal critical defense (defined in Section 1.2),  $M_{D,\phi}(s)$  where square  $s$  blocks the threat, perform the operation of semi-null-move proof search as follows.
  - a.** Return the value 0, if the recursive  $V_{C6-O1}(P_s, S)$  returns 0 where  $P_s = P \oplus M_{D,\phi}(s)$ .
  - b.** Let  $\Psi(P) = \Psi(P) \cup (\Psi(P_s) \ll 1)$ .
  - c.** For each defensive move  $M_D(s, s')$ , where  $s' \in Z_I(P_s)$ , perform both operations O1-T2-1.a and O1-T2-1.b.
- O1-T1-2.** For all relaxed critical defenses  $M_D(s, s')$ , perform both operations O1-T2-1.a and O1-T2-1.b.
- O1-T1-3.** Perform both operations O1-T3-1 and O1-T3-2, and return 1.

**Lemma 22.** From the above, assume that Defender is to move and Attacker has one threat in  $P$ . Assume that all the recursive  $V_{C6-O1}$  in both operations O1-T1-1 and O1-T1-2 satisfy Property RZV. Then, the verifier  $V_{C6-O1}(P,S)$  satisfies Property RZV too.

**Proof.** Assume that this verifier  $V_{C6-O1}(P,S)$  returns 1. For this lemma, it suffices to prove that the constructed  $\Psi(P)$  is in  $RZ(P)$ . Since  $V_{C6-O1}(P,S)$  returns 1, all the recursive  $V_{C6-O1}$  in both operations O1-T1-1 and O1-T1-2 must also return 1. Since all the recursive  $V_{C6-O1}$  satisfy Property RZV from this lemma, all  $\Psi(P_s)$  constructed from O1-T1-1.a are in  $RZ(P_s)$  and all  $\Psi(P_D)$  from O1-T1-1.c and O1-T1-2 are in  $RZ(P_D)$ .

To prove  $\Psi(P) \in RZ(P)$ , it suffices to prove from Lemma 19 the following. For all Defender moves  $M_D$ , there exists some  $\Psi_D$  such that  $\Psi_D$  is in  $RZ(P \oplus M_D)$  and  $\Psi_D \subseteq \Psi(P)$ . All Defender moves  $M_D$  are classified into the following cases.

1. All Defender moves  $M_D(s,s')$  where  $s$  blocks the threat as described in O1-T1-1. Let  $P_s = P \oplus M_{D,\phi}(s)$ . Furthermore, this case is separated into the following two subcases.
  - a.  $s' \in Z_I(P_s)$ . Let  $P_D$  denote  $P \oplus M_D(s,s')$ . The zones  $\Psi(P_D)$  is constructed in operation O1-T1-1.c, and is in  $RZ(P_D)$  according to the first paragraph of this proof. Since  $\Psi(P_D)$  is merged into  $\Psi(P)$  in O1-T1-1.c, we obtain  $\Psi(P_D) \subseteq \Psi(P)$ . Thus,  $\Psi(P_D)$  is the  $\Psi_D$ .
  - b.  $s' \in \neg_{P_s}(Z_I(P_s))$ . From the above,  $\Psi(P_s)$  is in  $RZ(P_s)$ . Since  $s' \in \neg_{P_s}(Z_I(P_s))$ , Lemma 14 shows that  $\Psi(P_s) \ll 1$  is in  $RZ(P_s + \sigma_D(s'))$ , meaning  $RZ(P \oplus M_D(s,s'))$ . From operation O1-T1-1.b,  $(\Psi(P_s) \ll 1) \subseteq \Psi(P)$ . Thus,  $\Psi(P_s) \ll 1$  is  $\Psi_D$ .
2. All Defender moves  $M_D(s,s')$  in operation O1-T1-2 are relaxed critical defenses. The proof is similar to that in Case 1.a and therefore omitted.

All Defender moves  $M_D(s,s')$  that do not block the threat. Attacker wins by connecting six on some unblocked threat segments, like strategy  $S_{3T}$ . Find  $\Psi_D$  by following the proof of Lemma 20. ■

#### 4.1.1.3.4. No Threats

When Attacker has no threats, it becomes more complicated since Defender has much more freedom to move. In this case, the verifier makes use of the constructed relevance zones to minimize the search space in the following operations.

- O1-T0-1.** Return the value 0 if  $V_{C6-O1}(P_\phi, S)$  returns 0, where  $P_\phi = P \oplus M_{D, \phi\phi}$ .
- O1-T0-2.** Let  $\Psi(P) = \Psi(P_\phi) \ll 2$ .
- O1-T0-3.** For each square  $s$  in  $Z_1(P_\phi)$ , perform the semi-null move proof search, as in operations O1-T1-1.a to O1-T1-1.c.
- O1-T0-4.** For each square  $s$  in  $\dot{Z}_2(P_\phi)$ , where  $\dot{Z}_2(P_\phi) = Z_2(P_\phi) \setminus Z_1(P_\phi)$ , perform the operation which satisfies Property INZ as follows.
  - a.** For each defensive move  $M_D(s, s')$ , where  $s' \in \dot{Z}_2(P_\phi)$  and  $s' \neq s$ , perform both operations O1-T2-1.a and O1-T2-1.b.
- O1-T0-5.** Return 1.

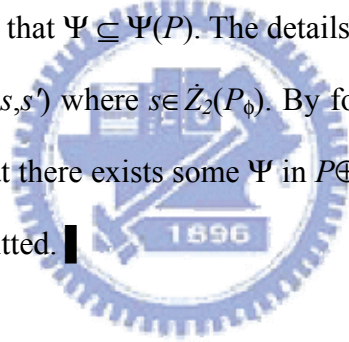
**Lemma 23.** Assume that Defender is to move and Attacker has no threats in  $P$ . From the above, assume that all recursive  $V_{C6-O1}$  in operations O1-T0-1, O1-T0-3 and O1-T0-4 satisfy Property RZV. Then, the verifier  $V_{C6-O1}(P, S)$  also satisfies Property RZV.

**Proof.** Assume that this verifier  $V_{C6-O1}(P, S)$  returns 1. For this lemma, it suffices to prove that the constructed  $\Psi(P)$  is in  $RZ(P)$ . Since  $V_{C6-O1}(P, S)$  returns 1, all the recursive  $V_{C6-O1}$  in operations O1-T0-1, O1-T0-3 and O1-T0-4 must also return 1. Since these recursive  $V_{C6-O1}$ , say for position  $P'$ , satisfy Property RZV from this lemma, the constructed zones  $\Psi(P')$  are in  $RZ(P')$ .

To prove  $\Psi(P) \in RZ(P)$ , it suffices to prove from Lemma 19 the following: For all Defender moves  $M_D$ , there exists some  $\Psi_D$  such that  $\Psi_D$  is in  $RZ(P \oplus M_D)$  and  $\Psi_D \subseteq \Psi(P)$ .

All Defender moves  $M_D$  are classified into the following cases:

1. All Defender moves  $M_D(s, s')$  where  $s \in \neg_{P_\phi}(Z_2(P_\phi))$  and  $s' \in \neg_{P_\phi}(Z_2(P_\phi))$ . From the first paragraph in this proof,  $\Psi(P_\phi)$  is in  $RZ(P_\phi)$ . Since  $s \in \neg_{P_\phi}(Z_2(P_\phi))$  and  $s' \in \neg_{P_\phi}(Z_2(P_\phi))$ ,  $\Psi(P_\phi) \ll 2$  is in  $RZ(P_\phi + \sigma_D(s) + \sigma_D(s'))$  from Lemma 14. Since  $P_\phi + \sigma_D(s) + \sigma_D(s') = P \oplus M_D(s, s')$ ,  $\Psi(P_\phi) \ll 2$  is also in  $RZ(P \oplus M_D(s, s'))$ . In addition,  $(\Psi(P_\phi) \ll 2) \subseteq \Psi(P)$  from operation O1-T0-2. Thus,  $\Psi(P_\phi) \ll 2$  is  $\Psi_D$ .
2. All Defender moves  $M_D(s, s')$  where  $s \in Z_I(P_\phi)$ . By following the proof for Case 1 (including Subcases 1.a and 1.b) in Lemma 22, we obtain that there exists some  $\Psi$  in  $P \oplus M_D(s, s')$  for all  $s'$  such that  $\Psi \subseteq \Psi(P)$ . The details are omitted. ■
3. All Defender moves  $M_D(s, s')$  where  $s \in \dot{Z}_2(P_\phi)$ . By following the proof for Case 2 in Lemma 22, we obtain that there exists some  $\Psi$  in  $P \oplus M_D(s, s')$  for all  $s'$  such that  $\Psi \subseteq \Psi(P)$ . The details are omitted. ■



### 4.1.2 Conclusion for the Verifier $V_{C6-01}$

Theorem 3 (below) concludes that the verifier  $V_{C6-01}(P,S)$  in all cases satisfy Property RZV. Therefore, if  $V_{C6-01}(P,S)$  returns the value 1, the constructed  $\Psi(P)$  is in  $RZ(P)$ , and Attacker wins in  $P$  from Corollary 3.

**Theorem 3.** The verifier  $V_{C6-01}(P,S)$  satisfies Property RZV in all cases.

**Proof.** By induction, the verifier  $V_{C6-01}(P,S)$  satisfies Property RZV in all cases from Lemma 17 to Lemma 23. ■

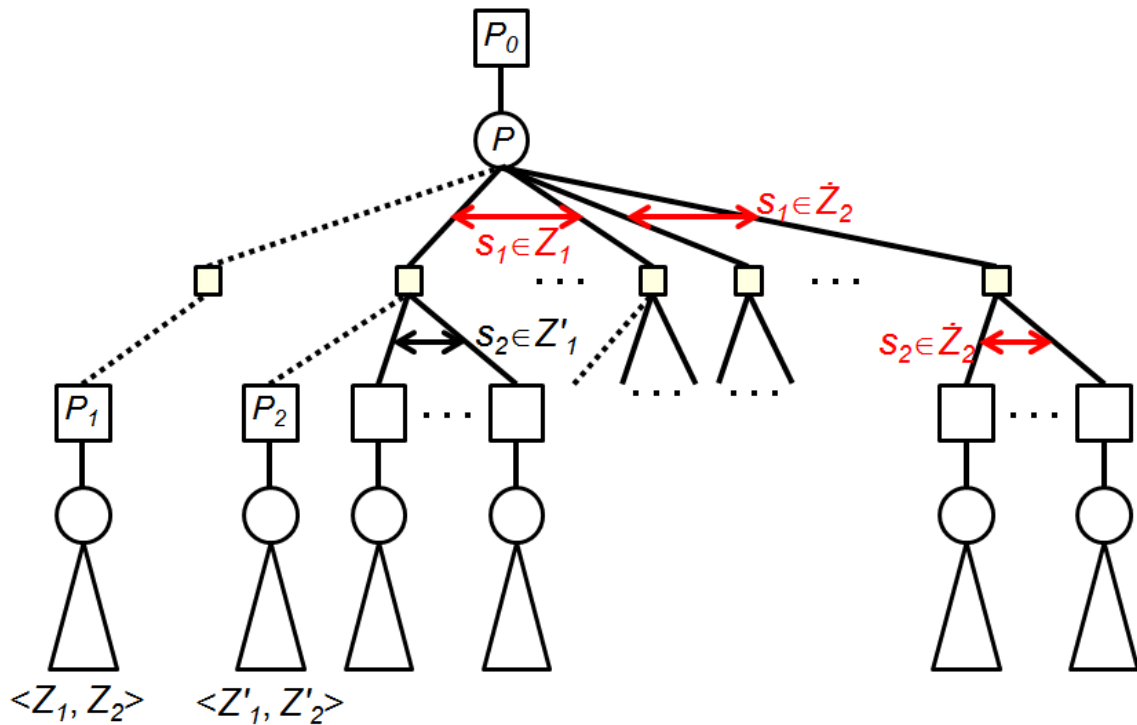


Figure 24. An example proof search tree, where  $\check{Z}_2 = Z_2 \setminus Z_1$ , for the Verifier  $V_{C6-01}(P,S)$ .

Figure 24 shows an example proof search tree that gives an overview for the Verifier  $V_{C6-01}(P,S)$ .

## 4.2 Counter-threat Sequences of Squares

In Section 4.1, we propose Property INZ which means  $\varphi$  is relevant to  $\Psi(P)$  if Defender places  $i$  stones inside  $Z_i(P)$ . Therefore, we consider only relevant  $\varphi$  for the verifier  $V_{C6-OI}(P,S)$ . However, there exist some  $\varphi \in_P(\Psi)$  such that Attacker wins by replaying when we perform the verifier  $V_{C6-OI}(P,S)$ .

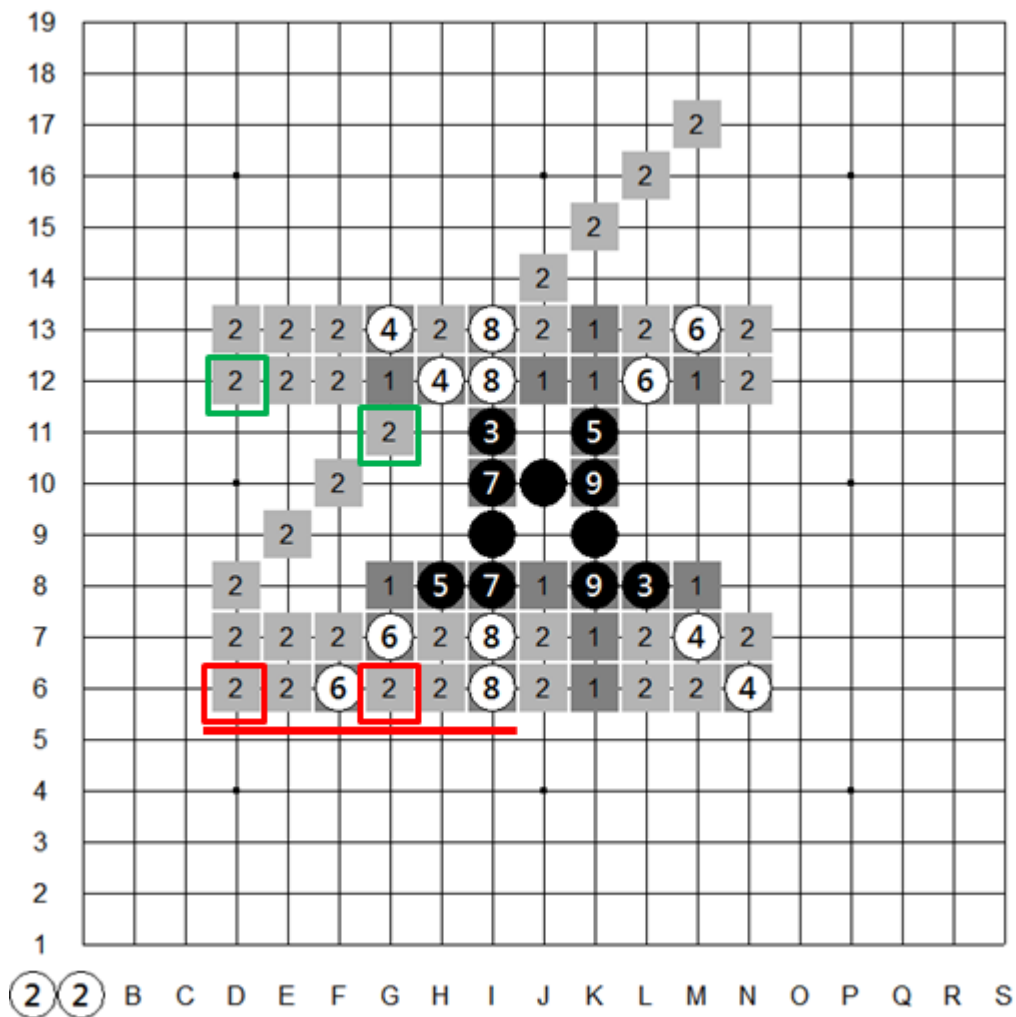


Figure 25. Two types of moves  $M'_D(D12, G11)$  and  $M''_D(D6, G6)$ .

For example, let  $P$  denote the position in Figure 6 and  $P' = P \oplus M_{D, \phi}$ . Figure 25 shows a VCDT for the position  $P'$ . Given two moves  $M'_D(D12, G11)$  and  $M''_D(D6, G6)$  as shown in Figure 25. It is obviously that Attacker wins in  $P \oplus M'_D$  by replaying the same VCDT in Figure 25. However, Attacker cannot win by simply replaying in  $P \oplus M''_D$ , since Defender makes a single-threat move 8 before Attacker makes the triple-threat move 9.

Let  $\Psi = \langle Z_1, Z_2 \rangle$  the gray area in  $P$  as shown in Figure 25. From the above observation and Property INZ in Section 4.1, the only chance for Defender to prevent Attacker wins by replaying is to place at least one stone in  $Z_1$  or make a threat move before Attacker's triple-threat move. Since placing one stone in  $Z_1$  always prevents Attacker from winning by replaying, we focus on placing two stone in  $\dot{Z}_2 = Z_2 \setminus Z_1$  by Property INZ.

From examples in Figure 25, we classify Defender moves into two types. First type is those Defender moves that may form threat segments, also called *counter-threat* segments, as  $M''_D(D6, G6)$  shown in Figure 25. Second type is those Defender moves that are sure not form threat segments as  $M'_D(D12, G11)$  shown in Figure 25. Therefore, in this section, we investigate first type of Defender moves.

Let  $\Psi = \langle Z_1, Z_2 \rangle$  be a sequence of relevance zones in  $P$  and a sequence of unoccupied squares  $\phi = \langle s, s' \rangle$ . For each square  $s$  in  $\dot{Z}_2$ , where  $\dot{Z}_2 = Z_2 \setminus Z_1$ , to form counter-threat segments,  $s'$  and  $s$  must be in a same active segment and  $s'$  must be in the eight directions from  $s$ . For example, Figure 26 (below) shows the possible area for squares  $s'$  after Defender places the first square  $s$ . Since the possible area looks like the Chinese word 米, we denote the area  $\dot{Z}_2 \text{米}$ .

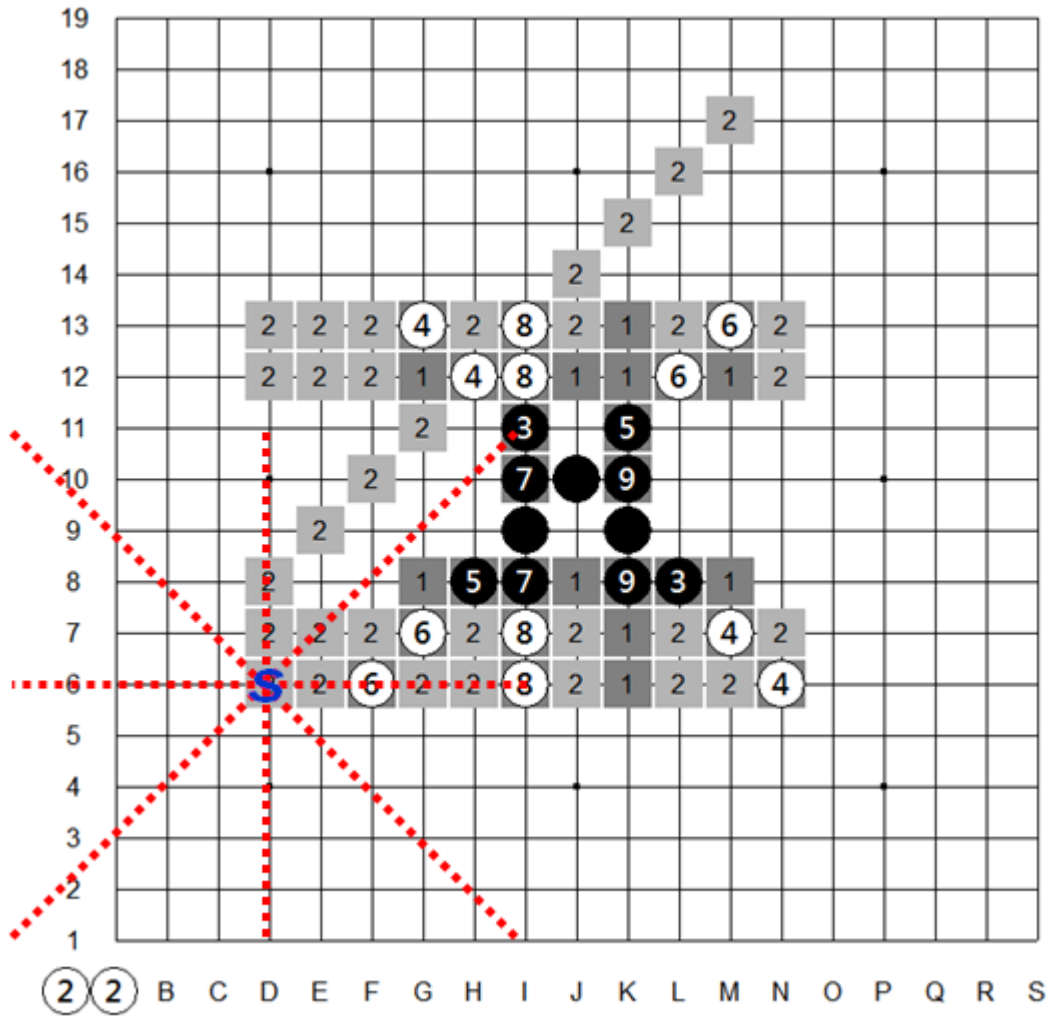


Figure 26. For Defender's first square  $s$ , the dash line indicates the possible area for the second square  $s'$  that may form counter-threat segments.

**Definition 5.** Let  $\Psi = \langle Z_1, Z_2, \dots, Z_r \rangle$  be a sequence of zones. For each square  $s \in \dot{Z}_i$ , where  $\dot{Z}_i = Z_i \setminus Z_{i-1}$  and  $2 \leq i \leq r'$ ,  $\dot{Z}_i^*(s)$  is constructed from  $\dot{Z}_i$  as follows. For each squares  $s' \in \dot{Z}_i$ , where  $s' \neq s$ , if  $s'$  and  $s$  are both in a same active segment of Defender, put  $s'$  into  $\dot{Z}_i^*(s)$ . ■

From Definition 5, for squares  $s$  and  $s'$  in  $\dot{Z}_2$ , if  $s' \notin \dot{Z}_2^*(s)$ ,  $s'$  and  $s$  must not in a same active segment of *Connect6*. This implies that if  $\Psi$  is in  $RZ(P)$ , Attacker wins in  $P \oplus M'_D(s, s')$



by replaying.

**Corollary 4.** Assume  $\Psi$  is in  $RZ(P)$ . Let  $\varphi = \langle s, s' \rangle$ , where both  $s$  and  $s' \in \dot{Z}_2$ . If  $s' \notin \dot{Z}_2 * (s)$ , Attacker wins in  $P$  by replaying. ■

### 4.2.1 The Proposed Verifier $V_{C6-02}$

This subsection presents a verifier, named  $V_{C6-02}(P,S)$ , improved from  $V_{C6-01}(P,S)$ . The verifier  $V_{C6-02}(P,S)$  is described in Subsections 4.2.1.1, 4.2.1.2 and 4.2.1.3 respectively for three distinct kinds of  $P$ , namely endgame positions, positions in Attacker's turn and positions in Defender's turn. Finally, Subsection 4.2.2 concludes with Theorem 4, showing that the verifier satisfies Property RZV in all cases.

#### 4.2.1.1. Endgame Positions

If Attacker does not win in the endgame position  $P$ , the verifier simply returns the value 0. If Attacker wins in  $P$  (i.e., Attacker has a win segment in  $P$ ), the verifier returns 1 and constructs  $\Psi(P)$  in the following operation.

**O2-EP-1.** For each active segment  $G$  of Defender containing *exactly*  $i$  unoccupied squares, these squares in  $G$  are all added into  $Z_i(P)$  or higher-order zones; that is,  $Z_j(P)$  for all  $j \geq i$ . In other words, for each active segment  $G$  of Defender containing at most  $i$  unoccupied squares, add all of these squares in  $G$  into  $Z_i(P)$ .

**Lemma 24.** Assume  $P$  to be an endgame position. Property RZV is satisfied for  $V_{C6-02}(P,S)$ .

**Proof.** Similar to Lemma 17, therefore omitted. ■

#### 4.2.1.2. Positions in Attacker's Turn

In such positions, Attacker simply follows strategy  $S$  to make the move  $S(P)$  in  $P$ . Let  $P_A$  denote  $P \oplus S(P)$ . This verifier first performs  $V_{C6-O2}(P_A, S)$  recursively. If  $V_{C6-O2}(P_A, S)$  returns the value 0, this verifier  $V_{C6-O2}(P, S)$  also returns 0. On the other hand, if  $V_{C6-O2}(P_A, S)$  returns 1, this verifier  $V_{C6-O2}(P, S)$  returns 1, too; and constructs  $\Psi(P)$  in the following operation.

**O2-AT-1.** Let  $\Psi(P) = \Psi(P_A) \cup Z_S$ , where  $Z_S = \{s \mid s \in S(P)\}$ .

**Lemma 25.** Assume a position  $P$  in Attacker's turn. From the above, assume that  $V_{C6-O2}(P_A, S)$  satisfies Property RZV, where  $P_A = P \oplus S(P)$ . This verifier  $V_{C6-O2}(P, S)$  satisfies Property RZV.

**Proof.** Similar to Lemma 18, therefore omitted. ■



#### 4.2.1.3. Positions in Defender's Turn

For positions in Defender's turn, the following lemma shows a very important property used in this subsection.

**Lemma 26.** Assume a position  $P$  in Defender's turn. For a given sequence of zones  $\Psi$ , assume that for all Defender moves  $M_D$  there exists some  $\Psi_D$  such that  $\Psi_D \subseteq \Psi$  and  $\Psi_D$  is in  $RZ(P \oplus M_D)$ . Then  $\Psi$  is in  $RZ(P)$ .

**Proof.** Similar to Lemma 19, therefore omitted. ■

In Connect6, the position  $P$  (in Defender's turn) can be classified into the following four cases. The number of Attacker threats in  $P$  is (1) three or more, (2) two, (3) one and (4) zero. The four cases are discussed respectively in the following four subsections.

#### 4.2.1.3.1. Three Threats or More

In this case, Attacker is sure to win by simply following the strategy,  $S_{3T}$ , as follows. For each Defender move, since the move must leave some threat segments unblocked, Attacker wins simply by making a win segment from the unblocked one. Since the strategy is a sure win, the verifier returns the value 1 and constructs the zones (initialized to be empty) in the following operations.

**O2-T3-1.** Add all unoccupied squares  $s$  on threat segments into all  $Z_i(P)$ .

**O2-T3-2.** For each active segment  $G$  of Defender containing *exactly*  $i + 2$  unoccupied squares, all these squares in  $G$  are added into all  $Z_j(P)$  or higher-order zones.

In other words, for each active segment  $G$  of Defender containing at most  $i + 2$  unoccupied squares, add all these squares in  $G$  into  $Z_i(P)$ .

**Lemma 27.** Assume that Defender is to move and Attacker has three or more threats in  $P$ . The verifier described above satisfies Property RZV.

**Proof.** Similar to Lemma 20, therefore omitted. ■

#### 4.2.1.3.2. Two Threats

When Attacker has two threats in  $P$ , Defender must defend by blocking the two threats. In this case, the verifier performs the following operations.

**O2-T2-1.** For each Defender move  $M_D$  that blocks the two threats, perform the following.

- a. Return the value 0 if the recursive  $V_{C6-O2}(P_D, S)$  returns the value 0, where  $P_D = P \oplus M_D$ .
- b. Let  $\Psi(P) = \Psi(P) \cup \Psi(P_D)$ .

**O2-T2-2.** Continue to construct zones by both operations O2-T3-1 and O2-T3-2, and return 1.

**Lemma 28.** From the above, assume that Defender is to move and Attacker has two threats in  $P$ . Assume that all the recursive  $V_{C6-O2}(P_D, S)$  in operation O2-T2-1 satisfy Property RZV. Then, the verifier  $V_{C6-O2}(P, S)$  satisfies Property RZV too.

**Proof.** Similar to Lemma 21, therefore omitted. ■

#### 4.2.1.3.3. One Threat

When Attacker has one threat, Defender must defend by blocking the threat. In this case, the verifier performs the following operations.

**O2-T1-1.** For each normal critical defense (defined in Section 1.2),  $M_{D,\phi}(s)$  where square  $s$  blocks the threat, perform the operation of semi-null-move proof search as follows.

- a. Return the value 0, if the recursive  $V_{C6-O2}(P_s, S)$  returns 0 where  $P_s = P \oplus M_{D,\phi}(s)$ .
- b. Let  $\Psi(P) = \Psi(P) \cup (\Psi(P_s) \ll 1)$ .
- c. For each defensive move  $M_D(s, s')$ , where  $s' \in Z_I(P_s)$ , perform both operations O2-T2-1.a and O2-T2-1.b.

**O2-T1-2.** For all relaxed critical defenses  $M_D(s, s')$ , perform both operations O2-T2-1.a and O2-T2-1.b.

**O2-T1-3.** Perform both operations O2-T3-1 and O2-T3-2, and return 1.

**Lemma 29.** From the above, assume that Defender is to move and Attacker has one threat in  $P$ . Assume that all the recursive  $V_{C6-O2}$  in both operations O2-T1-1 and O2-T1-2 satisfy Property RZV. Then, the verifier  $V_{C6-O2}(P,S)$  satisfies Property RZV too.

**Proof.** Similar to Lemma 22, therefore omitted. ■

#### 4.2.1.3.4. No Threats

When Attacker has no threats, it becomes more complicated since Defender has much more freedom to move. In this case, the verifier makes use of the constructed relevance zones to minimize the search space in the following operations.

- O2-T0-1.** Return the value 0 if  $V_{C6-O2}(P_\phi,S)$  returns 0, where  $P_\phi = P \oplus M_{D,\phi\phi}$ .
- O2-T0-2.** Let  $\Psi(P) = \Psi(P_\phi) \ll 2$ .
- O2-T0-3.** For each square  $s$  in  $Z_1(P_\phi)$ , perform the semi-null move proof search, as in operations O2-T1-1.a to O2-T1-1.c.
- O2-T0-4.** For each square  $s$  in  $\dot{Z}_2(P_\phi)$ , where  $\dot{Z}_2(P_\phi) = Z_2(P_\phi) \setminus Z_1(P_\phi)$ , perform the operation which satisfies Property INZ as follows.
  - a.** For each defensive move  $M_D(s,s')$ , where  $s' \in \dot{Z}_2 \neq(s)$  and  $s' \neq s$ , perform both operations O2-T2-1.a and O2-T2-1.b.
- O2-T0-5.** Return 1.

**Lemma 30.** Assume that Defender is to move and Attacker has no threats in  $P$ . From the above, assume that all recursive  $V_{C6-O2}$  in operations O2-T0-1, O2-T0-3 and O2-T0-4 satisfy Property RZV. Then, the verifier  $V_{C6-O2}(P,S)$  also satisfies Property RZV.

**Proof.** Assume that this verifier  $V_{C6-O2}(P,S)$  returns 1. For this lemma, it suffices to prove that the constructed  $\Psi(P)$  is in  $RZ(P)$ . Since  $V_{C6-O2}(P,S)$  returns 1, all the recursive  $V_{C6-O2}$  in operations O2-T0-1, O2-T0-3 and O2-T0-4 must also return 1. Since these recursive  $V_{C6-O2}$ ,

say for position  $P'$ , satisfy Property RZV from this lemma, the constructed zones  $\Psi(P')$  are in  $RZ(P')$ .

To prove  $\Psi(P) \in RZ(P)$ , it suffices to prove from Lemma 26 the following: For all Defender moves  $M_D$ , there exists some  $\Psi_D$  such that  $\Psi_D$  is in  $RZ(P \oplus M_D)$  and  $\Psi_D \subseteq \Psi(P)$ .

All Defender moves  $M_D$  are classified into the following cases:

1. All Defender moves  $M_D(s, s')$  where  $s \in \neg_{P_\phi}(Z_2(P_\phi))$  and  $s' \in \neg_{P_\phi}(Z_2(P_\phi))$ . From the first paragraph in this proof,  $\Psi(P_\phi)$  is in  $RZ(P_\phi)$ . Since  $s \in \neg_{P_\phi}(Z_2(P_\phi))$  and  $s' \in \neg_{P_\phi}(Z_2(P_\phi))$ ,  $\Psi(P_\phi) \ll 2$  is in  $RZ(P_\phi + \sigma_D(s) + \sigma_D(s'))$  from Lemma 14. Since  $P_\phi + \sigma_D(s) + \sigma_D(s') = P \oplus M_D(s, s')$ ,  $\Psi(P_\phi) \ll 2$  is also in  $RZ(P \oplus M_D(s, s'))$ . In addition,  $(\Psi(P_\phi) \ll 2) \subseteq \Psi(P)$  from operation O2-T0-2. Thus,  $\Psi(P_\phi) \ll 2$  is  $\Psi_D$ .
2. All Defender moves  $M_D(s, s')$  where  $s \in Z_1(P_\phi)$ . By following the proof for Case 1 (including Subcases 1.a and 1.b) in Lemma 22, we obtain that there exists some  $\Psi$  in  $P \oplus M_D(s, s')$  for all  $s'$  such that  $\Psi \subseteq \Psi(P)$ . The details are omitted. ■
3. All Defender moves  $M_D(s, s')$  where  $s \in \check{Z}_2(P_\phi)$  and  $s' \in \check{Z}_2(P_\phi)$ . This case is separated into the following two subcases.
  - a.  $s' \in \check{Z}_2 \not\#(s)$ . Let  $P_D$  denote  $P \oplus M_D(s, s')$ . The zones  $\Psi(P_D)$  is constructed in operation O2-T2-1.b, and is in  $RZ(P_D)$  according to the first paragraph of this proof. Since  $\Psi(P_D)$  is merged into  $\Psi(P)$  in O2-T2-1.b, we obtain  $\Psi(P_D) \subseteq \Psi(P)$ . Thus,  $\Psi(P_D)$  is the  $\Psi_D$ .
  - b.  $s' \notin \check{Z}_2 \not\#(s)$ . From the above and Corollary 4, Attacker wins by replaying. Since  $\Psi(P_D) \subseteq (\Psi(P_\phi) \ll 2)$  and  $(\Psi(P_\phi) \ll 2) \subseteq \Psi(P)$ ,  $\Psi(P_D)$  is the  $\Psi_D$ . ■

### 4.2.2 Conclusion for the Verifier $V_{C6-02}$

Theorem 4 (below) concludes that the verifier  $V_{C6-02}(P,S)$  in all cases satisfy Property RZV. Therefore, if  $V_{C6-02}(P,S)$  returns the value 1, the constructed  $\Psi(P)$  is in  $RZ(P)$ , and Attacker wins in  $P$  from Corollary 3.

**Theorem 4.** The verifier  $V_{C6-02}(P,S)$  satisfies Property RZV in all cases.

**Proof.** By induction, the verifier  $V_{C6-02}(P,S)$  satisfies Property RZV in all cases from Lemma 24 to Lemma 30. ■

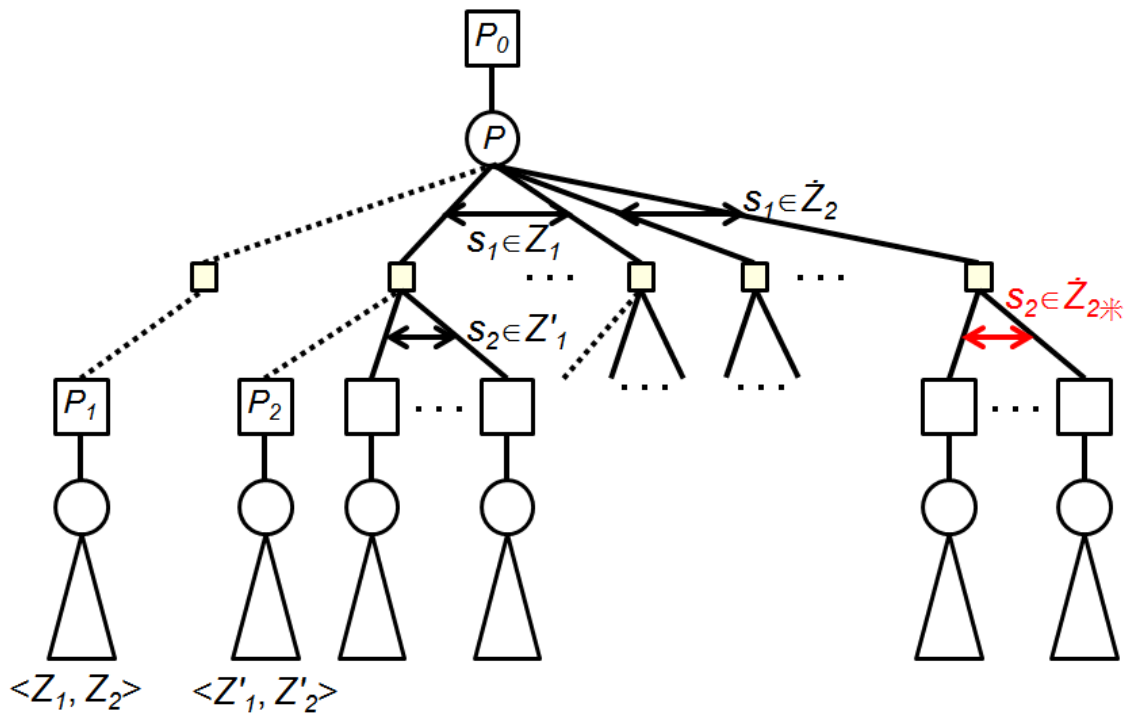


Figure 27. An example proof search tree, where  $\check{Z}_2 = Z_2 \setminus Z_1$ , for the Verifier  $V_{C6-02}(P,S)$ .

Figure 27 shows an example proof search tree that gives an overview for the Verifier  $V_{C6-02}(P,S)$ .

## Chapter 5 Experiments

In Chapter 3 and Chapter 4, we present verifiers  $V_{C6}(P,S)$ ,  $V_{C6-O1}(P,S)$  and  $V_{C6-O2}(P,S)$  to verify whether Attacker wins in a Connect6 position  $P$  by following strategy  $S$ . However, in order to solve positions, we still need to provide the verifier with winning strategies  $S$ . Winning strategies can be provided in the following three ways.

1. Let human experts offer the winning strategies manually.
2. Let programs find the winning strategies automatically.
3. Find the winning strategies by mixing the above two.

Traditionally, human experts used the first way to claim that some positions are winning, e.g., Go-Moku and Renju [44]. However, it becomes complicated and tedious for human players to traverse all positions to prove it thoroughly. Hence, it is more feasible to solve these positions by programs using the second way. However, programs may not be smart enough sometimes to find the correct winning moves. Therefore, some researchers chose the third way by following human experts' suggestions for some opening moves and then letting programs solve the subsequent moves. For example, Allis [1][2] solved Go-Moku in the free style, and Wágner and Virág [53] solved Renju. In Section 5.1, we developed some assistant programs to help find the winning strategies for Connect6. In Section 5.2, we illustrate our new proof search method in Chapter 3 by solving the positions in Figure 7 (a) and Figure 7 (b). Finally, we give more results in Section 5.3.



## 5.1 Assistant Programs

This section describes our assistant programs.

### 5.1.1 Solver

Given a position  $P$  in Attacker's turn, a solver is to return a winning move as well as the relevance zones, if found; and, otherwise, a null move is returned to indicate failure of finding a winning move. A solver of finding a VCDT strategy, denoted by  $S_{VCDT}$ , is described as follows.

1. If there exist connect-six moves or triple-threat-or-higher moves, simply choose one of them to win.
2. Evaluate all the double-threat moves and choose some *good* ones for further expansion (according to the evaluations).
3. For each chosen move  $M$ , return  $M$  if  $V_{C6}(P \oplus M, S_{VCDT})$  returns 1.
4. Return the null move to indicate failure of finding a winning move.

A solver of finding a VCST (VCNT) is similar to the above, except that single-threat (non-threat) moves are also evaluated and chosen at Step 2. Actual solvers are implemented in a more complicated way to reduce the size of search tree and control the timing. For example, the techniques of iterative deepening and transposition table are normally incorporated.

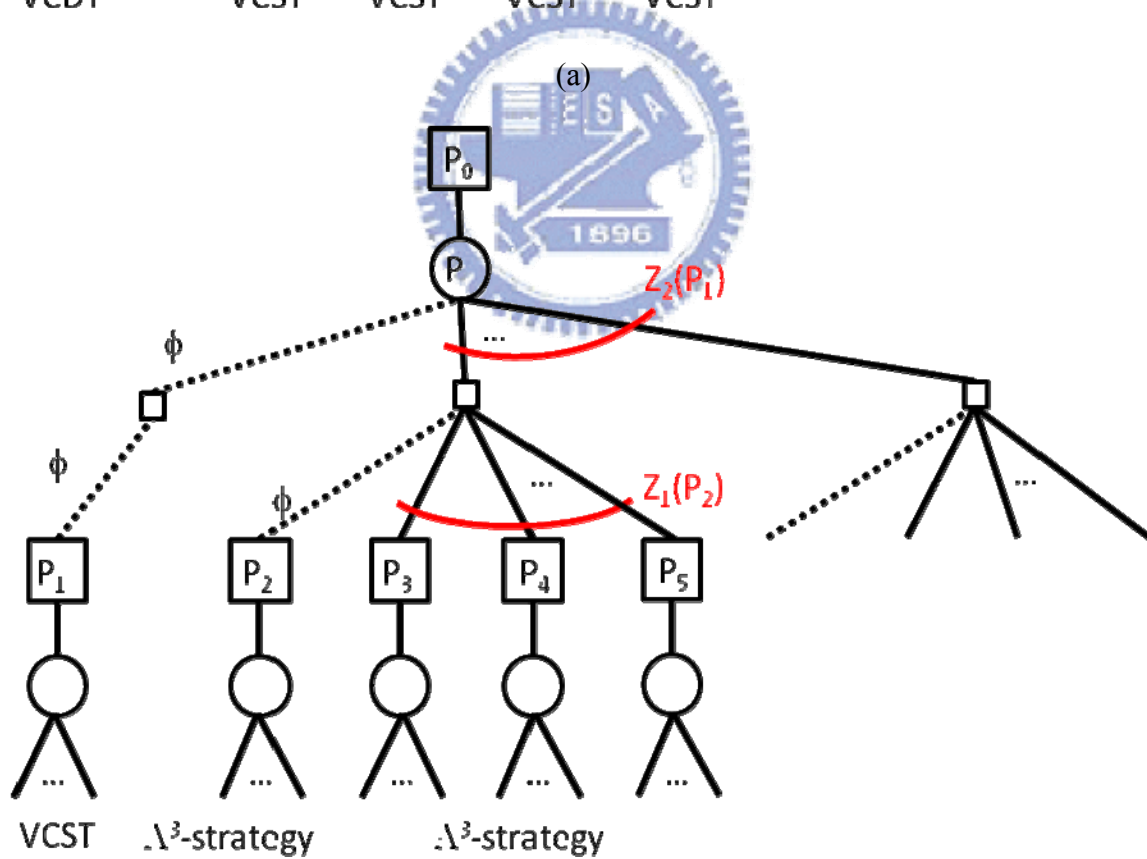
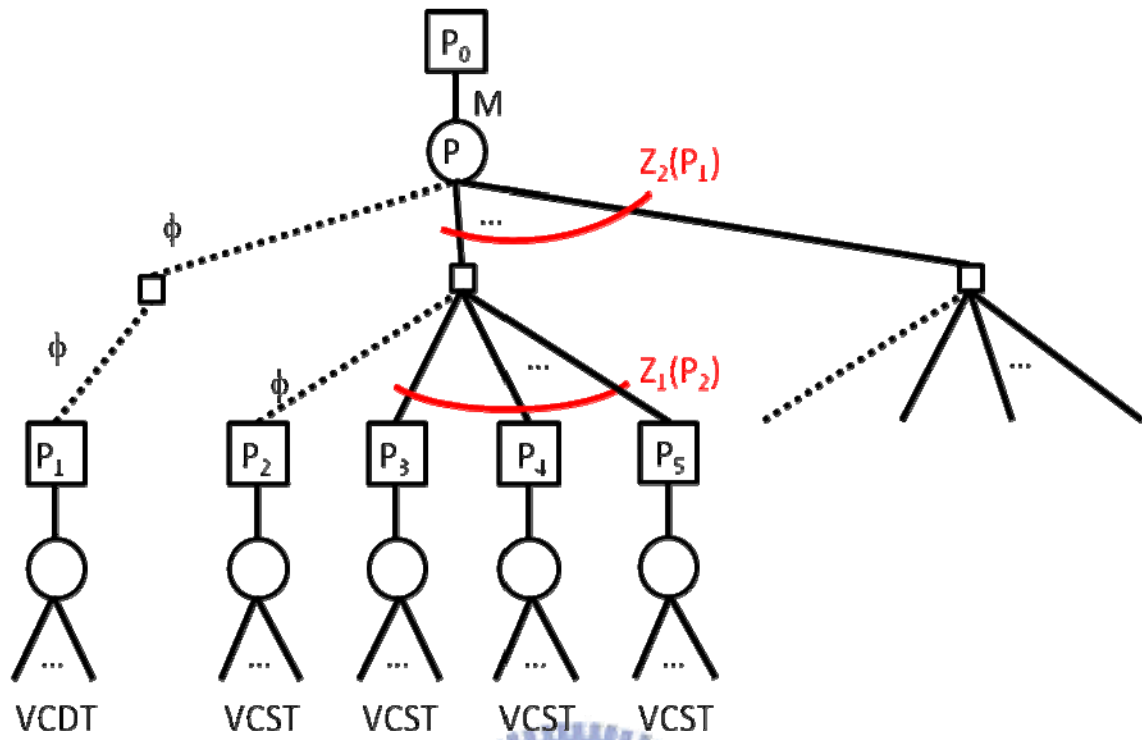
In this thesis, we implemented a solver with VCDT, named *VCDT-Solver*, and another solver with VCST, named *VCST-Solver*. More accurately, the *VCDT-Solver* is to find a  $\Lambda^1$ -strategy, while the *VCST-Solver* is to find a  $\Lambda^2$ -strategy. Our *VCST-Solver* also tends to

find VCDTs, if any, unless some single-threat moves are evaluated to be much better. Currently, this solver is able to find a  $\Lambda^2$ -strategy up to depth 25 where the size of the longest path with  $\Lambda^2$ -moves is 13. This solver was also incorporated into our Connect6 program *NCTU6*, which won the gold in the 11<sup>th</sup> and 13<sup>th</sup> Computer Olympiads [59][67] in 2006 and 2008, respectively; and also won eight games and lost none against top Connect6 players in Taiwan in 2009 [30]. From our experiences, *VCST-Solver* is able to find  $\Lambda^2$ -strategies, if any, in most cases accurately.

### 5.1.2 Verifier

Regarding solvers for  $\Lambda^3$ -strategies or strategies of higher orders, the time complexities become much higher, since the numbers of defensive moves to be verified grow much higher. Therefore, we did not implement it directly.

First, we implemented a verifier, named *NCTU6-Verifier*, to verify whether Attacker wins for all Defender moves. In other words, given a position  $P$  in Defender's turn as shown in Figure 28 (a) below, the verifier uses *VCDT-Solver* for null moves and *VCST-Solver* for all semi-null moves and non-null moves. If null and semi-null moves are all solved, then move  $M$  (from the parent of  $P$  to  $P$ ) in Figure 28 (a) is an Attacker  $\Lambda^3$ -move. If some non-null moves are not solved by *VCST-Solver*, these moves are reported or generated. Note that Defender  $\Lambda^3$ -moves must be reported. Since our *VCST-Solver* can find  $\Lambda^2$ -strategies accurately in most cases, most reported moves are Defender  $\Lambda^3$ -moves in our experiments.



(b)

Figure 28. (a) A proof search tree of *NCTU6-Verifier* and (b) the verifier of one higher order.

When our Connect6 program *NCTU6* mentioned above cannot find  $\Lambda^2$ -strategies (*VCSTs*), *NCTU6* then chooses some *promising* moves including non-threat moves using heuristic evaluations. The details of heuristic evaluations are beyond the scope of this thesis and therefore omitted.

Since *NCTU6* may not be able to find winning moves all the time, human experts are allowed to help find winning moves. (Like [1][2] and [53], knowledge of human experts were utilized to help solve Go-Moku and Renju, respectively.) Hence, the above programs, such as *NCTU6* and *NCTU6-Verifier*, were integrated into a Connect6 editor named Connect6Lib [14], modified from Renlib [42], in order to accommodate hints from human experts. In the integrated system [57][58], the users (human experts) are allowed to suggest some Attacker moves directly or let *NCTU6* suggest *possibly good* moves in a designated position. Then, for suggested moves, users invoke *NCTU6-Verifier* to verify and report all the defensive moves (most are  $\Lambda^3$ -moves). Then, users repeat the above for the subsequent moves, until a  $\Lambda^3$ -strategy is found.

Second, for  $\Lambda^4$ -strategies, the integrated system (on top of the editor Connect6Lib) needs to maintain a global verifier and modify the search by incrementing the order by one as shown in Figure 28 (b).

### 5.1.3 Desktop Grids and Volunteer Computing for Connect6

In this subsection, we discuss our proposed desktop grids and volunteer computing for Connect6 [57][58]. *Grid computing* [20] has recently become a promising trend for both high performance and high throughput computing. Applications include scientific computing and bioinformatics. Many universities, research institutes, and commercial companies have been devoted to the development of related technologies and applications [4][8][19][20][21][26][68]. Among these grid computing models and applications, *desktop grids* [4][47] were developed for volunteer computing which aimed to harvest Internet-scale idle computing resources for speeding up high throughput applications.

In contrast to most current grid computing applications, the applications investigated in this subsection are related to games, more specifically for Connect6 [65][66]. In the Connect6 applications described in this subsection, huge computation resources are consumed and on-demand responses are required. In order to satisfy these requirements, this subsection proposes and designs a *volunteer-computing-based grid environment* or called a *desktop grid environment* for Connect6 applications. The Connect6 application described in this subsection is to let professional Connect6 players to develop or solve openings, based on two programs *NCTU6* and *NCTU6-Verifier* in Subsections 5.1.1 and 5.1.2. The proposed desktop grid environment is also allowed to be applied to other computation-intensive applications requiring on-demand responses.

Most current desktop grid systems, such as BOINC [4][8], XtremWeb [19][68], adopt the *pull* model. In such systems, one or more centralized databases or global servers normally keep many jobs (most for scientific or engineering applications) for idle workers (desktops). The idle workers automatically request (pull) jobs for execution from the centralized databases or global servers, and in turn may create new jobs and upload to the

databases or the global servers. These available jobs are usually not aborted.

The desktop grid system for Connect6 aims to achieve on-demand computing, since the jobs for Connect6 applications are highly dynamic and may be created and aborted at any time. To better cope with the needs of Connect6 applications, our desktop grid environment features a *push* model and has a close collaboration between Connect6Lib and workers in the grid. Our desktop grid environment is expected to harvest idle resources for free CPU time and use them collectively to meet the real-time response requirement of interactive Connect6 applications.

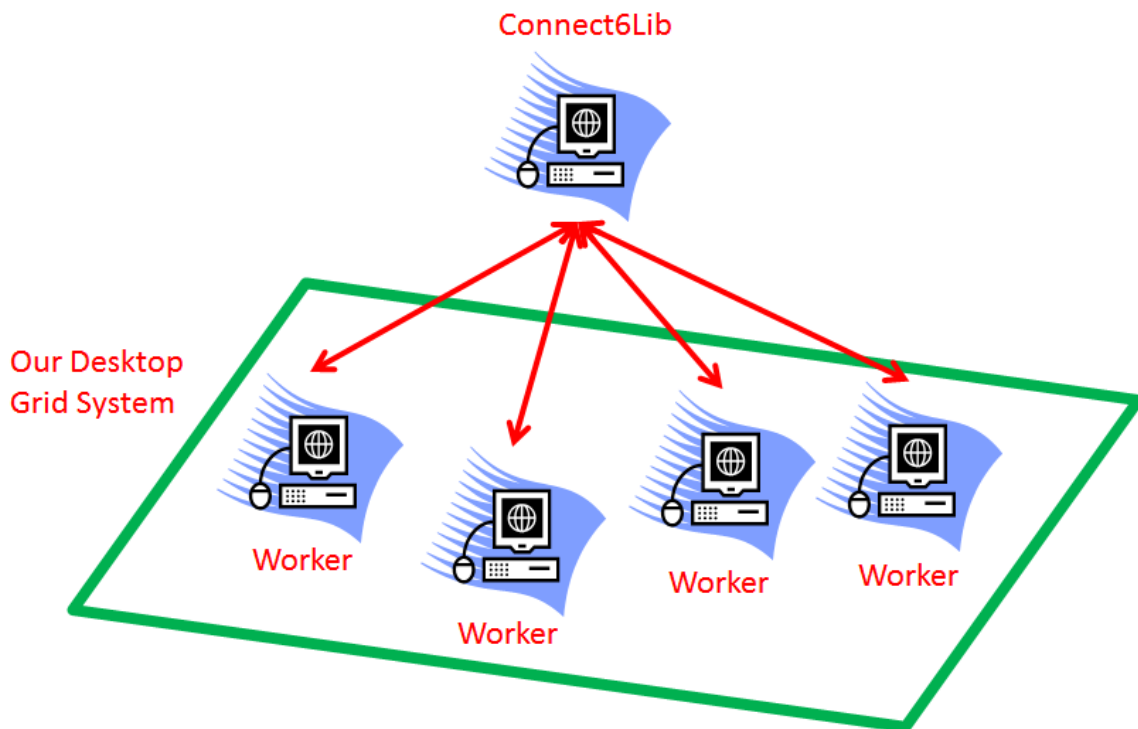


Figure 29. Desktop grid architecture.

Unlike many other desktop grid systems that are normally based on databases in the pull model, Connect6Lib is directly connected to desktops in our current environment as shown in Figure 29. When Connect6 jobs, NCTU6s and Verifiers, are created, these jobs are

sent to remote desktops to run. All messages generated by these jobs are directly sent back (pushed back) to Connect6Lib to create more branches. The push model is used, since users in the application expect to receive responses real time so that they can decide where to exploit next. For this purpose, the communication between Connect6Lib and workers is connection-oriented, using TCP.

In the future we plan to extend our current implementation to support different applications, and support the automation of helping openings or solving positions.



### 5.1.4 Job-Level Proof-Number Search for Connect6

This subsection describes a new approach for proof number (PN) search, named job-level proof-number (JL-PN) search [64]. Proof-number (PN) search, proposed by Allis *et al.* [1][3], is a kind of best-first search algorithm that was successfully used to prove or solve theoretical values [22] of game positions for many games [1][2][3][23][43][45][46][52], such as Connect-Four, Gomoku, Renju, Checkers, Lines of Action, Go, Shogi. Like most best-first search, PN search has a well-known disadvantage, the requirement of maintaining the whole search tree in memory. Therefore, many variations [9][29][35][36][45][54] were proposed to avoid this problem, such as PN<sup>2</sup>, DF-PN, PN\*, PDS, and parallel PN search [27][43] were also proposed. For example, PN<sup>2</sup> used two-level PN search to reduce the size of the maintained search tree.

The JL-PN search, where the PN search tree is maintained by a process, the *client* in this subsection, and search tree nodes are evaluated or expanded by *heavy-weight jobs*, which can be executed remotely in a parallel system. Heavy-weight jobs take normally tens of seconds or more (perhaps up to one day).

For simplicity of discussion about proof-number (PN) search, we follow in principle the definitions and algorithms in [1][3]. PN search is based on an AND/OR search tree where each node  $n$  is associated with proof/disproof numbers,  $p(n)$  and  $d(n)$ , which represent the minimum numbers of nodes to be expanded to prove/disprove  $n$ . The values  $p(n)/d(n)$  are  $0/\infty$  if the node  $n$  is proved, and  $\infty/0$  if it is disproved. PN search repeatedly chooses a leaf called *the most-proving node (MPN)* to expand, until the root is proved or disproved. The details of choosing MPN and maintaining the proof/disproof numbers can be found in [1][3] and therefore is omitted in this subsection. If the selected MPN is proved (disproved), the proof (disproof) number of the root of the tree is decreased by one.



Our JL-PN search is parallel PN search with the following two features. First, well-written programs such as NCTU6 and Verifier are used to expand and generate MPNs. These programs are viewed as jobs, sent to and done by free workers in a desktop grid. Second, multiple MPNs are allowed to be chosen simultaneously and therefore can be done by different workers in parallel.



## 5.2 Illustration of Solving Positions

In this section, we illustrate the proof search method in Chapter 3 by solving the two positions in Figure 7 (a) and Figure 7 (b). First, consider the one in Figure 7 (a). The position is solved by simply running *NCTU6-Verifier*. In the proof search tree shown in Figure 30 (below),  $P$  indicates the position at 7 in Figure 7 (a);  $P_0$ , the position at 6;  $P_1$ , the position after a null move;  $P_2$ , the position after the semi-null move 8 in Figure 8 (b); and  $P_{21}$ , the position after another semi-null move at 10 in Figure 8 (b). As can be seen, Attacker wins in a  $\Lambda^3$ -strategy.

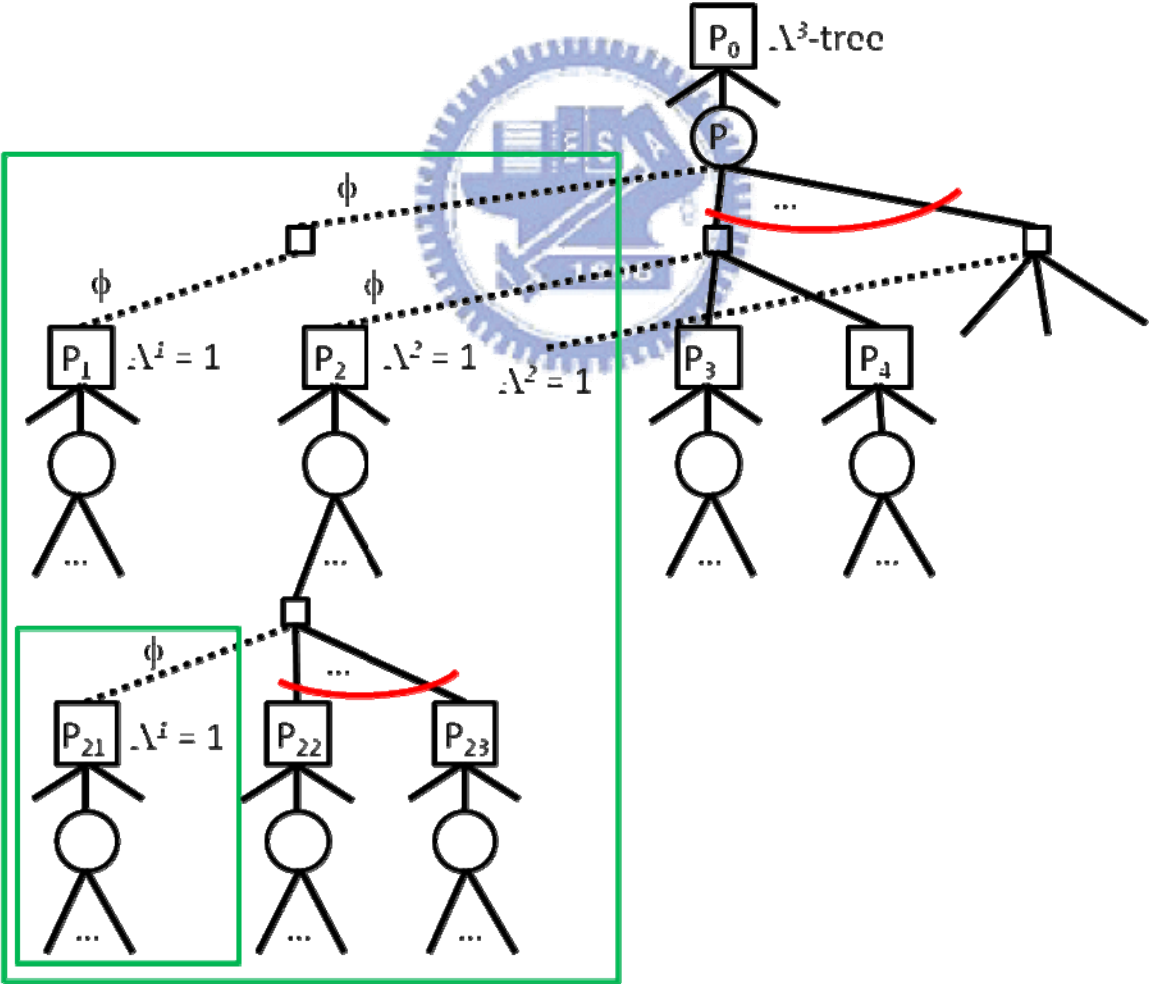


Figure 30. The proof search tree for the position in Figure 7 (a).

Second, consider the position in Figure 7 (b), which is much more complicated than the previous one. This position is solved via the integrated system supporting  $\Lambda^4$ -trees, as described in Section 5.1. In the proof search tree shown in Figure 31 (below),  $P$  indicates this position,  $P_1$  does the position after a null move, and  $P_2$  does the position after a semi-null move at 7 in Figure 9 (b). Initially, let *NCTU6-Verifier* of one higher order run in  $P$ . Since *VCST-Solver* is able to find the winning move for  $P_1$ , Defender (Black) should place at least one stone in zone  $Z_2(P_1)$ . Consider one square  $s$  in  $Z_2(P_1)$ , say square 7 in Figure 9 (b). For the semi-null move at 7, choose move 8 and then use *NCTU6-Verifier* (without raising one order) to derive that Attacker wins at 8. Thus, move 8 is a  $\Lambda^3$ -move. By verifying all null and semi-null moves in  $P$ , we show that move 6 in Figure 7 (b) is a  $\Lambda^4$ -move (from Definition 1).

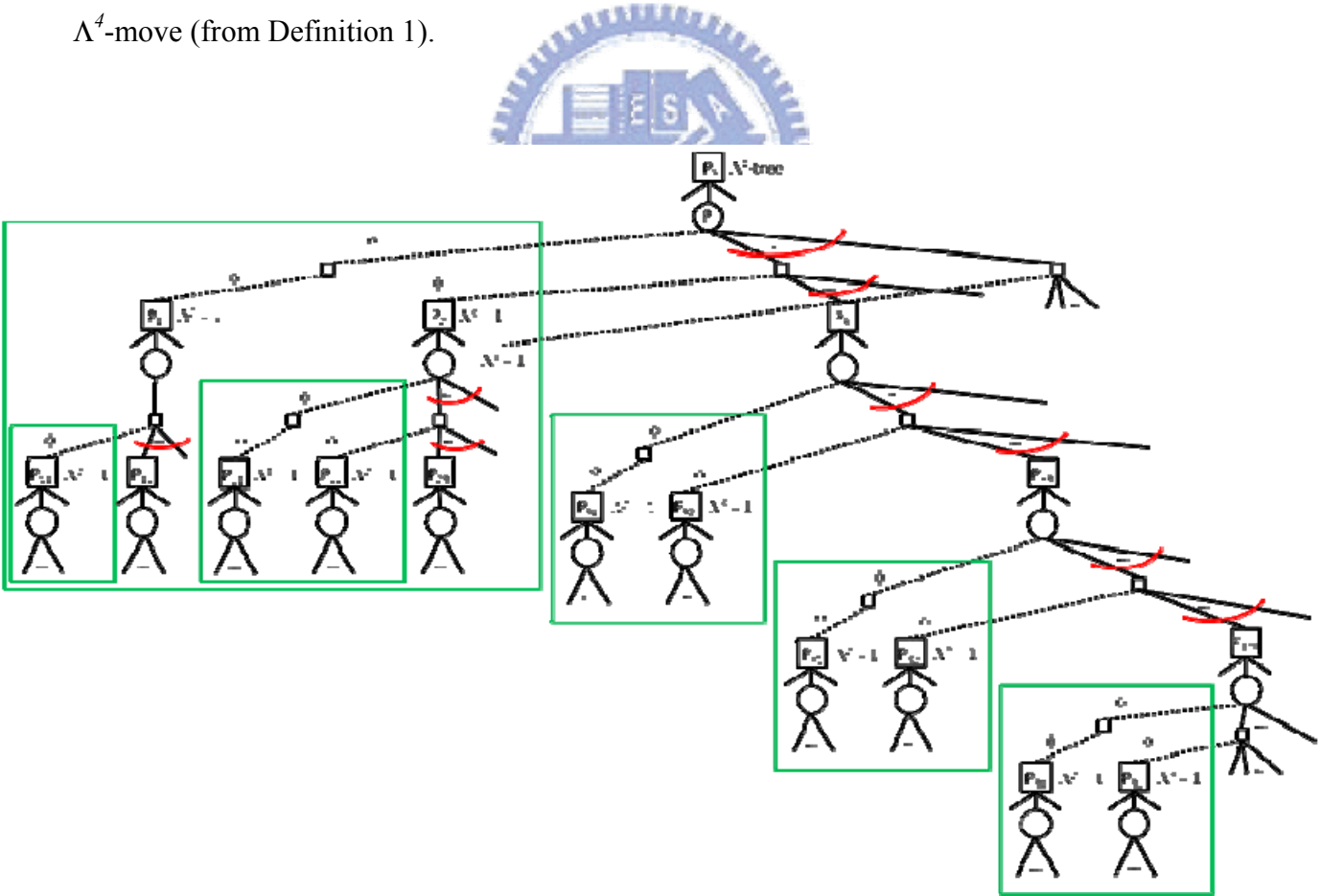


Figure 31. The proof search tree for the position in Figure 7 (b).

Furthermore, Attacker is shown to win at 6 in a  $\Lambda^4$ -strategy as follows. In our experiment, Attacker wins for all defensive (non-null) moves by finding  $\Lambda^3$ -strategies. For example, for move 7 in Figure 32 (below), *NCTU6-Verifier* is recursively employed to find a  $\Lambda^3$ -strategy, where moves 8 to 12 are shown to be  $\Lambda^3$ -moves.

In the proof search tree shown in Figure 31, we found three semi-null moves that are  $\Lambda^3$ -moves with value 1 (like  $P_2$  which is also 7 in Figure 9 (b)), and 569 Defender  $\Lambda^3$ -moves in total. Move 12 in Figure 32 is the deepest  $\Lambda^3$ -move. In this experiment, human experts helped find 26 winning non-threat moves, including move 6 discovered by Huang [25].

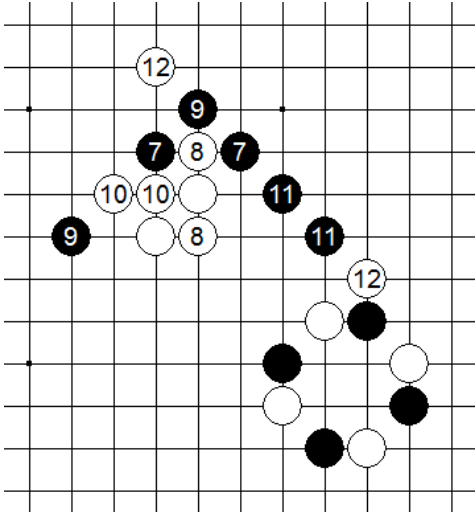


Figure 32. A sequence of  $\Lambda^3$ -move starting from 7.

Now, the question is whether there exist more cases requiring  $\Lambda^4$ -strategies like the one in Figure 7 (b). Since the one in Figure 7 (b) is the only one that we found so far, it is still an open problem to find some more.

### 5.3 Results

Initially, we had human experts use the integrated system to help us solve about 10 more positions. Wu *et al.* [64] had recently automated with success the proof process by developing a new search algorithm, called job-level proof-number (JL-PN) search (described in Subsection 5.1.4). Using the JL-PN search together with our RZOP search, we solved many more positions, up to 65 positions in total, with  $\Lambda^3$ -strategy, within a couple of months. The details of the 65 positions are listed in Appendix A. The detail results are listed in Appendix B and C. All experiments ran on Intel Pentium Dual 2.00 GHz machines and were performed on  $19 \times 19$  boards that most current Connect6 tournaments use. Besides, we develop six verifiers as follows.

- $V_{wu}$ : implement the simple proof search method in Wu *et al.*, 2006.
- $V_{C6}$ : implement the RZOP search method.
- $V_{C6-01}$ : implement the SRZOP search method in Section 4.1.
- $V_{C6-02}$ : implement the SRZOP search method in Section 4.2.

Before we discuss the 65 positions, we illustrate the three puzzles, shown in Figure 6, Figure 7. The purpose of the verifier  $V_{C6}$  is to solve positions, therefore we measure solvability first. Since the verifiers  $V_{C6-02}$  and  $V_{C6-01}$  are developed from  $V_{C6}$ , they have same solvability. Table 1 (below) shows that  $V_{C6-02}$ ,  $V_{C6-01}$  and  $V_{C6}$  solve all three puzzles, while  $V_{wu}$  can only solve the puzzle, *Connect(6,2,3)*, in Figure 6.

<b>Solvability</b>	$V_{C6-02}$ , $V_{C6-01}$ and $V_{C6}$	$V_{Wu}$
Figure 6	yes	yes
Figure 7 (a)	yes	no
Figure 7 (b)	yes	no

Table 1. The solvability of verifiers for the three puzzles. “yes” means solved and “no” means unsolved.

Then, we compare the performance to solve puzzles between RZOP and SRZOP verifiers in number of nodes and time (in seconds).

<b>Number of nodes</b>	$V_{C6-02}$	$V_{C6-01}$	$V_{C6}$
Figure 6	35,425	43,689	59,895
Figure 7 (a)	573,818	583,541	808,511
Figure 7 (b)	51,898,841	58,227,391	81,636,536

(a)

<b>Speedups</b>	$V_{C6-02}$	$V_{C6-01}$	$V_{C6}$
Figure 6	1.69	1.37	—
Figure 7 (a)	1.41	1.39	—
Figure 7 (b)	1.57	1.40	—

(b)

Table 2. (a) The statistics of verifiers for the three puzzles in number of nodes. (b) Speedups compare to  $V_{C6}$ .

Time (in seconds)	$V_{C6-02}$	$V_{C6-01}$	$V_{C6}$
Figure 6	20.92	28.69	36.77
Figure 7 (a)	147.52	158.30	192.91
Figure 7 (b)	17,919.70	23,656.79	30,184.90

(a)

Speedups	$V_{C6-02}$	$V_{C6-01}$	$V_{C6}$
Figure 6	1.76	1.28	—
Figure 7 (a)	1.31	1.22	—
Figure 7 (b)	1.68	1.28	—

(b)

Table 3. (a) The statistics of verifiers for the three puzzles in time (in seconds). (b)

Speedups compare to  $V_{C6}$ .

Table 2 (a) shows the number of nodes used by verifiers to solve the three puzzles. Table 2 (b) shows speedups comparing to the RZOP search method. Table 3 (a) shows the time used by verifiers to solve the three puzzles. Table 3 (b) shows speedups comparing to the RZOP search method. From Table 3, the verifier  $V_{C6-02}$  achieves 1.76 speedups, an improvement, to solve *Connect(6,2,3)* in Figure 6. For the puzzle in Figure 7 (a), the verifier  $V_{C6-02}$  achieves 1.31 speedups. For the hardest puzzle currently in Figure 7 (b), the verifier  $V_{C6-02}$  can achieve 1.68 speedups.

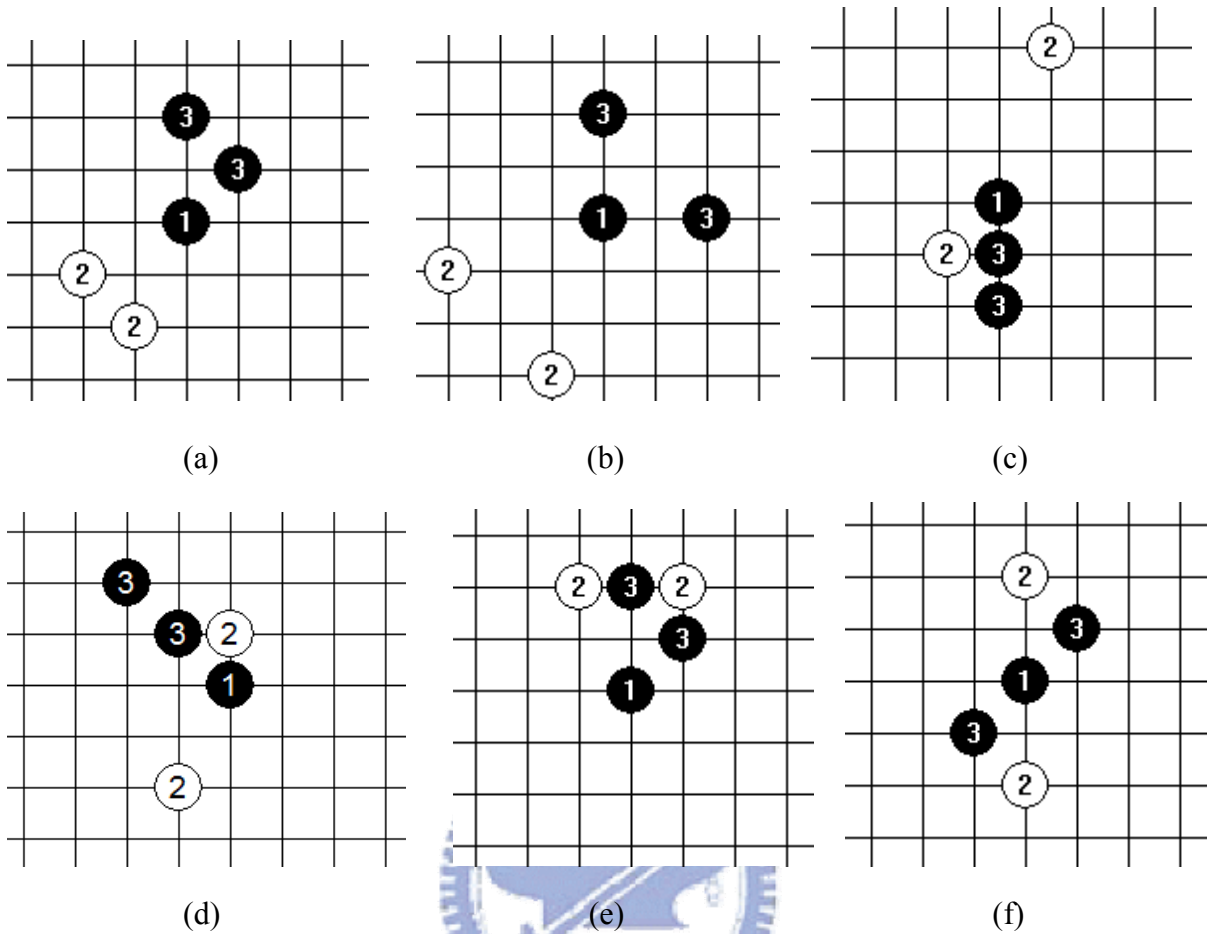


Figure 33. Six openings in which Black wins at 3.

Next, we choose 65 winning positions (shown in Appendix A) which include 12 openings and six of them are shown in Figure 33. In particular, the fifth one, Mickey-Mouse Opening, used to be one of the popular openings before we solved it. *Mickey-Mouse Opening* was so named in [50], since White 2 and Black 1 together look like the face of Mickey Mouse. The sixth one, also called Straight Opening, is another difficult one.

The purpose of the verifier  $V_{C6}$  is to solve positions, therefore we measure sovability first. Since the verifiers  $V_{C6-02}$  and  $V_{C6-01}$  are developed from  $V_{C6}$ , they have same sovability. Table 4 (below) shows that  $V_{C6-02}$ ,  $V_{C6-01}$  and  $V_{C6}$  solve all 65 positions, while  $V_{Wu}$  only solves 31 positions.



Solvability	Total solved	Total unsolved
$V_{Wu}$	31	34
$V_{C6-02}$ , $V_{C6-01}$ and $V_{C6}$	65	0

Table 4. The solvability of verifiers for 65 winning positions.

Then, we compare the performance to solve 65 positions between RZOP and SRZOP verifiers in number of nodes and time (in seconds).

Number of nodes	$V_{C6-02}$	$V_{C6-01}$	$V_{C6}$
65 positions	178,020,119	179,532,383	304,485,291

(a)

Speedups	$V_{C6-02}$	$V_{C6-01}$	$V_{C6}$
65 positions	1.71	1.70	—

(b)

Table 5. (a) The statistics of verifiers for 65 winning positions in number of nodes. (b)

Speedups compare to  $V_{C6}$ .

Time (in seconds)	$V_{C6-02}$	$V_{C6-01}$	$V_{C6}$
65 positions	26,356.62	27,981.87	38,753.57

(a)

Speedups	$V_{C6-02}$	$V_{C6-01}$	$V_{C6}$
65 positions	1.47	1.38	—

(b)

Table 6. (a) The statistics of verifiers for 65 winning positions in time (in seconds). (b)

Speedups compare to  $V_{C6}$ .

Table 5 (a) shows the number of nodes used by verifiers to solve 65 positions. Table 5 (b) shows speedups comparing to the RZOP search method. Table 6 (a) shows the time used by verifiers to solve 65 positions. Table 6 (b) shows speedups comparing to the RZOP search method. From Table 6, the verifier  $V_{C6-02}$  achieves 1.47 speedups to solve 65 positions (listed in Appendix A). Since some of positions are easy to solve, so we choose the 12 openings to measure the performance again.

<b>Number of nodes</b>	$V_{C6-02}$	$V_{C6-01}$	$V_{C6}$
12 openings	66,515,413	67,383,224	161,071,884

(a)

<b>Speedups</b>	$V_{C6-02}$	$V_{C6-01}$	$V_{C6}$
12 openings	2.42	2.39	—

(b)

Table 7. (a) The statistics of verifiers for 12 openings in number of nodes. (b) Speedups compare to  $V_{C6}$ .

<b>Time (in seconds)</b>	$V_{C6-02}$	$V_{C6-01}$	$V_{C6}$
12 openings	9,035.26	9,815.09	18,472.73

(a)

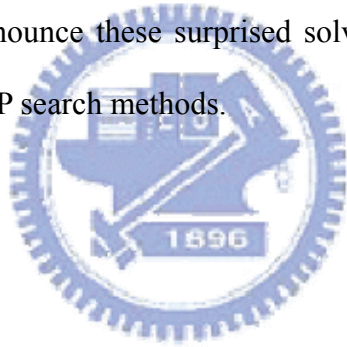
<b>Speedups</b>	$V_{C6-02}$	$V_{C6-01}$	$V_{C6}$
12 openings	2.04	1.88	—

(b)

Table 8. (a) The statistics of verifiers for 12 openings in time (in seconds). (b) Speedups compare to  $V_{C6}$ .

Table 7 (a) shows the number of nodes used by verifiers to solve the 12 openings. Table 7 (b) shows speedups comparing to the RZOP search method. Table 8 (a) shows the time used by verifiers to solve the 12 openings. Table 8 (b) shows speedups comparing to the RZOP search method. From Table 8, the verifier  $V_{C6-02}$  achieves 2.04 speedups to solve the 12 openings.

From the above experimental results, the performance of verifiers is roughly  $V_{C6-02} \geq V_{C6-01} \geq V_{C6}$ . Surprisingly, the verifier  $V_{C6-02}$  can save more than half time to solve harder positions like the 12 openings shown in Appendix A as well as the currently hardest puzzle shown in Figure 7 (b) that is solved by a  $\Lambda^4$ -strategy shown in Figure 31. The experimental results demonstrate a milestone of *NCTU6* and *NCTU6-verifiers* since year 2005 [65][66]. The author is very proud to announce these surprised solvability and performance of the RZOP search method and SRZOP search methods.



## Chapter 6 Conclusions

This thesis proposes a novel, general and elegant proof search method, named Relevance-Zone-Oriented Proof (RZOP) search that uses relevance zones to help solve many positions in Connect6 as well as Connect games. In theory, this method can be applied to Connect games with infinite boards. Practically, this thesis demonstrates the method by solving two typical winning positions in Figure 7 (a) and Figure 7 (b) on  $19 \times 19$  boards, as well as many Connect6 positions and openings in Appendix A. In addition, the method can also be easily incorporated into Connect6 program, such as *NCTU6*.

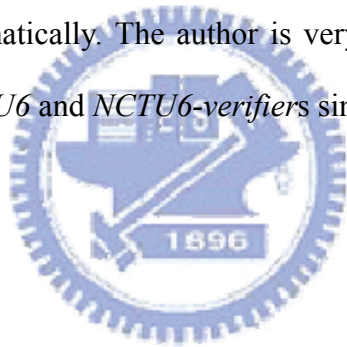
This thesis also leaves some open problems.

- Investigate more winning positions in Connect6 that require  $\Lambda^4$ -strategies, such as the one in Figure 7 (b).
- Investigate whether there exists a  $\Lambda^5$ -strategy in Connect6.
- Apply the new method (in the Appendix D) to solving some real positions in general Connect games.
- Investigate whether dual lambda search [48][49] is useful for Connect6 or Connect games.

Using the JL-PN search together with our RZOP search, we successfully solved up to 65 positions with  $\Lambda^3$ -strategy. The 65 positions include 12 openings; in particular, Mickey-Mouse Opening, which used to be one of the popular openings before we solved it. One might ask whether or when Connect6 on  $19 \times 19$  boards will be solved. So far, we still could not solve tens of the common openings, many of which human experts believed were

well balanced for both players. Hence, the answer to this question is still unknown.

In addition, this thesis further improves the RZOP method, named Segmented Relevance-Zone-Oriented Proof (SRZOP) search that speeds up the time to solve Connect6 positions. The experimental results in Chapter 5 archive 2.04 speedups to solve the 12 openings. This thesis also demonstrates records of our Connect6 program *NCTU6* in Appendix F, which won the gold in the 11<sup>th</sup> and 13<sup>th</sup> Computer Olympiads in 2006 and 2008, respectively; and also won eight games and lost none against top Connect6 players in Taiwan in 2009. Finally, this thesis applies the RZOP method and SRZOP method into *NCTU6* and *NCTU6-verifiers* which are used in the two systems (described in Subsection 5.1.3 and 5.1.4): (a) desktop grid system (b) JL-PN system. These two systems help us solve many Connect6 openings automatically. The author is very proud to announce this thesis because it is a milestone of *NCTU6* and *NCTU6-verifiers* since year 2005 [65][66].



## References

- [1] L.V. Allis, *Searching for solutions in games and artificial intelligence*, Ph.D. Thesis, University of Limburg, Maastricht, The Netherlands, 1994.
- [2] L.V. Allis, H.J. van den Herik and M.P.H. Huntjens, “Go-Moku Solved by New Search Techniques,” *Computational Intelligence*, vol. 12, pp. 7–23, 1996.
- [3] L.V. Allis, M. van der Meulen and H.J. van den Herik, “Proof-number Search,” *Artificial Intelligence*, vol. 66 (1), pp. 91–124, 1994.
- [4] D.P. Anderson, “Boinc: A System for Public-resource Computing and Storage,” *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, IEEE CS Press, Pittsburgh, USA, pp. 4-10, 2004.
- [5] J. Beck, “On Positional Games,” *Combinatorial Theory Series A 30*, pp. 117–133, 1981.
- [6] C. Berge, *Graphs and Hypergraphs*, North Holland, Amsterdam, 1973.
- [7] E.R. Berlekamp, J.H. Conway and R.K. Guy, *Winning Ways for your Mathematical Plays*, vol. 3, 2nd ed., A K Peters. Ltd. Canada, 2003.
- [8] BOINC, Available: <http://boinc.berkeley.edu/>.
- [9] D.M. Breuker, J. Uiterwijk and H. J. van den Herik, “The PN2-search Algorithm,” in H. J. van den Herik, B. Monien (Eds.), *Advances in Computer Games*, vol. 9, IKAT, Universiteit Maastricht, Maastricht, The Netherlands, pp. 115–132, 2001.
- [10] A. de Bruin, W. Pijls and A. Plaat, “Solution Trees as a Basis for Game-Tree Search,” *ICCA Journal*, vol 17(4), pp. 207–219, December 1994.
- [11] T. Cazenave, “Abstract Proof Search,” *Computers and Games* (eds. T. A. Marsland and I. Frank), *Lecture Notes in Computer Science*, vol. 2063, pp. 39–54, 2001.
- [12] T. Cazenave, “A Generalized Threats Search Algorithm,” *Computers and Games, Lecture Notes in Computer Science*, vol. 2883, pp. 75–87, 2003.
- [13] G.M. Chaslot, M.H.M. Winands and H.J. van den Herik, “Parallel Monte-Carlo Tree Search,” *International Conference on Computers and Games (CG2008)*, Beijing, China, 2008.

- [14] C.-P. Chen, I.-C. Wu and Y.-C. Chan, “ConnectLib – A Connect6 Editor,” Available: [http://www.connect6.org/Connect6Lib\\_Manual.htm](http://www.connect6.org/Connect6Lib_Manual.htm), 2009.
- [15] S.-H. Chiang, I.-C. Wu and P.-H. Lin, “On Draw K-in-a-row Games,” *Advances in Computer Games Conference (ACG2009)*, vol. 6048, pp. 158–169, 2010.
- [16] Chinese Association for Artificial Intelligence, Chinese Computer Games Contest (in Chinese), Available: <http://www.caii.cn/>.
- [17] L. Csirmaz, “On a Combinatorial Game with An Application to Go-moku,” *Discrete Math.* 29, pp. 19–23, 1980.
- [18] R. Diestel, *Graph Theory*, Springer, New York, 2nd edition, 2000.
- [19] G. Fedak, C. Germain, V. Neri and F. Cappello, “Xtremweb: A Generic Global Computing System,” *Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID2001): Workshop on Global Computing on Personal Devices*, IEEE CS Press, Brisbane, Australia, pp. 582–587, 2001.
- [20] I. Foster, C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, Inc., 1999.
- [21] The globus project, Available: <http://www.globus.org/>.
- [22] H.J. van den Herik, J.W.H.M. Uiterwijk and J.V. Rijswijk, “Games solved: Now and in the future,” *Artificial Intelligence*, vol. 134 (1-2), pp. 277–311, 2002.
- [23] H.J. van den Herik and M.H.M. Winands, “Proof-Number Search and its Variants,” *Oppositional Concepts in Computational Intelligence*, pp. 91-118, 2008.
- [24] M.-Y. Hsieh and S.-C. Tsai, “On the Fairness and Complexity of Generalized K-in-a-row Games,” *Theoretical Computer Science*, vol. 385, pp. 88–100, 2007.
- [25] Yu-Chun Huang, *private communication*, 2008.
- [26] B. Jacob, L. Ferreira, N. Bieberstein, C. Gilzean, J.Y. Girard, R. Strachowski and S.S. Yu, *Enabling Applications for Grid Computing with Globus*, IBM Redbooks, 2003.
- [27] A. Kishimoto and Y. Kotani, “Parallel AND/OR Tree Search Based on Proof and Disproof Numbers,” *Fifth Games Programming Workshop*, vol. 99(14) of IPSJ Symposium Series, pp. 24–30, 1999.
- [28] A. Kishimoto and M. Müller, “A general solution to the graph history interaction problem,” *Nineteenth National Conference on Artificial Intelligence (AAAI2004)*, pp.

644–649, San Jose, CA, 2004.

- [29] A. Kishimoto and M. Müller, “Search versus Knowledge for Solving Life and Death Problems in Go,” *Twentieth National Conference on Artificial Intelligence (AAAI2005)*, pp. 1374–1379, 2005.
- [30] P.-H. Lin and I.-C. Wu, “NCTU6 Wins Man-Machine Connect6 Championship 2009,” *ICGA Journal*, vol. 32(4), pp. 230–232, 2009.
- [31] P.-H. Lin and I.-C. Wu, “Segmented Relevance-Zone-Oriented Proof Search for Connect6,” *in preparation*, 2010.
- [32] T.W. Lee, One of Early Tsumegos for Connect6, Available: [http://www.connect6.org/web/index.php?option=com\\_tsumego&task=loadTsumegoHistoryList&class\\_id=32](http://www.connect6.org/web/index.php?option=com_tsumego&task=loadTsumegoHistoryList&class_id=32), 2005.
- [33] Littlegolem, Online Connect6 games, Available: <http://www.littlegolem.net/>, 2006.
- [34] V. Manohararajah, *Parallel Alpha-beta Search on Shared Memory Multiprocessors*, Master’s thesis, Graduate Department of Electrical and Computer Engineering, University of Toronto, Canada, 2001.
- [35] A. Nagai, *Df-pn Algorithm for Searching AND/OR Trees and Its Applications*, Ph.D. thesis, University of Tokyo, Japan, 2002.
- [36] J. Pawlewicz and L. Lew, “Improving Depth-first pn-search:  $1+\epsilon$  Trick,” In H. J. van den Herik, P. Ciancarini, and H.H.L.M. Donkers, editors, *Fifth International Conference on Computers and Games*, vol. 4630 of LNCS, pp. 160–170, Computers and Games, Springer, Heidelberg, 2006.
- [37] W. Pijls and A. de Bruin, “Game Tree Algorithms and Solution Trees,” *Computers and Games, Lecture Notes in Computer Science*, vol. 1558, pp. 195–204, 1999.
- [38] A. Pluhar, “The Accelerated K-in-a-row Game,” *Theoretical Computer Science*, vol. 270(1-2), pp. 865–875, 2002.
- [39] V. N. Rao and V. Kumar, “Superlinear Speedup in State-space Search,” *Proceedings of the 1988 Foundation of Software Technology and Theoretical Computer Science*, no. 338 of LNCS, pp. 161–174, Springer-Verlag, 1988.
- [40] Red-bean.com, SGF File Format, Available: <http://www.red-bean.com/sgf/>.
- [41] Renju International Federation, The International Rules of Renju, Available:



<http://www.renju.net/study/rifrules.php>, 1998.

- [42] Renlib, Renju – A Ranju Editor, Available: <http://www.renju.se/renlib/>.
- [43] J.T. Saito, M.H.M. Winands and H.J. van den Herik, “Randomized Parallel Proof-Number Search,” *Advances in Computer Games Conference (ACG2009), Lecture Notes in Computer Science (LNCS 6048)*, pp. 75–87, Palacio del Condestable, Pamplona, Spain, 2009.
- [44] G. Sakata and W. Ikawa, *Five-In-A-Row, Renju*. The Ishi Press, Inc., Tokyo, Japan, 1981.
- [45] J. Schaeffer, N. Burch, Y.N. Björnsson, A. Kishimoto, M. Müller, R. Lake, P. Lu and S. Sutphen, “Checkers is Solved,” *Science*, vol. 5844(317), pp. 1518–1552, 2007.
- [46] M. Seo, H. Iida and J. Uiterwijk, “The PN\*-search algorithm: Application to Tsumeshogi,” *Artificial Intelligence*, vol. 129(1-2), pp. 253–277, 2001.
- [47] SETI@home, Available: <http://setiathome.ssl.berkeley.edu>.
- [48] S. Soeda, T. Kaneko and T. Tanaka, “Dual Lambda Search and its Application to Shogi Endgames,” *Advances in Computer Games Conference (ACG2005)*, Taipei, Taiwan, 2005.
- [49] S. Soeda, T. Kaneko and T. Tanaka, “Dual Lambda Search and Shogi Endgames,” *Advances in Computer Games Conference (ACG'11)*, Lecture Notes in Computer Science, vol. 4250, pp. 126–139, 2006.
- [50] Taiwan Connect6 Association, Connect6 homepage, Available: <http://www.connect6.org/>.
- [51] ThinkNewIdea Inc, CYC game (in Chinese), Available: <http://cycgame.com/>, 2005.
- [52] T. Thomsen, “Lambda-Search in Game Trees - With Application to Go,” *ICGA Journal*, vol. 23(4), pp. 203–217, 2000.
- [53] J. Wagner and I. Virag, “Solving Renju,” *ICGA Journal*, vol. 24(1), pp. 30–34, 2001.
- [54] M.H.M. Winands, J.W.H.M. Uiterwijk and H.J. van den Herik, “PDS-PN: A new proof-number search algorithm: Application to Lines of Action,” In J. Schaeffer, M. Müller, and Y. Björnson, editors, *Computers and Games 2002*, vol. 2883 of LNCS, pp. 170–185. Computers and Games, Springer, Heidelberg, 2003.
- [55] I.-C. Wu, Proposal for a New Computer Olympiad Game – Connect6, Available: <http://ticc.uvt.nl/icga/news/Olympiad/Olympiad2006/connect6.pdf>, or <http://www.connect6.org/articles/RZOP/connect6.pdf>, 2005.

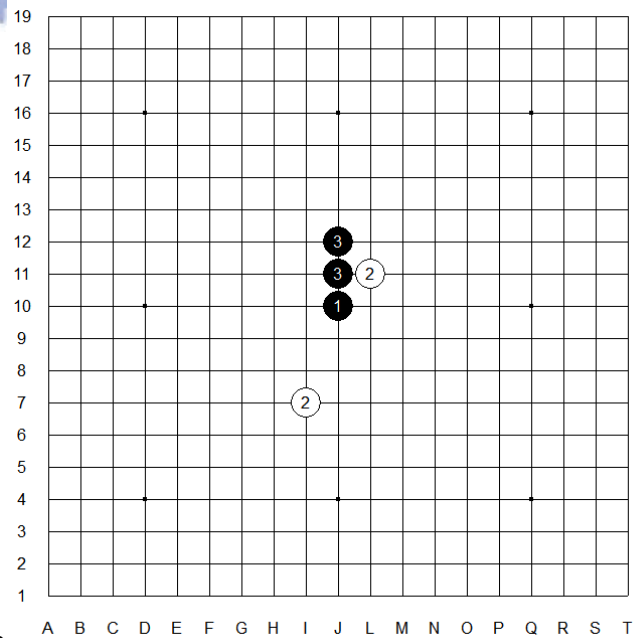
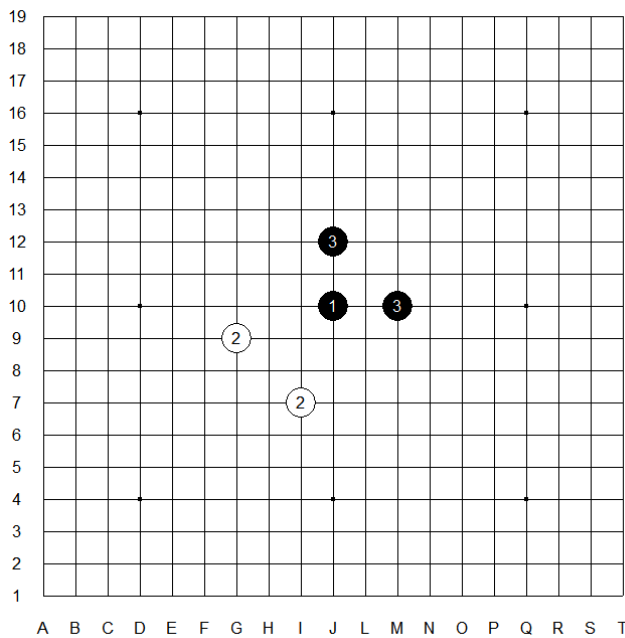
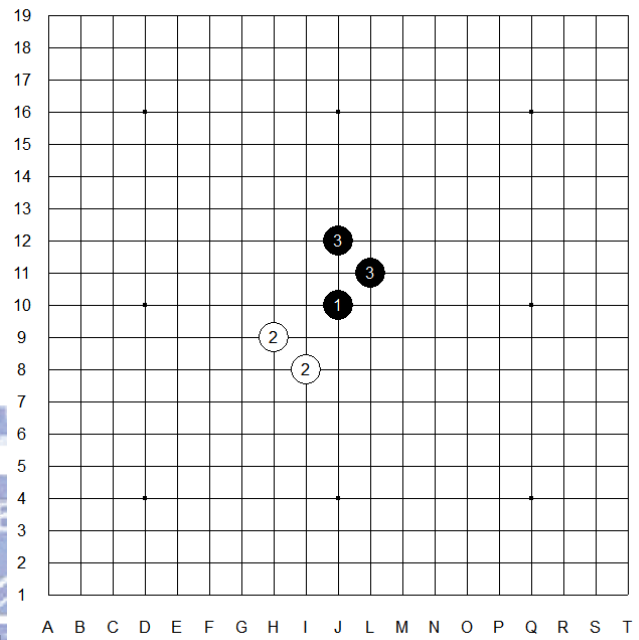
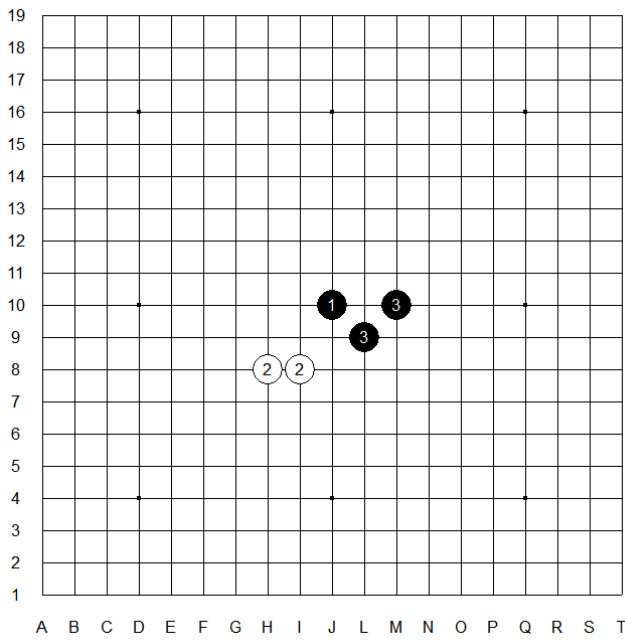
- [56] I.-C. Wu, B.-H. Lin, L.-B. Chen, J.-Y. Su and P.-C. Hsu, "HybridDiff: An Algorithm for A New Tree Editing Distance Problem," *International Computer Symposium (ICS2006)*, Taipei, Taiwan, 2006.
- [57] I.-C. Wu, C.-P. Chen, P.-H. Lin, K.-C. Huang, L.-P. Chen, D.-J. Sun, Y.-C. Chan and H.-Y. Tsou, "A Volunteer-Computing-Based Grid Environment for Connect6 Applications," *IEEE International Conference on Computational Science and Engineering (CSE2009)*, vol. 1, pp. 110–117, 2009.
- [58] I.-C. Wu, C.-P. Chen, P.-H. Lin, G.-Z. Huang, L.-P. Chen, D.-J. Sun and H.-Y. Tsou, "A Desktop Grid Computing Service for Connect6 Applications," *International Symposium on Grid Computing (ISGC2009)*, Taipei, Taiwan, 2009.
- [59] I.-C. Wu and P.-H. Lin, "NCTU6-Lite Wins Connect6 Tournament," *ICGA Journal*, vol. 31(4), pp. 240–243, 2008.
- [60] I.-C. Wu and P.-H. Lin, "Relevance-Zone-Oriented Proof Search for Connect6," *IEEE Transaction on Computational Intelligence and AI in Games*, vol. 2(3), September 2010.
- [61] I.-C. Wu and P.-H. Lin, Benchmark for RZOP search, Available: <http://www.connect6.org/articles/RZOP/>.
- [62] I.-C. Wu and P.-H. Lin, Benchmark for SRZOP search, Available: <http://www.connect6.org/articles/SRZOP/>.
- [63] I.-C. Wu and P.-H. Lin, Search tree for drawn Connect(11,2), Available: <http://www.connect6.org/articles/drawn-connect-games/>.
- [64] I.-C. Wu, H.-H. Lin, P.-H. Lin, D.-J. Sun, Y.-C. Chan and B.-T. Chen, "Job-Level Proof-Number Search for Connect6," *International Conference on Computers and Games (CG2010)*, Kanazawa, Japan, 2010.
- [65] I.-C. Wu, D.-Y. Huang and H.-C. Chang, "Connect6," *ICGA Journal*, vol. 28(4), pp. 234–242, 2006.
- [66] I.-C. Wu and D.-Y. Huang, "A New Family of K-in-a-row Games," *Advances in Computer Games Conference (ACG2005)*, Taipei, Taiwan, 2005.
- [67] I.-C. Wu and S.-J. Yen, "NCTU6 Wins Connect6 Tournament," *ICGA Journal*, vol. 29(3), pp. 157–158, September 2006.
- [68] XtremWeb, Available: <http://www.xtremweb.net/>.

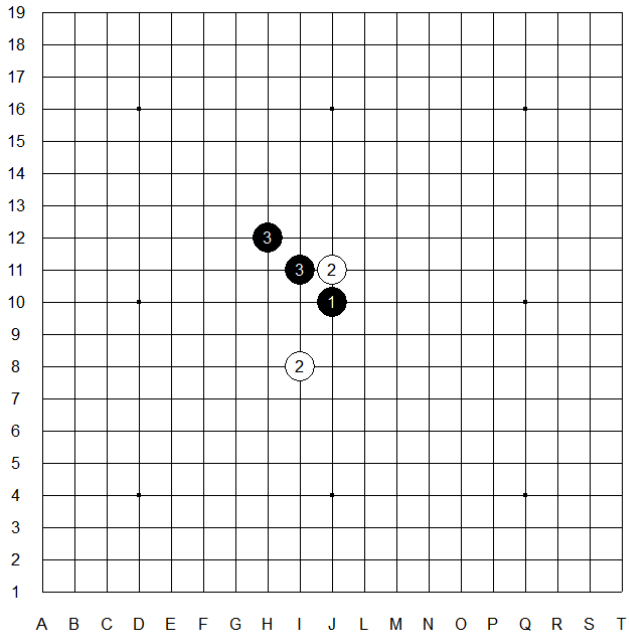
- [69] T. G. L. Zetters, “8(or more) In a Row,” *American Mathematical Monthly* 87, pp. 575–576, 1980.



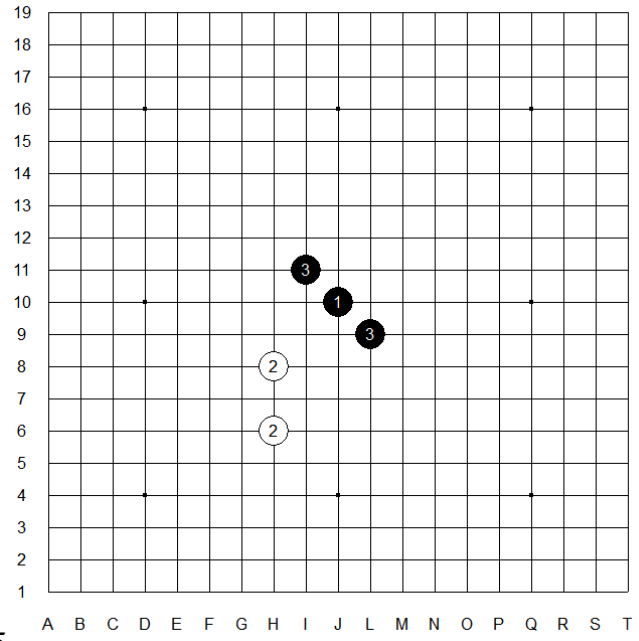
# Appendix A Sample Positions

Figure 34. 65 winning positions.

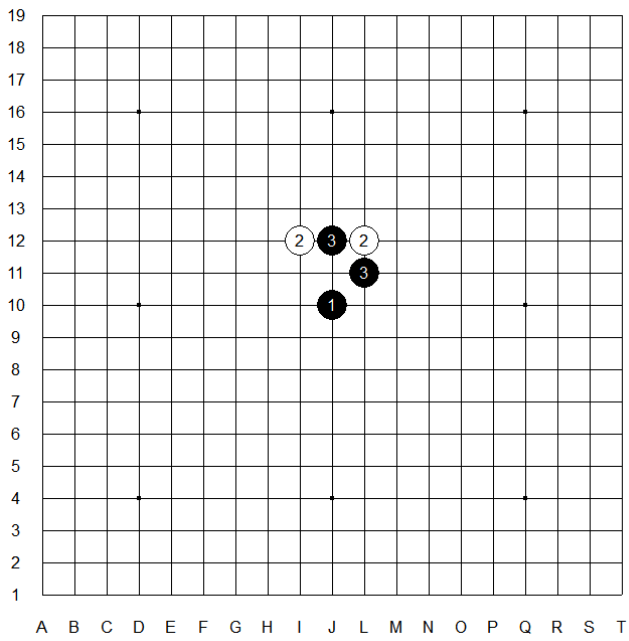




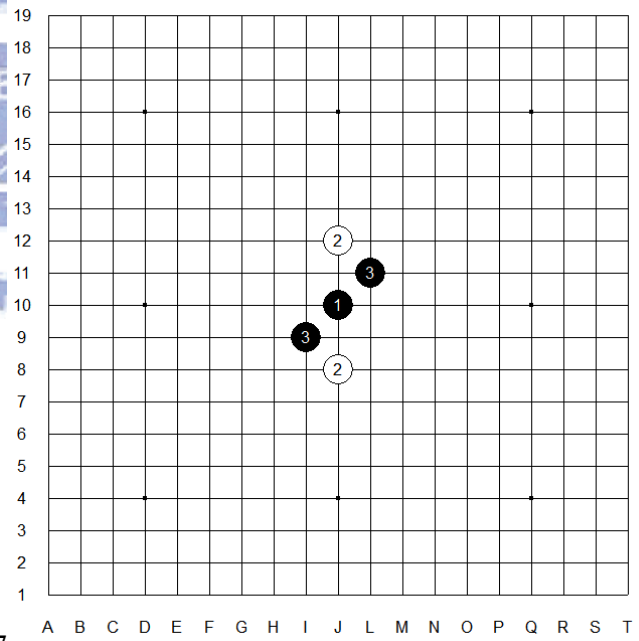
05



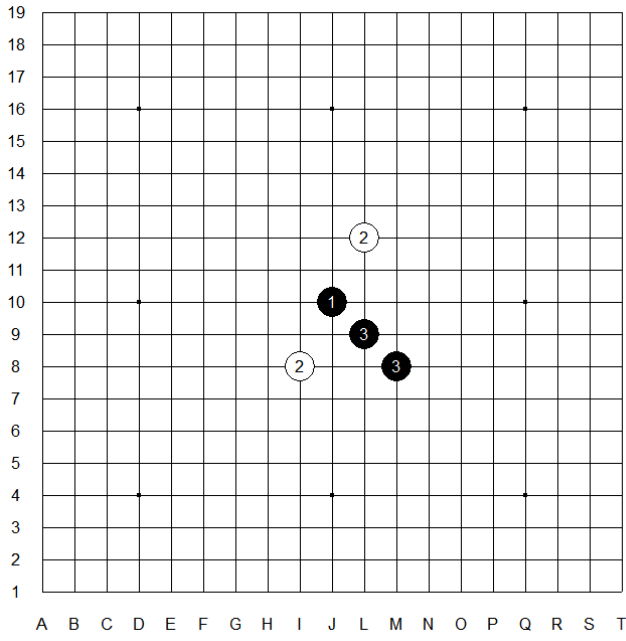
06



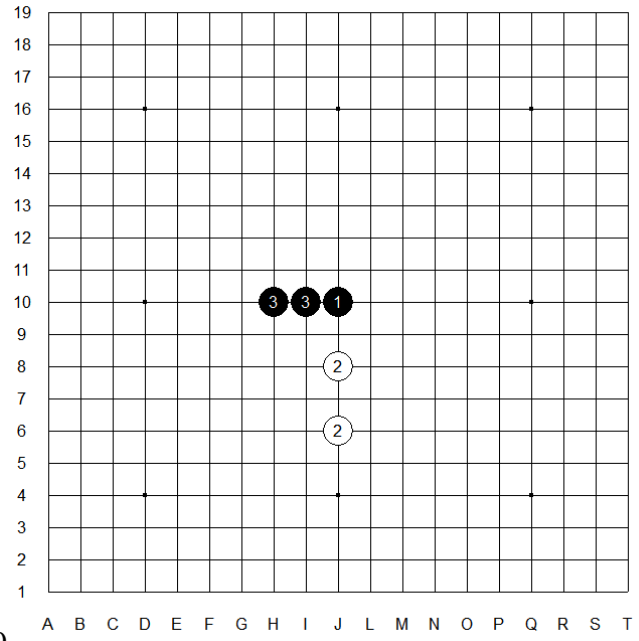
07



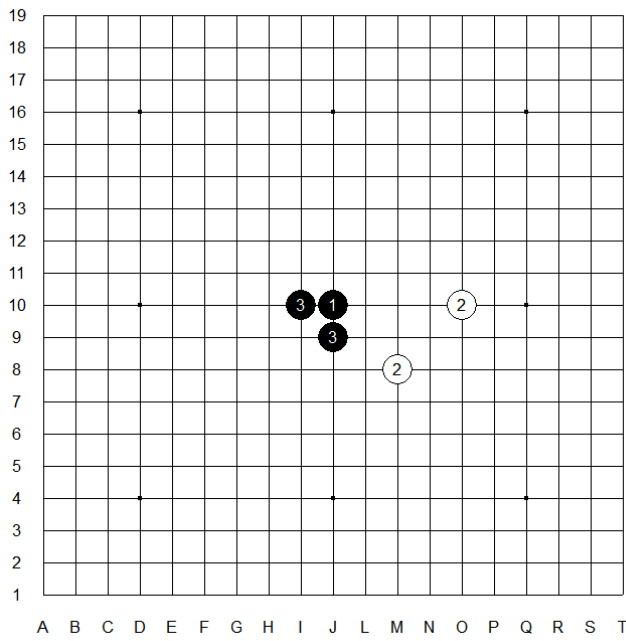
08



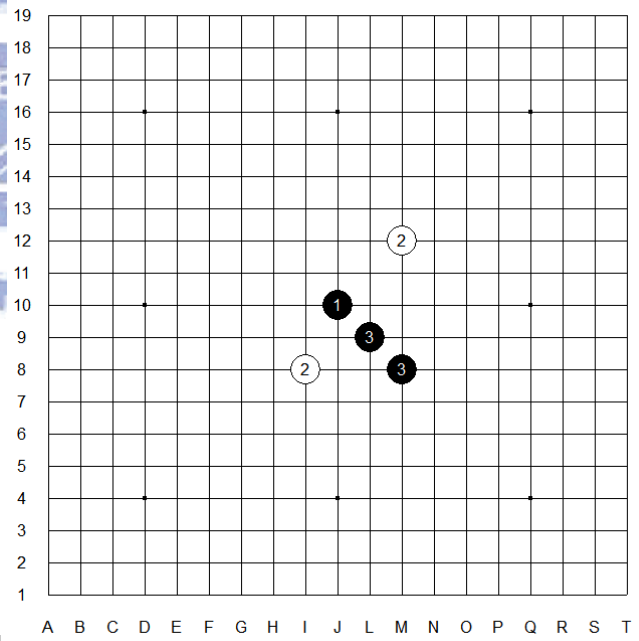
09



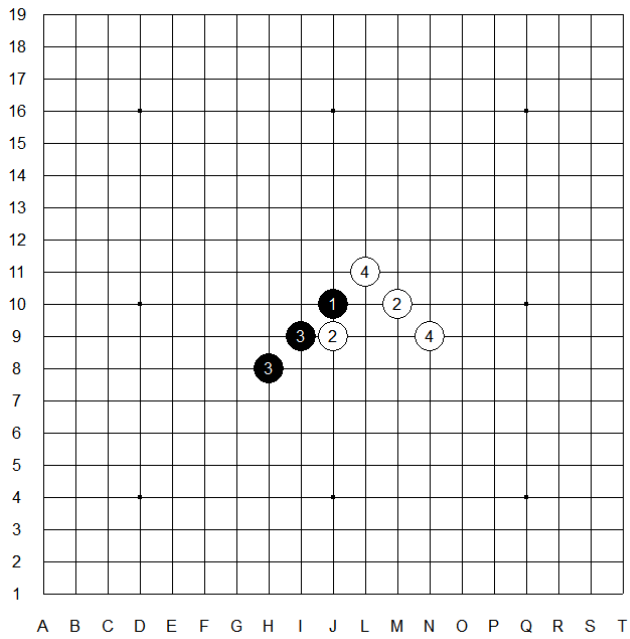
10



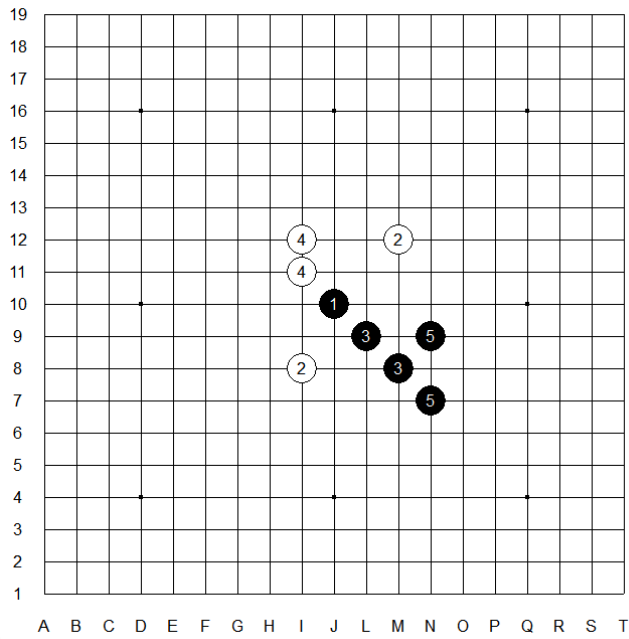
11



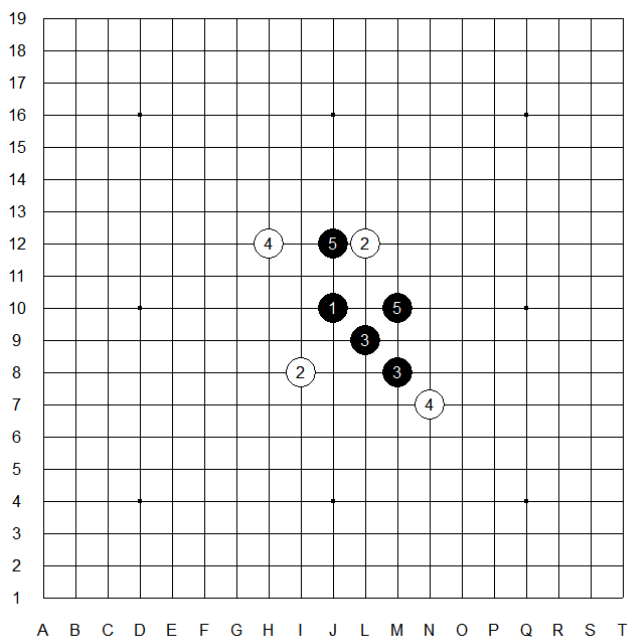
12



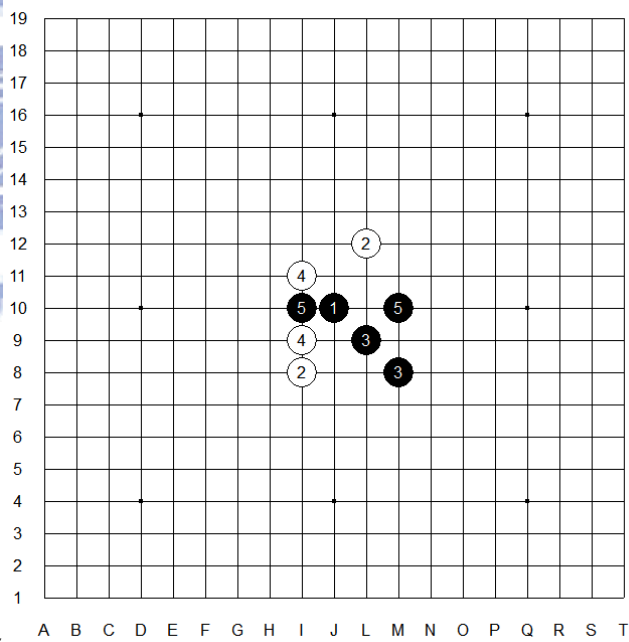
13



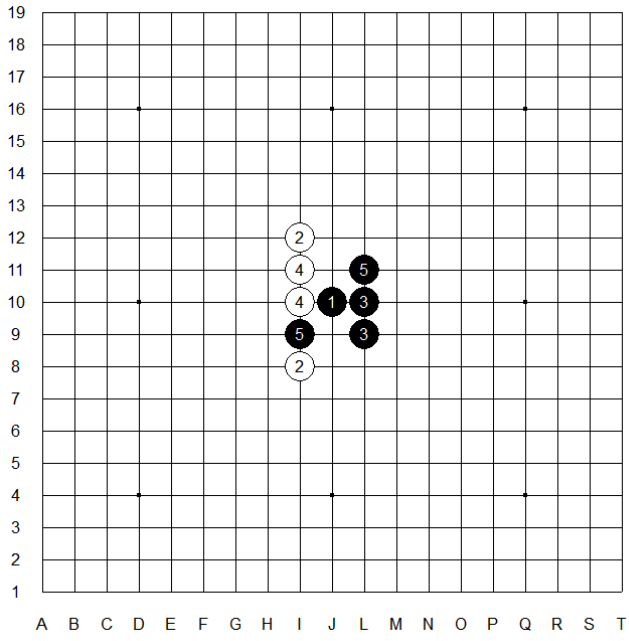
14



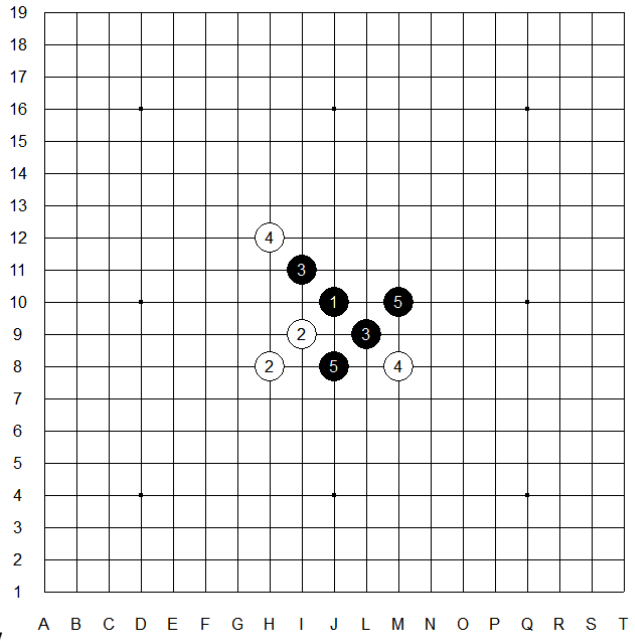
15



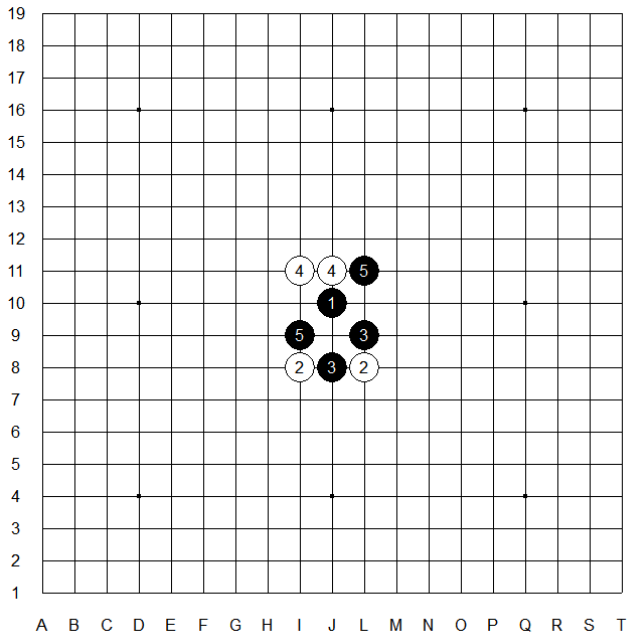
16



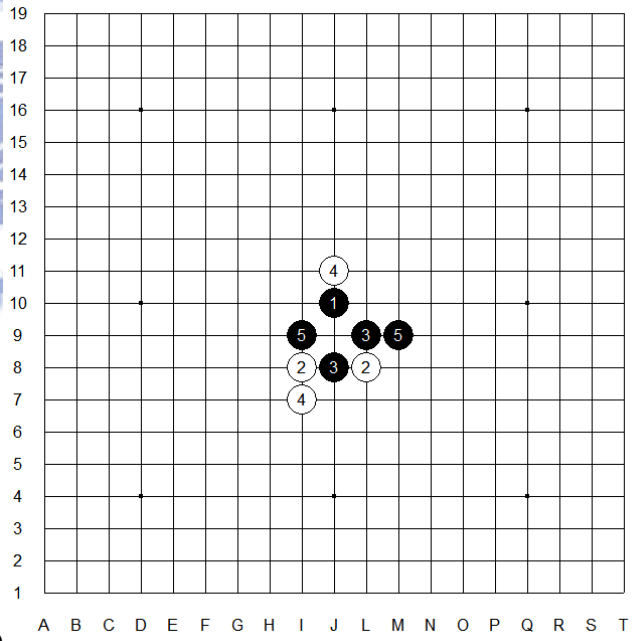
17



18

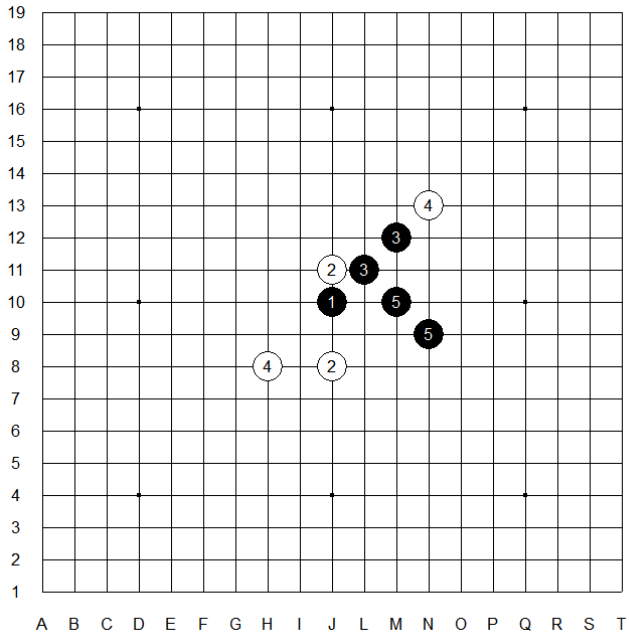


19

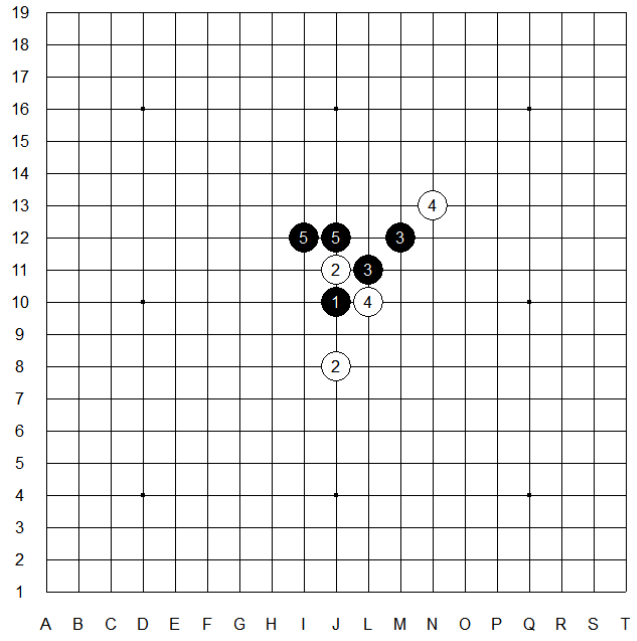


20

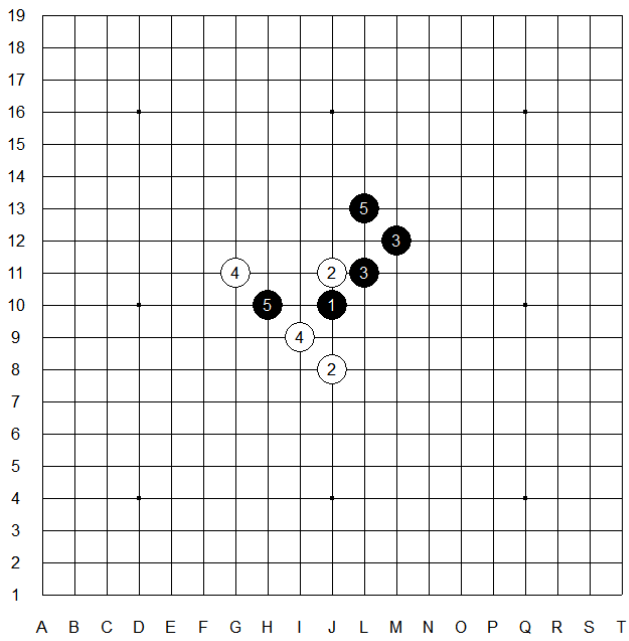




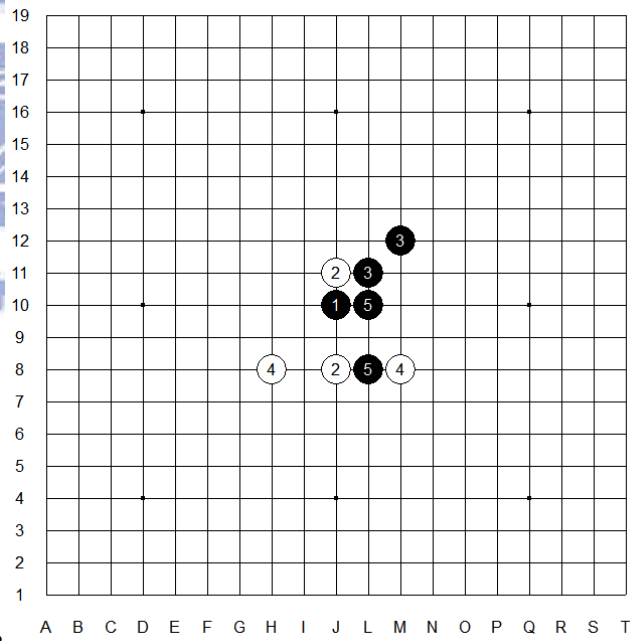
21



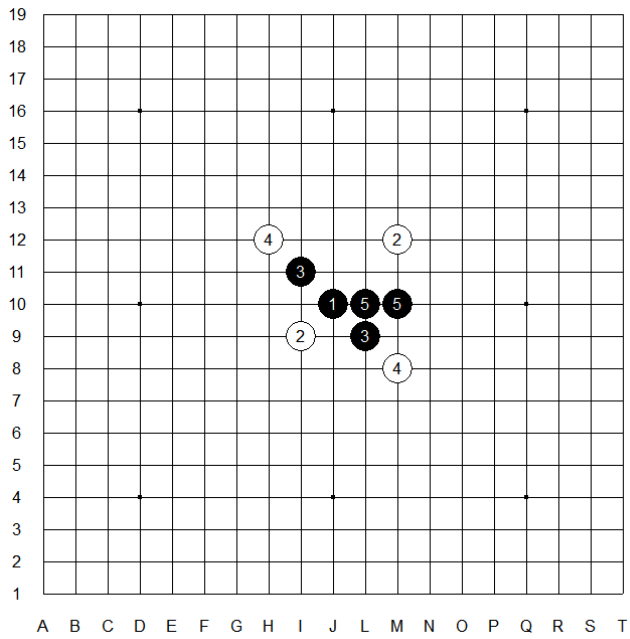
22



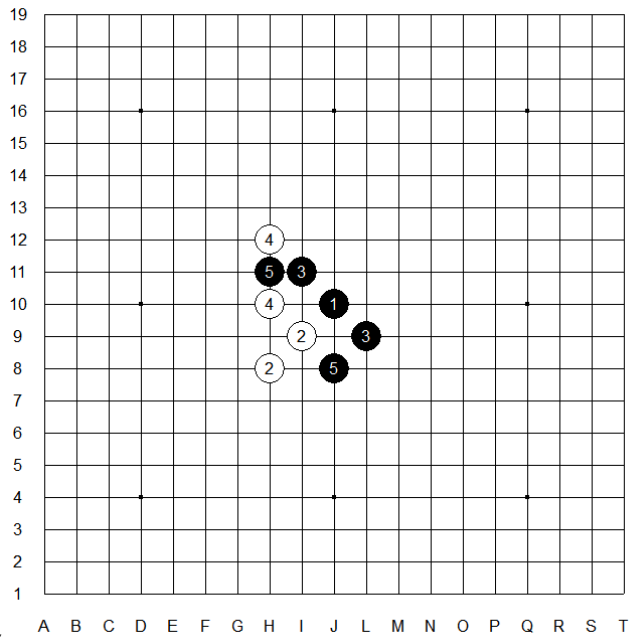
23



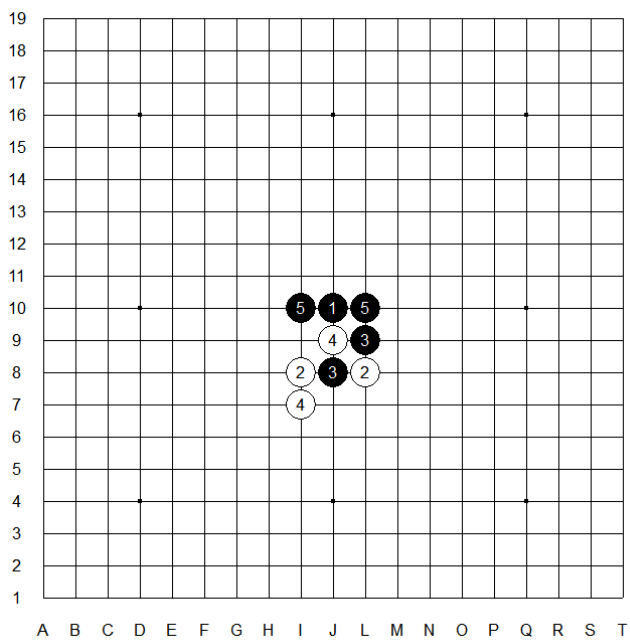
24



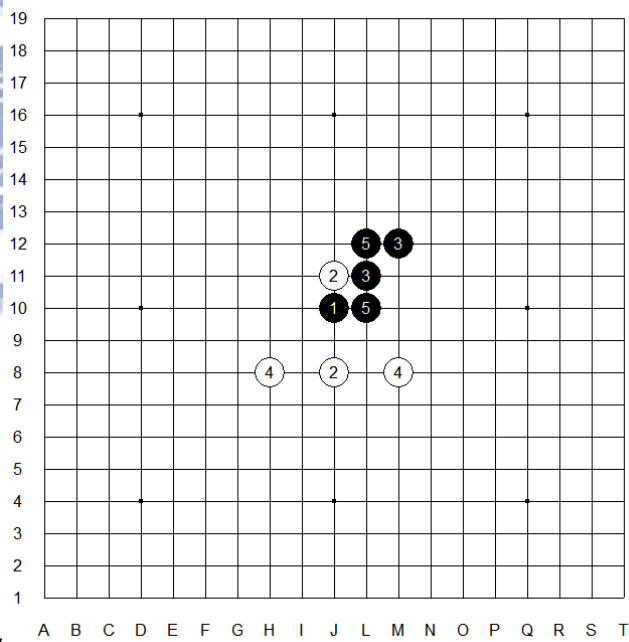
25



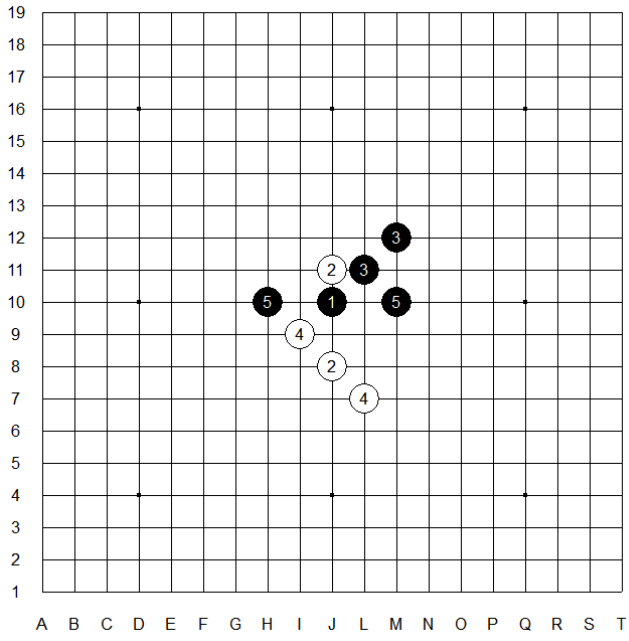
26



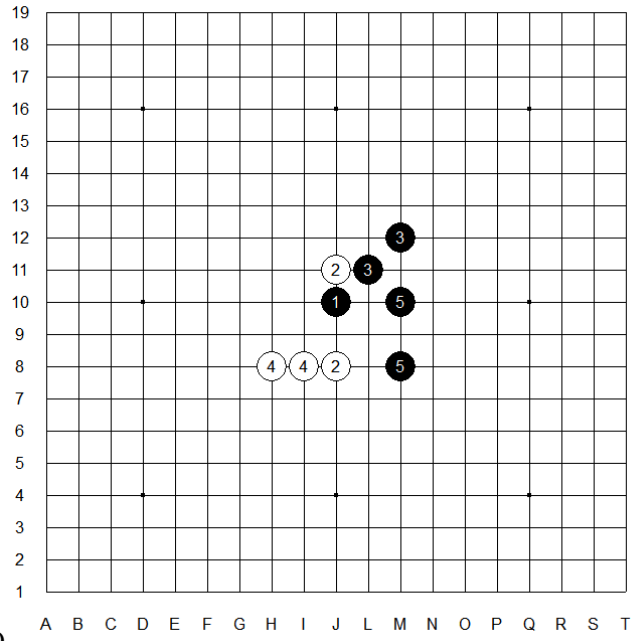
27



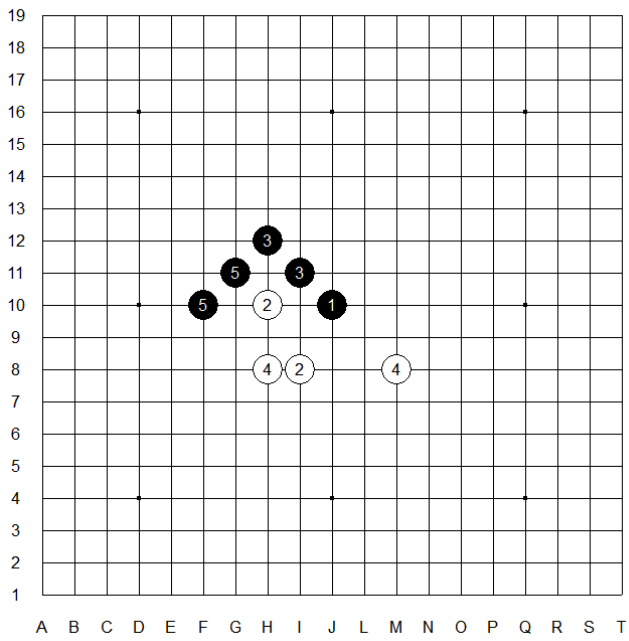
28



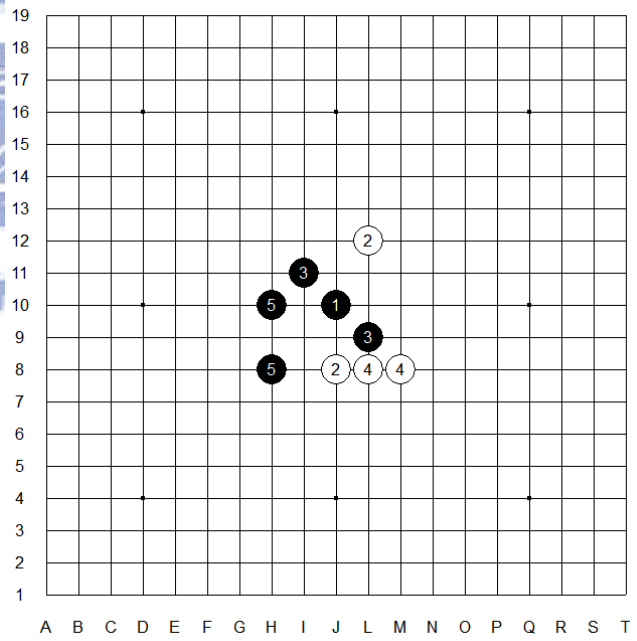
29



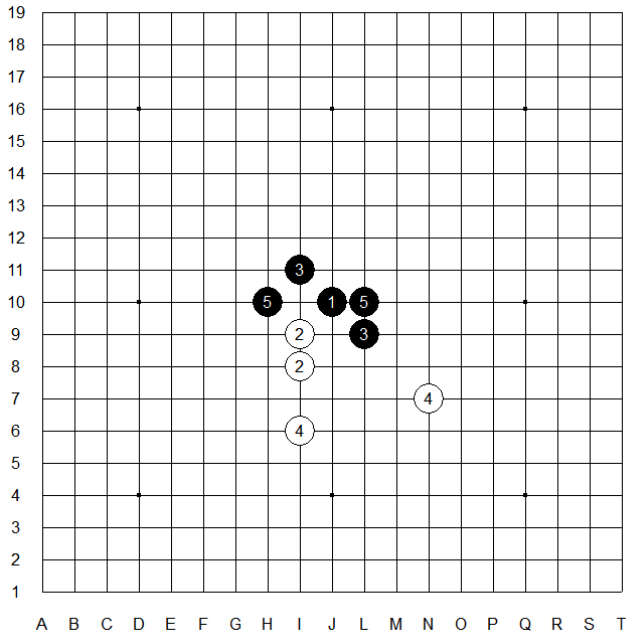
30



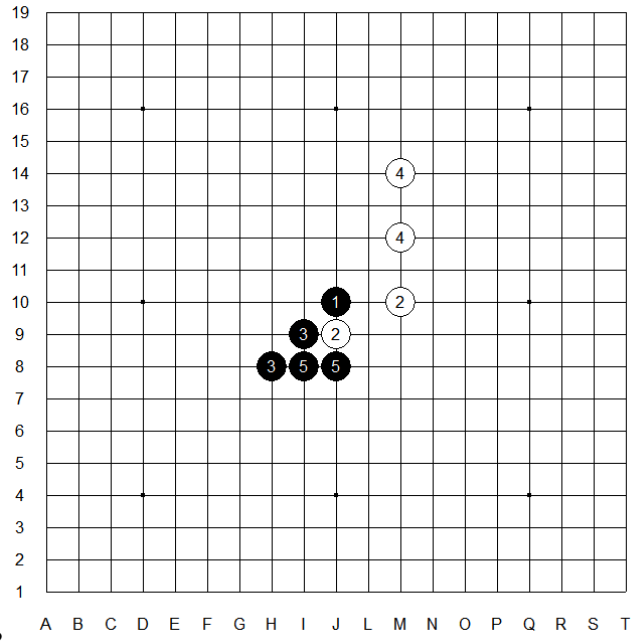
31



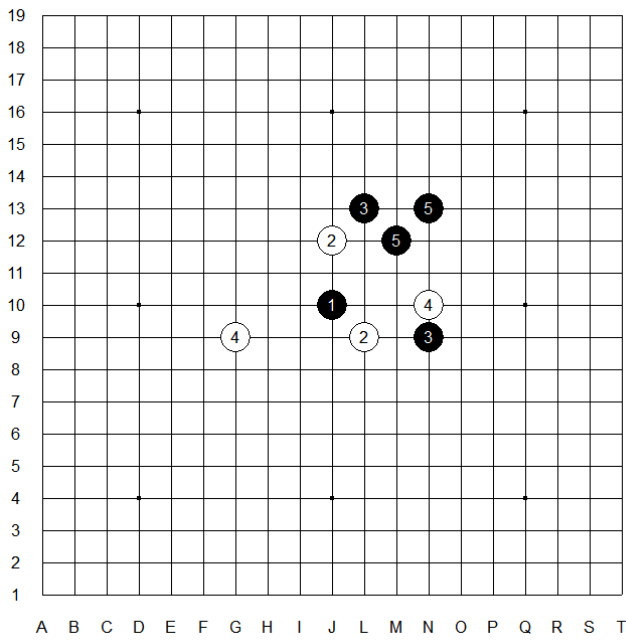
32



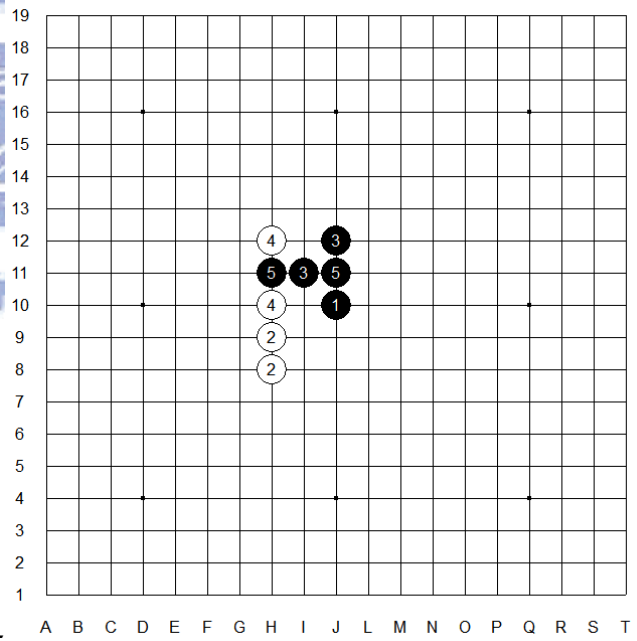
33



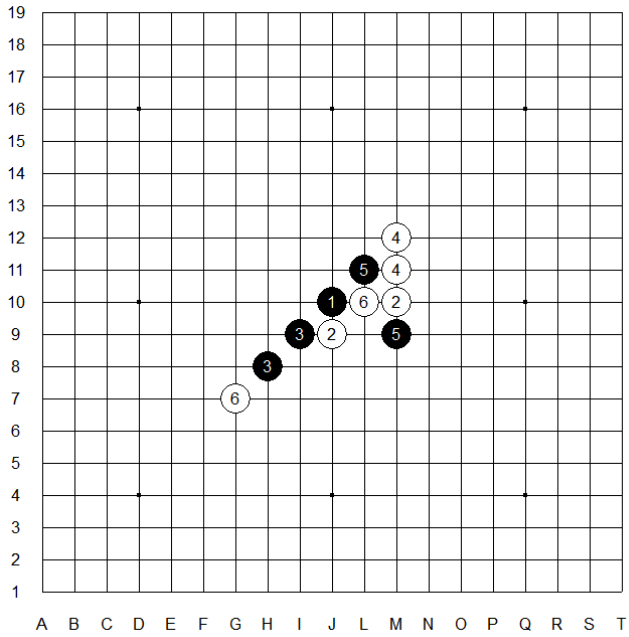
34



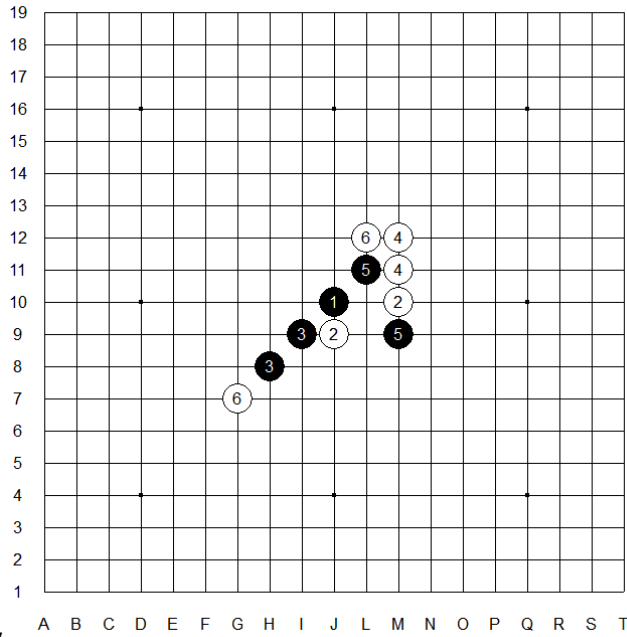
35



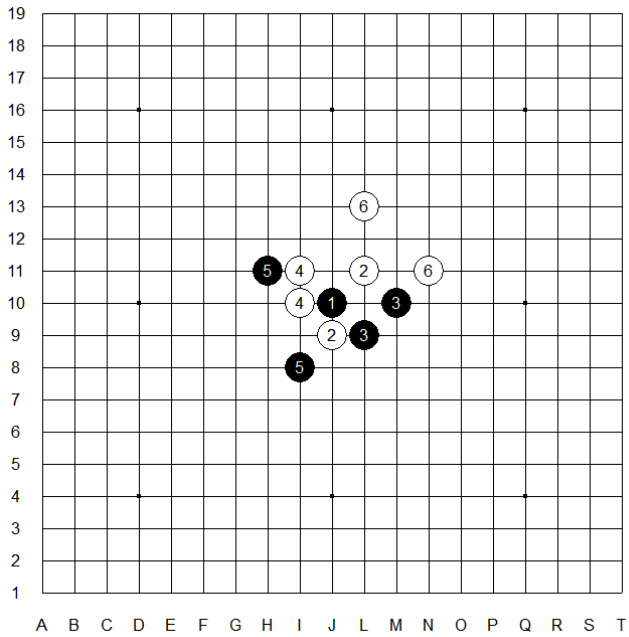
36



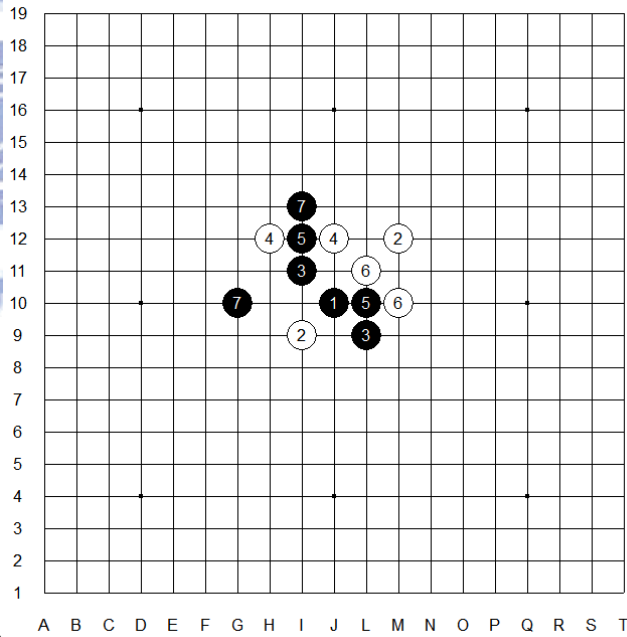
37



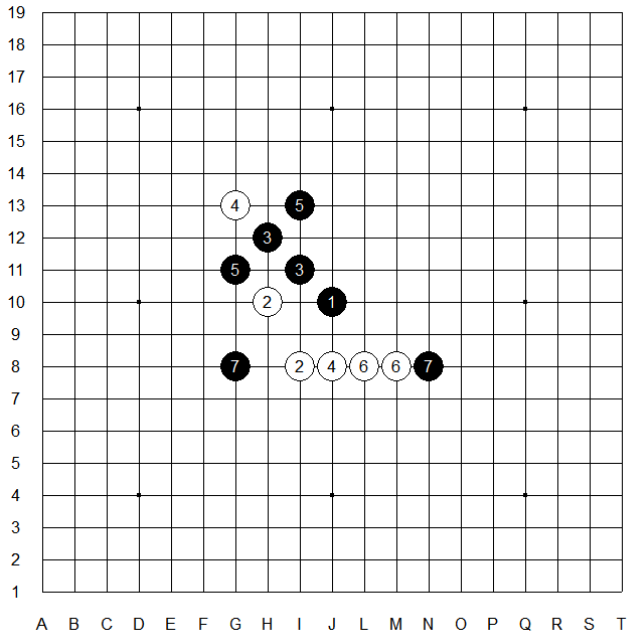
38



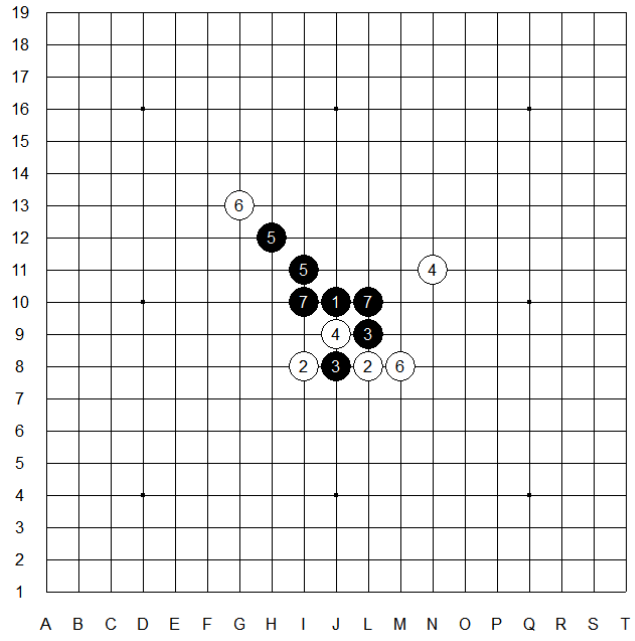
39



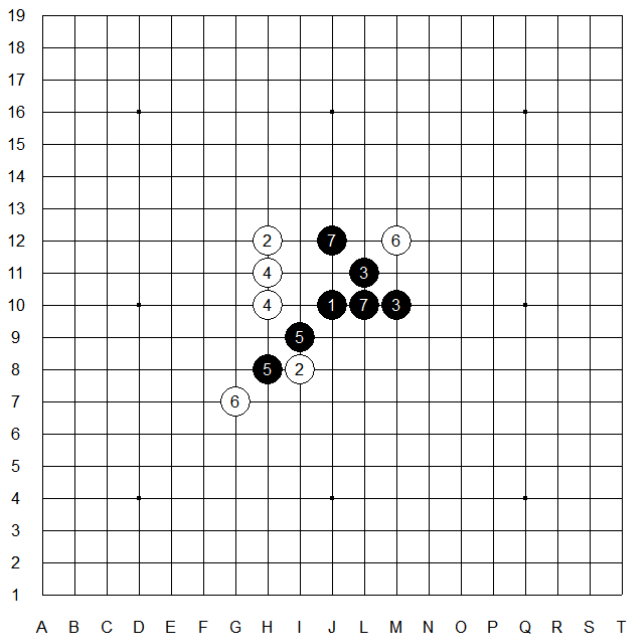
40



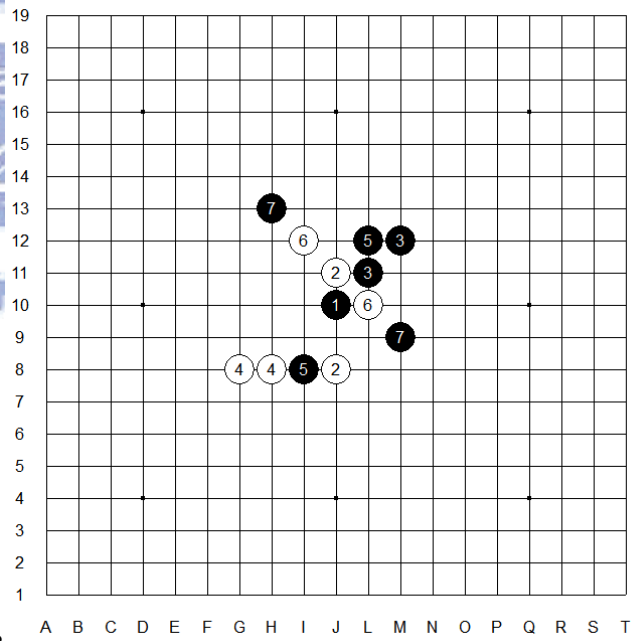
41



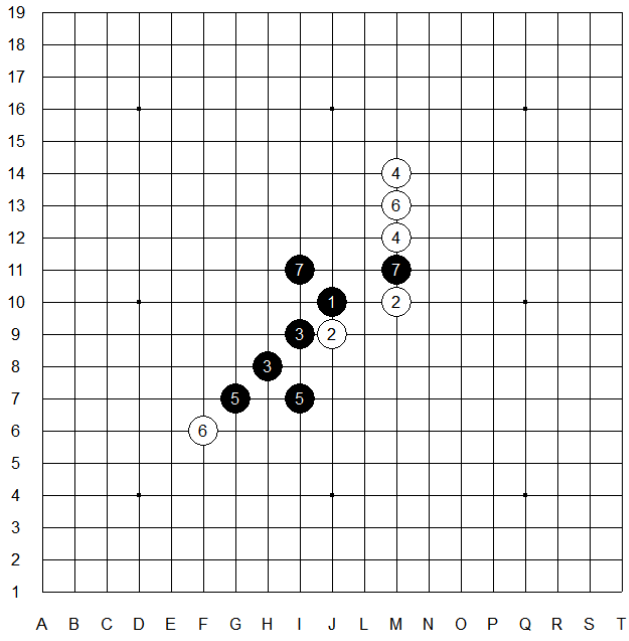
42



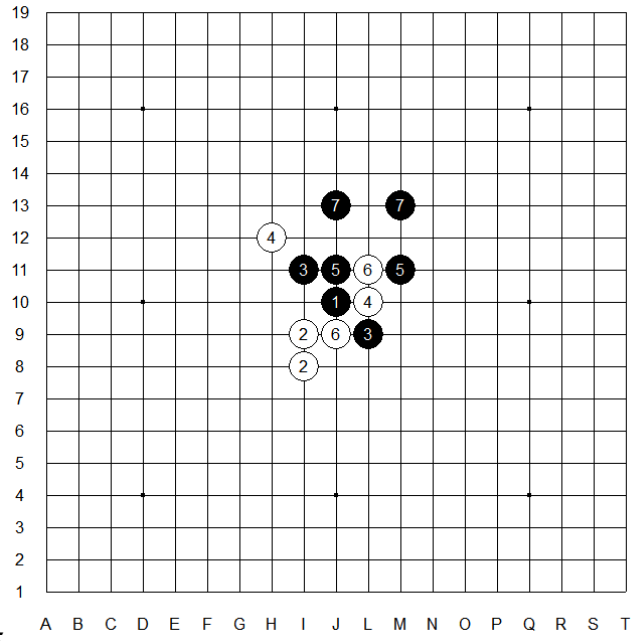
43



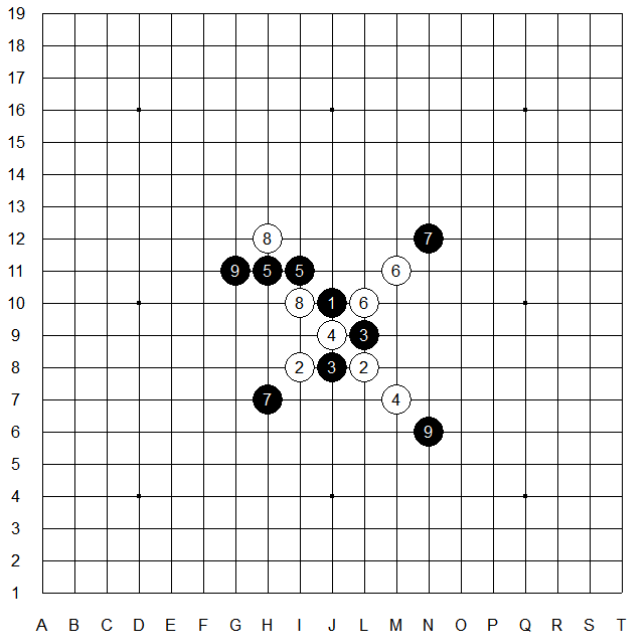
44



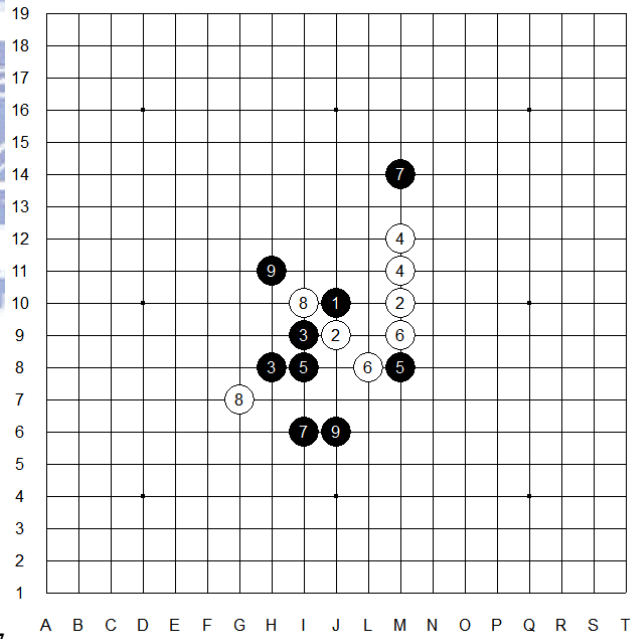
45



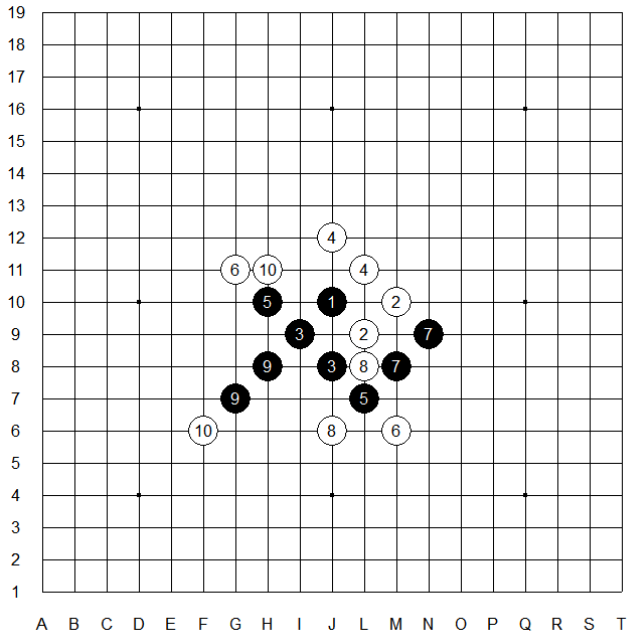
46



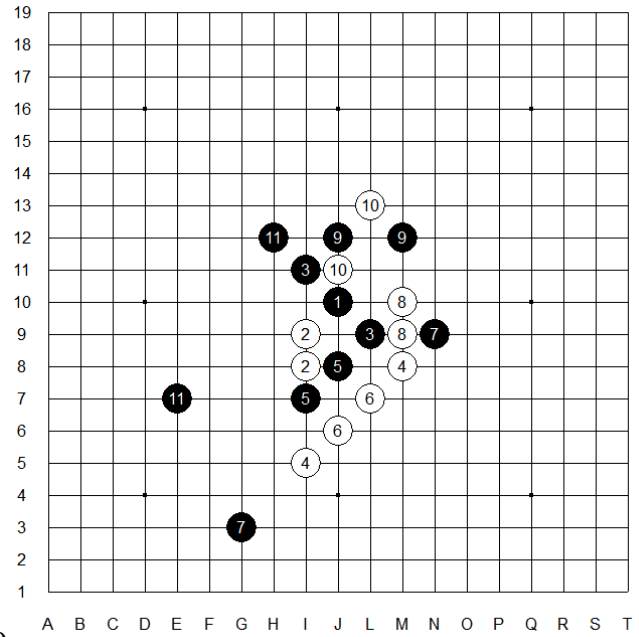
47



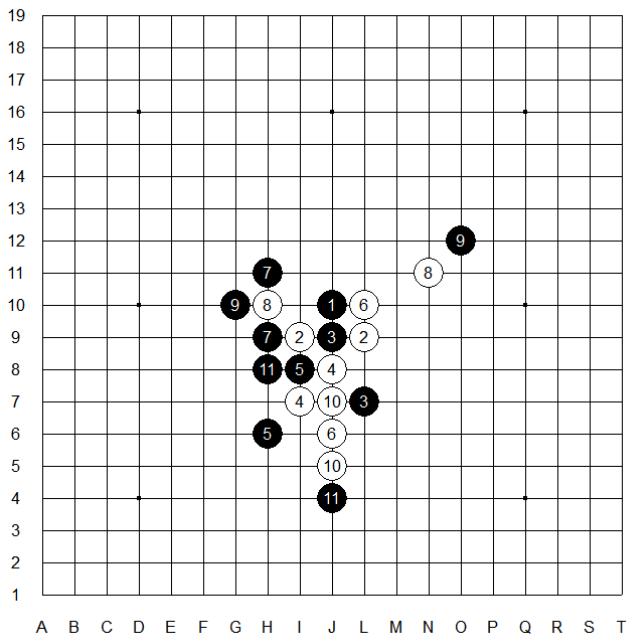
48



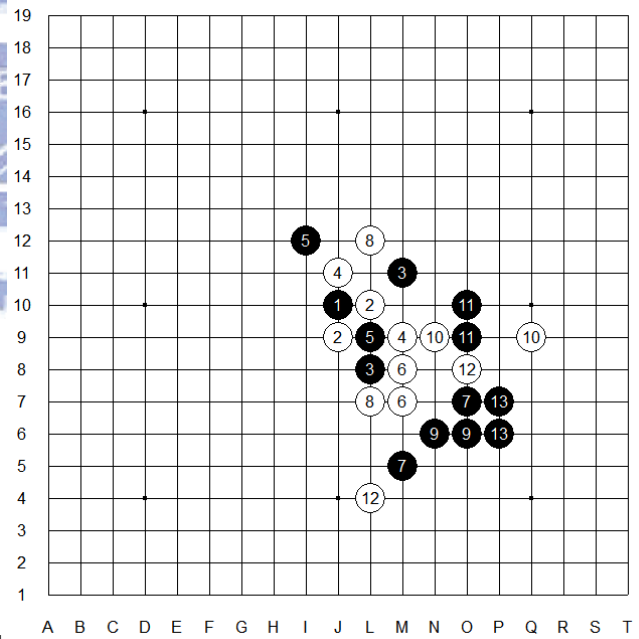
49



50

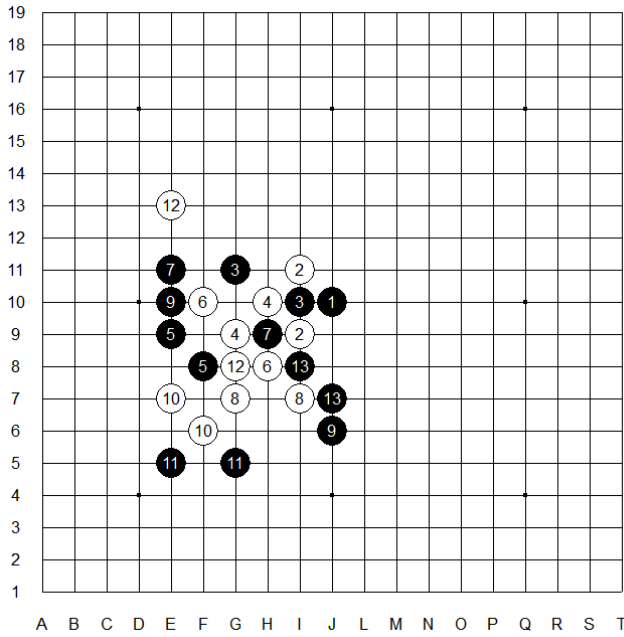


51

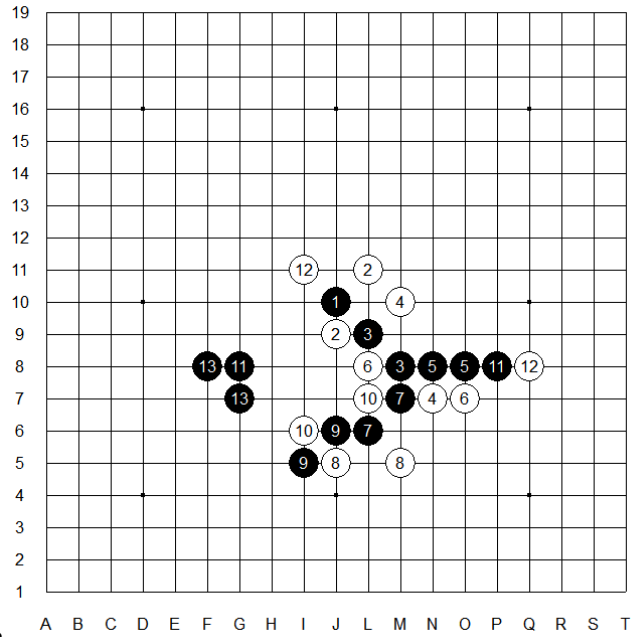


52

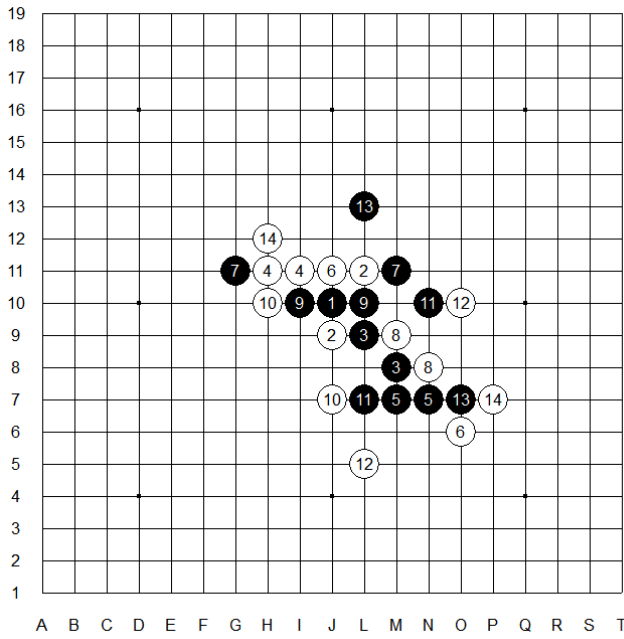




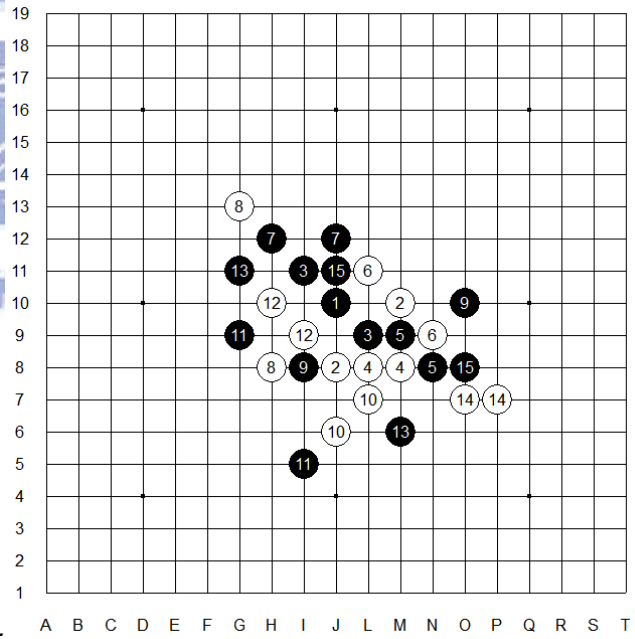
53



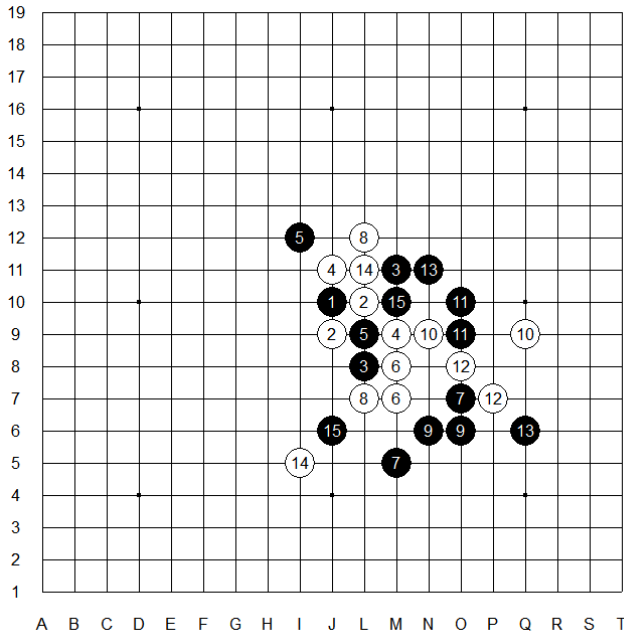
54



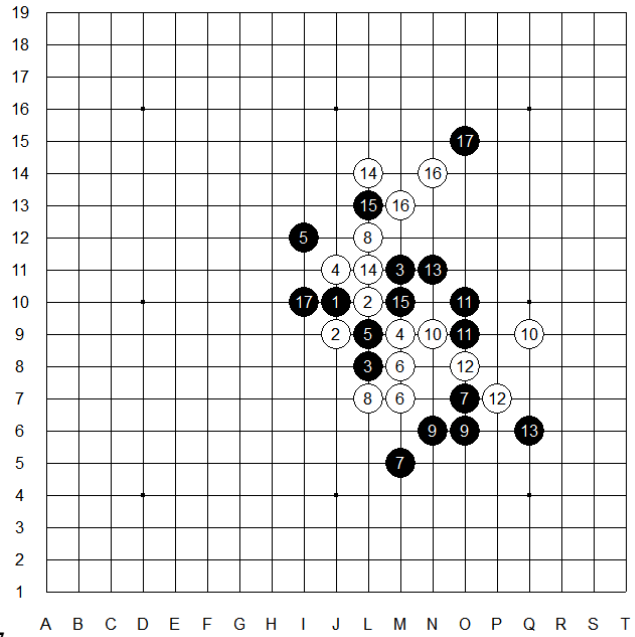
55



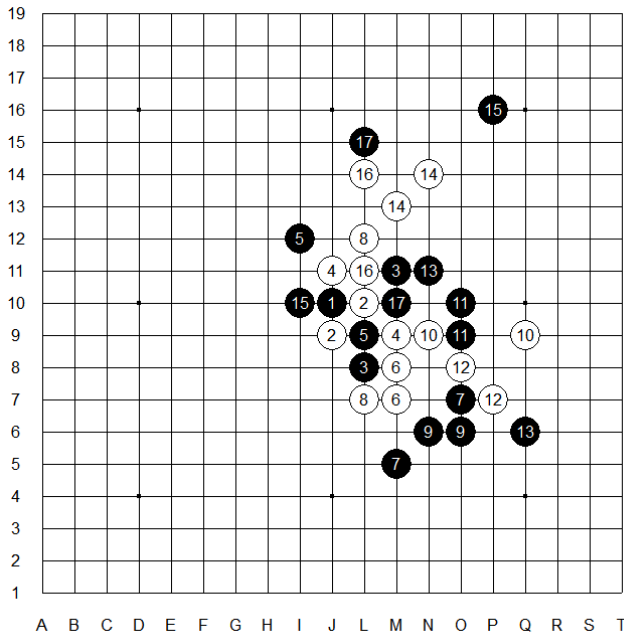
56



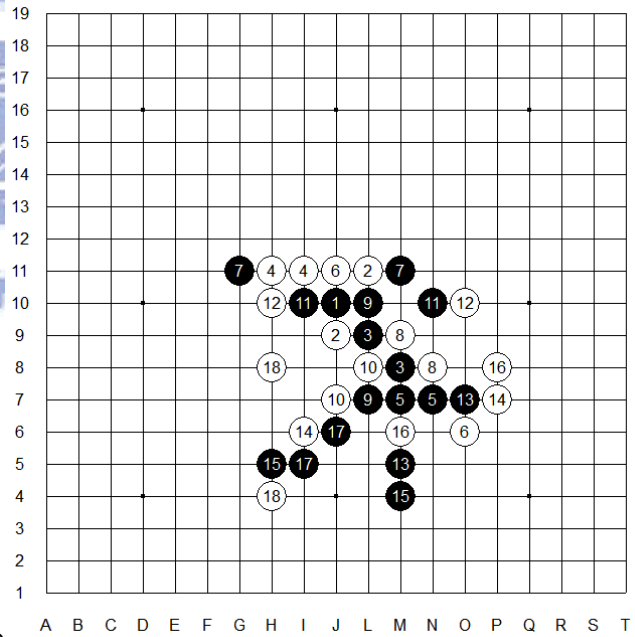
57



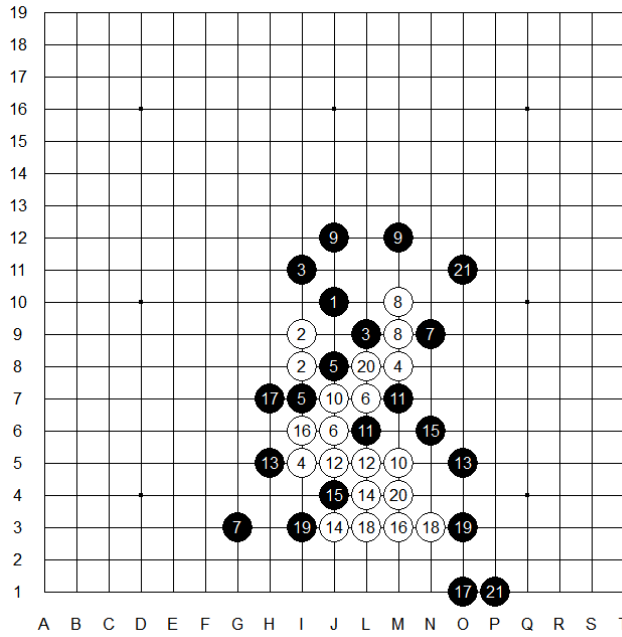
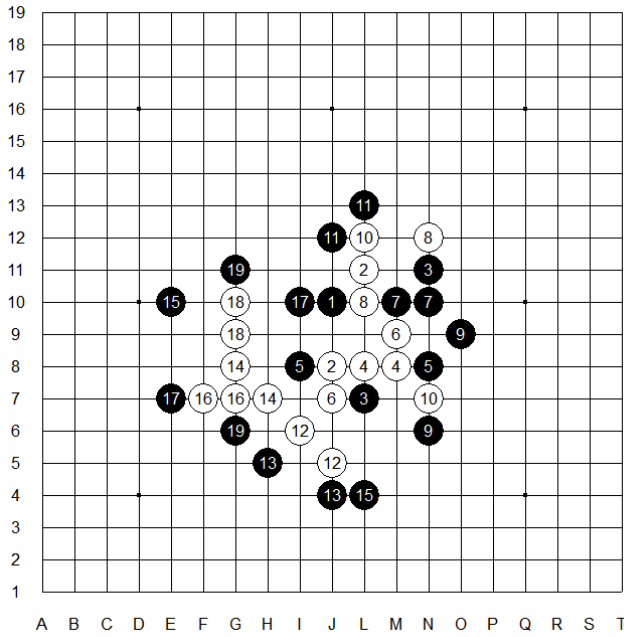
58



59

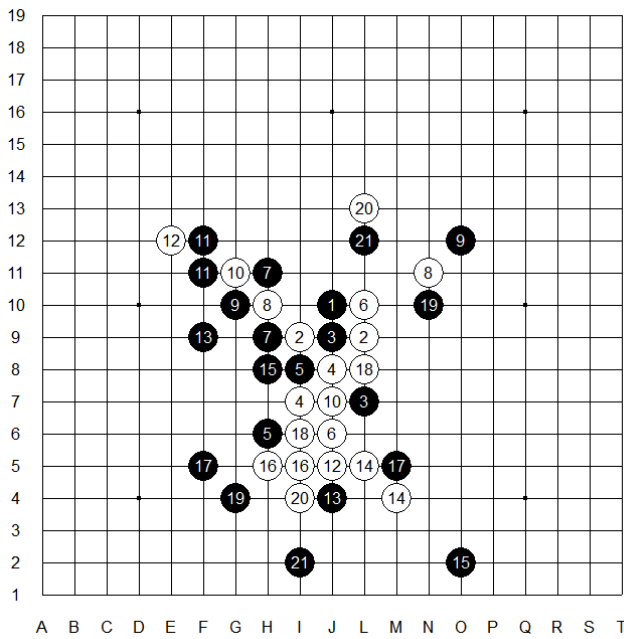


60

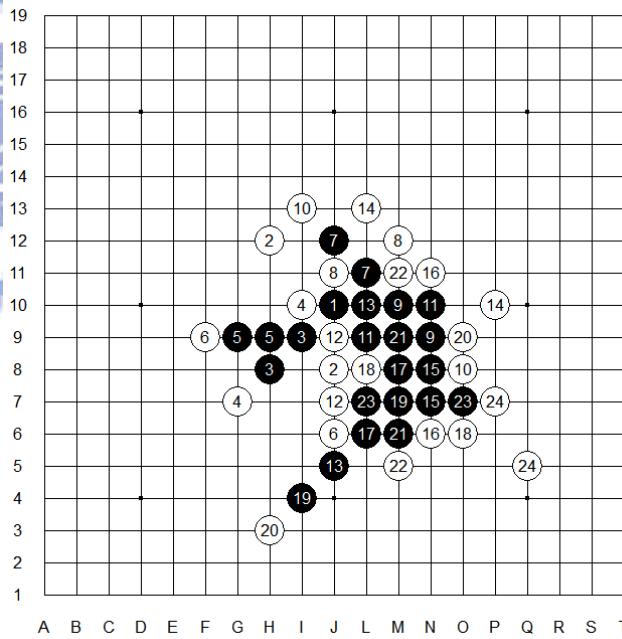


61

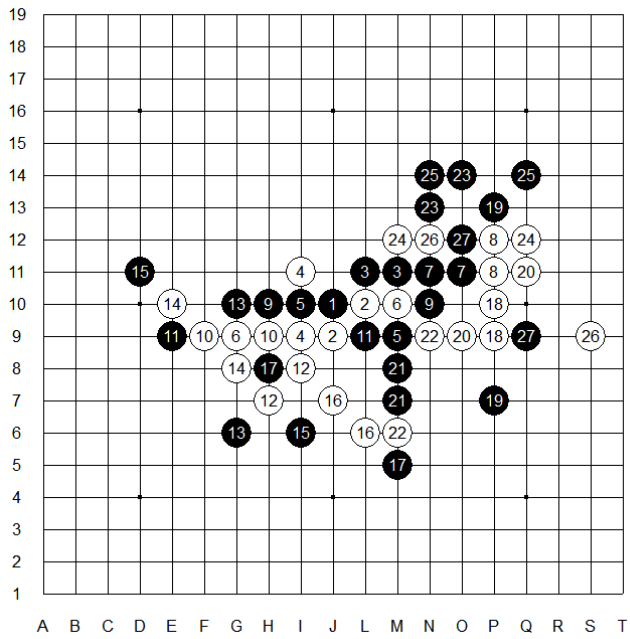
62



63



64



65



## Appendix B Results of RZOP Benchmark

Table 9. The solvability of verifiers for 65 winning positions in Appendix A, where “yes” means solved and “no” means unsolved.

Positions	Solvability	
	$V_{C6-02}$ , $V_{C6-01}$ and $V_{C6}$	$V_{Wu}$
01	yes	yes
02	yes	no
03	yes	no
04	yes	no
05	yes	no
06	yes	no
07	yes	no
08	yes	no
09	yes	no
10	yes	no
11	yes	yes
12	yes	no
13	yes	no
14	yes	yes
15	yes	yes
16	yes	yes
17	yes	yes

18	yes	yes
19	yes	yes
20	yes	yes
21	yes	yes
22	yes	yes
23	yes	yes
24	yes	yes
25	yes	yes
26	yes	no
27	yes	no
28	yes	no
29	yes	no
30	yes	no
31	yes	no
32	yes	no
33	yes	no
34	yes	no
35	yes	yes
36	yes	yes
37	yes	no
38	yes	no
39	yes	no
40	yes	yes
41	yes	yes

42	yes	no
43	yes	no
44	yes	no
45	yes	no
46	yes	no
47	yes	yes
48	yes	no
49	yes	no
50	yes	yes
51	yes	no
52	yes	yes
53	yes	yes
54	yes	yes
55	yes	yes
56	yes	no
57	yes	yes
58	yes	yes
59	yes	yes
60	yes	yes
61	yes	yes
62	yes	no
63	yes	yes
64	yes	no
65	yes	yes

Total solved	65	31
Total unsolved	0	34





## Appendix C Results of SRZOP Benchmark

Table 10. The statistics of verifiers for 65 winning positions in Appendix A: (a) number of nodes and (b) times.

(a)

Number of nodes	$V_{C6-02}$	$V_{C6-01}$	$V_{C6}$
01	530077	571795	565926
02	3523754	3590585	5527224
03	786302	845114	1575020
04	1079619	1116986	1331077
05	6174061	6167216	7317401
06	1661531	2240151	2752565
07	23908274	24134145	35585383
08	20639430	20700223	96128237
09	2548691	2581346	3387769
10	1834412	1460326	1710532
11	1146025	1247001	994735
12	2683237	2728336	4196015
13	28526645	28686442	40210832
14	74803	87805	94972
15	17450	22272	29299
16	16368	16368	16547
17	10718	10718	10778

18	67852	69610	71727
19	84402	92292	90870
20	10145	10145	10708
21	22344	25320	27540
22	135089	135089	135277
23	35420	36960	37823
24	704489	704589	705905
25	37603	39307	46686
26	2119172	2120955	2813168
27	109546	109632	110342
28	408452	420402	426156
29	137182	141895	142339
30	126007	134785	137648
31	551373	552034	902124
32	642143	651712	665022
33	59855	60111	62018
34	181191	149634	172601
35	2105266	2111350	3377070
36	2761	2761	2824
37	9030296	9030296	10674017
38	765141	765705	768514
39	12363329	12492105	18321351
40	731486	731584	732403
41	393962	394038	394528

42	4045655	4046601	5511607
43	56696	56944	57784
44	7660096	7682555	11245412
45	860379	940666	1047951
46	209423	209987	211208
47	22278	22298	22459
48	706779	715971	660724
49	7816945	7819769	6893666
50	5041761	5041761	5042228
51	2054406	2066930	2851817
52	254617	296116	353003
53	898630	899734	954147
54	925359	940417	939580
55	5266143	5276358	5962500
56	297172	300223	456519
57	239318	246013	247606
58	1077354	1093460	1117277
59	1455834	1518244	1512576
60	1292694	1307825	1321979
61	779609	835836	832882
62	256408	257724	261066
63	1336806	1286361	1436594
64	8740	9380	9372
65	9471114	9472070	13272361

Total	178020119	179532383	304485291
-------	-----------	-----------	-----------

(b)

Time (in seconds)	$V_{C6-02}$	$V_{C6-01}$	$V_{C6}$
01	202.29	255.17	233.15
02	638.47	700.26	824.64
03	182.85	245.59	290.27
04	212.47	245.73	266.15
05	719.22	794.18	750.77
06	700.42	843.82	915.39
07	2309.55	2437.09	3051.79
08	2325.26	2407.76	9998.87
09	413.40	461.54	531.23
10	649.50	592.64	683.68
11	286.61	379.38	382.67
12	395.24	451.95	544.13
13	4121.33	4387.67	5082.57
14	47.79	66.59	66.35
15	7.59	12.41	14.16
16	6.17	6.17	6.28
17	1.91	1.91	1.89
18	10.28	11.23	12.94
19	15.69	25.77	24.55
20	2.91	2.91	3.20
21	14.72	18.25	19.48

22	16.27	16.25	16.34
23	9.27	11.13	11.89
24	78.48	78.24	80.33
25	11.13	13.25	16.66
26	211.06	212.94	240.45
27	44.89	44.67	45.11
28	150.06	189.81	188.37
29	56.56	64.16	64.70
30	113.45	141.08	134.59
31	61.31	62.34	76.09
32	167.33	177.30	191.63
33	15.56	15.80	16.61
34	101.03	84.44	90.84
35	194.44	200.88	264.47
36	1.05	1.05	1.05
37	739.55	738.89	810.45
38	116.78	117.47	117.50
39	1034.14	1048.08	1286.75
40	84.20	84.23	84.42
41	26.28	26.36	26.69
42	329.26	328.83	394.71
43	15.70	16.31	17.44
44	1219.40	1269.82	1648.19
45	608.90	855.99	831.91

46	84.09	85.14	86.02
47	6.47	6.48	6.61
48	202.47	215.02	204.85
49	1974.56	1981.13	1776.99
50	537.95	537.72	534.55
51	489.94	516.23	594.78
52	136.75	182.25	231.73
53	249.05	249.22	262.16
54	112.28	112.85	124.41
55	1153.77	1169.50	1353.61
56	46.30	46.88	58.14
57	65.14	66.37	66.03
58	304.24	305.89	314.13
59	600.23	600.51	625.50
60	362.66	373.33	430.28
61	341.85	381.89	481.20
62	166.36	165.78	185.66
63	160.17	146.62	160.78
64	27.78	27.77	27.91
65	664.81	664.02	866.92
Total	26356.62	27981.87	38753.57

## Appendix D Verifiers for General Connect Games

In this Appendix, the verifier  $V_{C6}(P,S)$  is generalized to general Connect games,  $Connect(m,n,k,p,q)$ , while maintaining Property RZV.

The generalized verifier is denoted by  $V_{CK}(P,S)$ . In the case that  $P$  is an endgame position or is in Attacker's turn (described in Subsections 3.2.1 and 3.2.2 respectively), the verifier  $V_{CK}(P,S)$  is the same as  $V_{C6}(P,S)$ . So, the rest of this appendix describes the verifier only in the case that  $P$  is in Defender's turn. Furthermore, the position  $P$  (in Defender's turn) can be classified into the following two. (1) The number of Attacker threats  $t$  in  $P$  is at least  $p + 1$ , and (2) the number  $t$  is at most  $p$ . In the first case, Attacker wins already. Therefore, the verifier returns 1 and construct relevance zones in the following operation.

**Tp1-1.** Construct relevance zones by following both operations T3-1 and T3-2, except that the terms " $i + 2$ " are replaced by " $i + p$ ".

Similar to Lemma 7, Lemma 31 shows that the verifier also satisfies Property RZV in this case.

**Lemma 31.** Assume that Defender is to move and the number of Attacker threats is at least  $p + 1$  in  $P$ . The verifier described above satisfies Property RZV.

**Proof.** The proof is similar to that of Lemma 7 and therefore omitted. ■

In the second case that the number of Attacker threats  $t$  is at most  $p$ , the verifier performs the following operations.

- Tp-1.** For each of critical defenses  $M_D$  (both normal and relaxed), perform the following.
- a. Return 0 if the sub-verifier  $V_{sub}(M_D, P, S)$  returns 0. Note that the sub-verifier is described below.
  - b. Let  $\Psi(P) = \Psi(P) \cup \Psi'(P_D)$ .
- Tp-2.** Continue to construct relevance zones in operation Tp1-1, and return 1.

In operation Tp-1.a, a sub-verifier  $V_{sub}(M_D, P, S)$  is used to verify whether Attacker wins for all Defender moves  $M'_D$  dominated by  $M_D$  in  $P$ , where  $M'_D$  has  $p$  squares (but  $M_D$  may have less than  $p$  squares). By *dominate*, we mean that all squares in  $M_D$  must also be in  $M'_D$ , but may not *vice versa*. For the sub-verifier  $V_{sub}(M_D, P, S)$ , the constructed zones is denoted by  $\Psi'(P_D) = \langle Z'_1(P_D), Z'_2(P_D), \dots, Z'_t(P_D) \rangle$ , where  $P_D = P \oplus M_D$ . In addition, the sub-verifier satisfies the following property (proved in Lemma 32).

**Property RZS.** If  $V_{sub}(M_D, P, S)$  returns 1, the following condition holds. For all Defender moves  $M'_D$  dominated by  $M_D$ , there exists some  $\Psi'_D$  such that  $\Psi'_D \subseteq \Psi'(P_D)$  and  $\Psi'_D$  is in  $RZ(P \oplus M'_D)$ .

The sub-verifier  $V_{sub}(M_D, P, S)$  performs the following operations.

- Par-1.** Assume that  $M_D$  has exactly  $p - u$  Defender stones, where  $u$  is the number of null stones in  $M_D$  and  $0 \leq u \leq p$ . In the case that  $u > 0$ , move  $M_D$  is a null or semi-null move.
- Par-2.** Return 0 if  $V_{CK}(P_D, S)$  returns 0, where  $P_D = P \oplus M_D$ .
- Par-3.** Let  $\Psi'(P_D) = \Psi(P_D) \ll u$ .
- Par-4.** Return 1 if  $u = 0$ , i.e., the move is not a null or semi-null move.



**Par-5.** For each of unoccupied square  $s \in \neg_{PD}(Z_u(P_D))$ , perform the following.

- a. Let Defender move  $M_{D,s}$  be  $M_D + \sigma_D(s)$ .
- b. Return 0 if  $V_{sub}(M_{D,s}, P, S)$  returns 0.
- c. Let  $\Psi'(P_D) = \Psi'(P_D) \cup \Psi'(P_{D,s})$ , where  $P_{D,s} = P \oplus M_{D,s}$ .

**Par-6.** Return 1.

Lemma 32 shows that the sub-verifier satisfies Property RZS, if all the recursive  $V_{sub}$  in Par-5.b satisfy Property RZS and the verifier  $V_{CK}$  in Par-2 satisfies Property RZV.

**Lemma 32.** For a sub-verifier  $V_{sub}(M_D, P, S)$  as described above, it satisfies Property RZS by assuming that all the recursive  $V_{sub}$  in Par-5.b satisfy Property RZS and that the verifier  $V_{CK}$  in Par-2 satisfies Property RZV.

**Proof.** Assume that  $V_{sub}(M_D, P, S)$  returns 1. Consider all Defender moves  $M'_D$  (including  $p$  stones) that are dominated by  $M_D$ . Namely, let  $M'_D = M_D + \sigma_D(\varphi)$ , where  $\varphi$  has  $u$  additional unoccupied squares. For this lemma, it suffices to prove that there exists some  $\Psi'_D$  such that  $\Psi'_D \subseteq \Psi'(P_D)$  and  $\Psi'_D$  is in  $RZ(P \oplus M'_D)$ . All of these Defender moves  $M'_D$  are classified into the following cases.

1. All Defender moves  $M'_D$  in which all additional squares  $s$  in  $\varphi$  are in  $\neg_{PD}(Z_u(P_D))$ .  
The proof for this case is similar to that for Case 1 in Lemma 10 as follows. Since this sub-verifier returns 1, the verifier  $V_{CK}(P_D, S)$  in Par-2 returns 1. Since the verifier  $V_{CK}$  returns 1 and also satisfies Property RZV (from this lemma),  $\Psi(P_D)$  is in  $RZ(P_D)$ . Since all additional  $s \in \neg_{PD}(Z_u(P_D))$ , we obtain from Lemma 3 that  $\Psi(P_D) \ll u$  is in  $RZ(P_D + \sigma_D(\varphi))$ . Since  $P_D + \sigma_D(\varphi) = (P \oplus M_D) + \sigma_D(\varphi) = P \oplus (M_D + \sigma_D(\varphi)) = P \oplus M'_D$ ,  $\Psi(P_D) \ll u$  is also in  $RZ(P \oplus M'_D)$ . In addition, since  $\Psi(P_D) \ll u \subseteq \Psi'(P_D)$  from Par-3 in  $V_{sub}$ ,  $\Psi(P_D) \ll u$  is the  $\Psi'_D$ .
2. All Defender moves  $M'_D$  where some additional square  $s$  in  $\varphi$  is in  $Z_u(P_D)$ . Since this

sub-verifier returns 1, the recursive  $V_{sub}(M_{D,s}, P, S)$  at Par-5.b returns 1 too and therefore satisfies Property RZS. From Property RZS, there exists some  $\Psi$  such that  $\Psi \subseteq \Psi'(P_{D,s})$  and  $\Psi$  is in  $RZ(P \oplus M'_D)$ . Since  $\Psi'(P_{D,s}) \subseteq \Psi'(P_D)$  from operation Par-5.c, we obtain  $\Psi \subseteq \Psi'(P_D)$ . Thus,  $\Psi$  is the  $\Psi'_D$ . ■

From Lemma 32, we derive Lemma 33 as follows.

**Lemma 33.** Assume that Defender is to move and the number of Attacker threats is at most  $p$  in  $P$ . The verifier described above satisfies Property RZV by assuming that all the recursive sub-verifiers in operation Tp-1.a satisfy Property RZS.

**Proof.** Assume that this verifier returns 1. For this lemma, it suffices to prove that the constructed  $\Psi(P)$  is in  $RZ(P)$ . Since the verifier returns 1, all the recursive sub-verifiers in operation Tp-1.a returns 1 too. Assume that these sub-verifiers satisfy Property RZS. For proving  $\Psi(P) \in RZ(P)$ , it suffices to prove from Lemma 6 the following: For all Defender moves  $M_D$  there exists some  $\Psi_D$  such that  $\Psi_D$  is in  $RZ(P \oplus M_D)$  and  $\Psi_D \subseteq \Psi(P)$ . All Defender moves  $M_D$  are classified into the following two cases:

1. All Defender moves  $M_D$  that block all the threats. There must exist some critical defense  $M'_D$  (either normal or relaxed) dominating  $M_D$ . Since  $V_{sub}(M'_D, P, S)$  returns 1 and satisfies Property RZS from above, there exists some  $\Psi_D$  from the property such that  $\Psi_D \subseteq \Psi'(P \oplus M'_D)$  and  $\Psi_D$  is in  $RZ(P \oplus M'_D)$ .
2. All Defender moves  $M_D$  that leave some threat unblocked. Attacker wins by connecting up to  $p$  on some unblocked threat segment, like  $S_{3T}$ . From the proof in Lemma 31, we obtain that there exists some  $\Psi_D$  such that  $\Psi_D \subseteq \Psi'(P)$  and  $\Psi_D$  is in  $RZ(P_D)$ . ■

Theorem 5 (below) concludes that the verifier  $V_{CK}(P,S)$  in all cases satisfy Property RZV. Therefore, if  $V_{CK}(P,S)$  returns 1, the constructed  $\Psi(P)$  is in  $RZ(P)$ , and Attacker wins in  $P$  from Corollary 2. It can also be observed that the operations in Subsection 3.2.3 are special cases of the operations described in this appendix.

**Theorem 5.** The verifier  $V_{CK}(P,S)$  satisfies Property RZV in all cases.

**Proof.** By induction, the verifier  $V_{CK}(P,S)$  satisfies Property RZV in all cases from the above lemmas. ■



## Appendix E Draw K-in-a-row Games

In the past, many researchers were engaged in solving  $Connect(m, n, k, p, q)$  games. One player, either Black or White, is said to *win* a game, if he has a winning strategy such that he wins for all the subsequent moves. Allis *et al.* [1][2] solved *Go-Moku* with Black winning. Herik *et al.* [22] and Wu *et al.* [65][66] also mentioned several  $k$ -in-a-row games with Black winning.

A game is said to be *drawn* if neither player has any winning strategy. For simplicity,  $Connect(k, p)$  refers to the collection of  $Connect(m, n, k, p, q)$  games for all  $m \geq 1, n \geq 1, 0 \leq q \leq p$ .  $Connect(k, p)$  is said to be *drawn* if all  $Connect(m, n, k, p, q)$  games in  $Connect(k, p)$  are drawn.

In the past, Zetters [69] derived that  $Connect(8, 1)$  is drawn. Pluhar [38] derived tight bounds  $k_{draw}(p) = p + \Omega(\log_2 p)$  for all  $p \geq 1000$  (cf. Theorem 1 in [38]). However, the requirement of  $p \geq 1000$  is unrealistic in real games. Thus, it is important to obtain tight bounds when  $p < 1000$ . Hsieh and Tsai [24] have recently derived that  $k_{draw}(p) = 4p + 7$  for all positive  $p$ . The ratio  $R = k_{draw}(p)/p$  is approximately 4 for sufficiently large  $p$ .

Given  $p$ , Chiang *et al.* [15] derive the value  $k_{draw}(p)$ , such that  $Connect(m, n, k, p, q)$  are drawn for all  $k \geq k_{draw}(p), m \geq 1, n \geq 1, 0 \leq q \leq p$ , as follows. (1)  $k_{draw}(p) = 11$ . (2) For all  $p \geq 3$ ,  $k_{draw}(p) = 3p + 3d - 1$ , where  $d$  is a logarithmic function of  $p$ . So, the ratio  $k_{draw}(p)/p$  is approximately 3 for sufficiently large  $p$ . The first result was derived with the help of a program. To our knowledge, our  $k_{draw}(p)$  values are currently the smallest for all  $2 \leq p < 1000$ .

## Appendix F Author's Records

The game Connect6 was first introduced by Wu and Huang (2005) and then described in more detail by Wu, Huang and Chang (2006). The rules of Connect6 are very simple. Two players, henceforth represented as B (designated as the first player) and W, alternately place two stones, black and white respectively, on one empty intersection of an 19×19 board, except for that B places one stone initially. The player who first obtains six consecutive stones (horizontally, vertically or diagonally) wins the game. When all intersections on the board are occupied without connecting six, the game draws.

Starting from 2007, Lin became the chief designer of the Connect6 program *NCTU6*. Though, *NCTU6* won the Gold Prize of the Connect6 Tournament in the 11th Computer Olympiad (2006), there were many unsolved positions and openings. Thus, Lin solved many unsolved VCST positions in the beginning and help developed some simple openings. With the improved strength of *NCTU6*, Lin developed a light weight version with accurate time control program named *NCTU6-LITE*, which won the Gold Prize of the Connect6 Tournament in the 13th Computer Olympiad (2008). The participants and the final standings are listed in Table 11 (below).

In the tournament, the games were played according to a round-robin system in which one program played twice against all the other programs. In each game, every program had to complete all of its moves in 30 minutes. For each game, the winner scored 1 point and the loser scored nothing. However, for a draw game, both scored 0.5. Figure 35 and Figure 36 (below) show some events in the 13th Computer Olympiad. The certificate of the 13th Computer Olympiad by NCTU is shown in Figure 37 (below).

Ranking	Program	Author	Organization	Points
1	<i>NCTU6-LITE</i>	Ping-Hung Lin, Hong-Xuan Lin, Yi-Chih Chan, Ching-Ping Chen and I-Chen Wu	National Chiao Tung University, Taiwan.	17
2	<i>BITSTRONGER</i>	Liang Li, Hao Cui, Ruijian Wang and Siran Lin	Beijing Institute of Technology, China	16
3	<i>NEUCONN6</i>	Chang-Ming Xu	Northeastern University, China	13
4	<i>BEAD CONNECT AND CHESS COMBINE (BCCC)</i>	XiaoChuan Zhang	Chongqing Institute of Technology, China	9
5	<i>KAVALAN</i>	Jung-Kuei Yang and Shi-Jim Yen	National Dong Hwa University, Taiwan	8
6	<i>NEU6STAR</i>	Xinhe Xu, Dongxu Huang, Junjie Tao, Kang Han, XinXing Li	Northeastern University, China	8
7	<i>ML</i>	Jiang Ke	Guilin University of Electronic Technology, China	6.5

8	<i>CV6</i>	Yao Yuping	Guilin University of Electronic Technology, China	5.5
9	<i>DREAM 6</i>	Siwei Liu and Zhenhua Huang	Dalian Jiaotong University, China	4
10	<i>NTNU C6</i>	Shih-Chieh Huang and Yun-Ching Liu	National Taiwan Normal University, Taiwan	3

Table 11. The participants and the final standings of the Connect6 Tournament in the 13th Computer Olympiad (2008).



Figure 35. P. H. Lin, I-C. Wu and H.J. van den Herik.



Figure 36. L. Lee (*BITSTRONGER*) and P. H. Lin (*NCTU6-LITE*).



Figure 37. The certificate of the 13th Computer Olympiad by NCTU.



In Taiwan, National Chiao Tung University hosts the annual NCTU Cup Open Tournaments for Connect6 human players. We saw more and more players played Connect6 every year. Before the second annual NCTU Cup Open Tournament 2008 took place, Wu invited Go Champion Chou Jun Xun to play Connect6 against the AI program *NCTU6* for the advertisement. In this championship, *NCTU6* won 3 and lost nothing against Chou. Figure 38 shows an event in the championship.

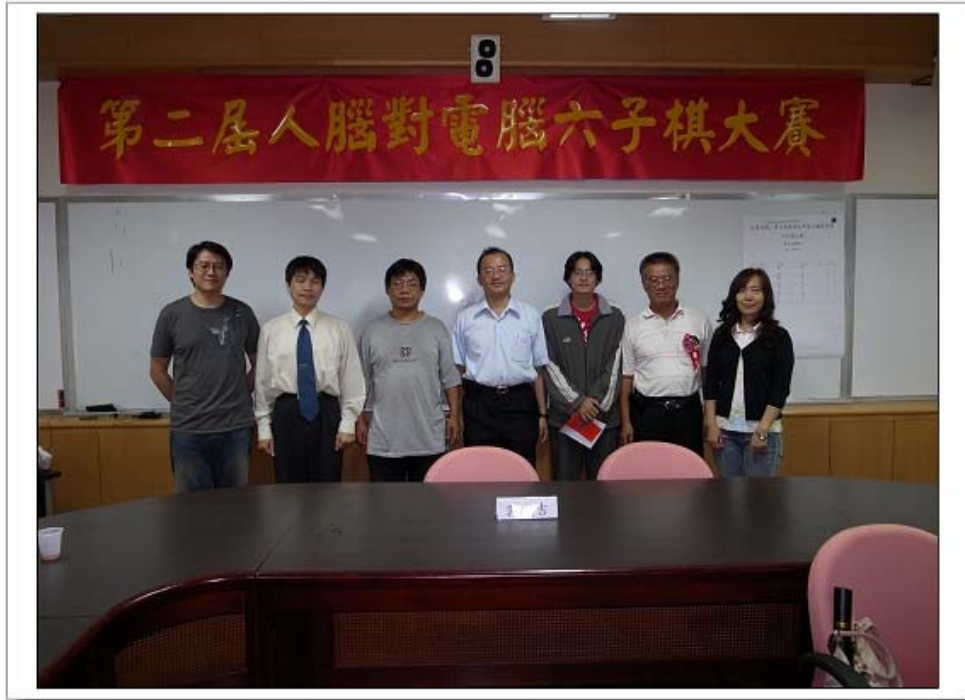


Figure 38. Go Champion Chou Jun Xun, the operating staff and P. H. Lin.

After annual NCTU Cup Open Tournaments, yearly top human players of Connect6 will appear. To survey the strength of *NCTU6*, Wu will invite three to four top players to play against *NCTU6*. Figure 39 and Figure 40 (below) show events of the first and the second Man-Machine Connect6 Championships.



Figure 39. Professor Shun-Chin Hsu (right most) and members of the Connect6 team lead  
by I-C. Wu.



6RDIGITAL 3 Fl.9 1/325150125

2009.10.11 - The Second Annual Man-Machine Connect6 Contest

Figure 40. Human players, I-C. Wu (center) and P. H. Lin (left most).

In Figure 39, Professor Shun-Chin Hsu is respected as the father of Computer Chinese Chess. He has received many awards and published many important papers. In the first annual Man-Machine Connect6 Contest, we are very happy to invite Professor Hus to host the contest. In the contest, *NCTU6* won 11 and lost one against top human players. It is a good record. Next year, in the second annual Man-Machine Connect6 Contest, *NCTU6* won 8 and lost nothing which is a memorable record.

From these records, Lin proved the strength of *NCTU6*. He will continue to develop *NCTU6* and keep *NCTU6* the top AI program of Connect6 in the world.



## Vita

Ping-Hung Lin was born in Hualien, Taiwan in 1978. He received the B.S., M.S. and Ph.D. degree in Computer Science and Information Engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2000, 2002 and 2010, respectively. He is the current chief designer of the Connect6 program *NCTU6* that won the gold twice in Computer Olympiad in both 2006 and 2008. His research interests include artificial intelligence and grid and cloud computing.

