

國立交通大學

資訊科學與工程研究所

博士論文

應用組合對局知識解決遊戲及改良
搜尋效能



Solving Games and Improving Search Performance
with Embedded Combinatorial Game Knowledge

研究生：單益章

指導教授：吳毅成 教授

高國元 教授

中華民國一〇二年十月

應用組合對局知識解決遊戲及改良搜尋效能
Solving Games and Improving Search Performance with Embedded
Combinatorial Game Knowledge

研究生：單益章
指導教授：吳毅成
高國元

Student : Yi-Chang Shan
Advisor : I-Chen Wu
Kuo-Yuan Kao

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

October 2013

Hsinchu, Taiwan, Republic of China

中華民國 一〇二 年 十 月

應用組合對局知識解決遊戲及改良搜尋效能

研究生：單益章

指導教授：吳毅成 博士

高國元 博士

國立交通大學資訊科學與工程研究所博士班

摘要

本篇論文，主要是應用組合對局知識解決遊戲及改良搜尋效能。組合對局理論已經成為許多益智遊戲分析的基本數學模型，其利用數學代數的特性來降低問題的複雜度。本論文主要研究三種組合對局遊戲，包括三角殺棋(Triangular Nim)、XT Domineering 及 NoGo。

首先，三角殺棋是一種 Nim 的變形，是流行台灣及中國的二人遊戲。本論文使用 Retrograde 方法，全解九層三角殺棋盤面，並運用旋轉及對稱方法，降低記憶空間需求達 5.72 倍，並增進運算速度達 4.62 倍。

第二、本論文介紹新的組合對局遊戲 XT Domineering 及其數學分析，XT Domineering 其變化來自於 Domineering。變化後的規則，在遊戲中所有盤面皆為微數字(Infinitesimal)，計算得出所有 3×3 盤面的遊戲值(Game values)，並得出每一盤面的遊戲值為 8 種基本微數字的線性組合，利用一個簡單代數和的式子，即可以迅速算出雙方勝敗，以取代搜尋全部遊戲樹(Game tree)。

第三、本論文分析 2011 年 BIRS 組合對局會議提出的一種新的組合對局遊戲 NoGo，計算其許多 4×4 NoGo 盤面的遊戲值，發現其最高溫度 (Temperature) 為 2，並在研究中推導出一些基本定理，以幫助我們瞭解 NoGo 遊戲特性。

Solving Games and Improving Search Performance with Embedded Combinatorial Game Knowledge

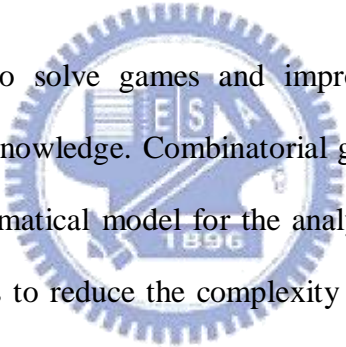
Student : Yi-Chang Shan

Advisor : Dr. I-Chen Wu

Dr. Kuo-Yuan Kao

Institute of Computer Science and Engineering
National Chiao Tung University

Abstract



In this thesis, we study to solve games and improve search performances with embedded combinatorial game knowledge. Combinatorial game theory (CGT) has become the common fundamental mathematical model for the analysis of many intelligent games. CGT uses algebra characteristics to reduce the complexity of many intelligent games. For this study, we investigate three combinatorial games, including Triangular Nim, XT Domineering and NoGo.

First, Triangular Nim, one variant of the game Nim, is a common two-player game in Taiwan and China. Using a retrograde method, we strongly solve nine layer Triangular Nim. In our design, improved by removing some rotated and mirrored positions, the program reduces the memory by a factor of 5.72 and the computation time by a factor of 4.62.

Secondly, we introduce a new combinatorial game, named XT Domineering, together with its mathematical analysis. XT Domineering is modified from the Domineering game with the game value of each position becoming an infinitesimal. We calculate the game values of all 3×3 positions and shows that each 3×3 position's game value is a linear

combination of 8 elementary infinitesimals. A simple rule is presented to determine the optimal outcome of any sum of these positions, instead of searching the whole game trees.

Thirdly, NoGo is a game introduced by the organizers of the BIRS workshop on Combinatorial Game Theory 2011 for being a completely new combinatorial game. We calculate the game values of many 4×4 NoGo positions and find the maximum of temperature is 2 among them. We also present some propositions to help us understand the characteristics of NoGo game.



致謝

感謝指導老師吳毅成教授及高國元教授多年來的提攜與照顧，不斷地指導及鼓勵我做研究。二位老師在理論和實務方面的教學研究，指引我研究的方向，我將秉持老師的教導，在未來的道路上，一步步努力前進。

除了影響我最深的老師之外，特別感謝我的內人，在研究過程中，一直鼓勵、扶持和砥礪我，讓我無後顧之憂的研究，順利通過這充滿困難、歡欣和淚水的博士道路。

感謝論文口試委員吳昇教授、林順喜教授、許舜欽教授、陳榮傑教授、蔡錫鈞教授和顏士淨教授（以上按姓氏筆劃排列），對論文的改進方向，提出寶貴的意見，讓我的研究能更上層樓。

培育了我多年的交大和我在 CYC Lab 的所有好友們，給予我各式各樣的支持，在此衷心的感謝大家，包括隆彬學長、秉宏學長，博班同學德中、宏軒、汶傑，及學弟冠翬、益嘉、博玄、挺富、元耀、振綱、庭築、育賢…等等，特別感謝學弟在我的口試時細心地幫我準備餐點。

最後，感謝從小教導我的爸爸、媽媽及和讓我無後顧之憂的岳父、岳母及所有幫助我的兄弟姊妹，讓我可以專心在論文研究，讓我在遇到困難時，內心仍能充滿溫暖向前。沒有您們，這篇論文將無法完成。謹以此論文獻給我敬愛的師長及最摯愛的家人。

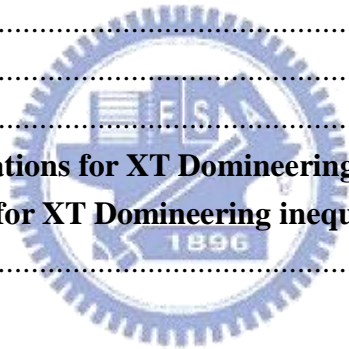
單益章

2013 年 10 月 12 日

Contents

摘要.....	i
Abstract.....	ii
致謝.....	iv
Contents.....	v
List of Figures.....	vii
List of Tables.....	ix
Chapter 1 Introduction.....	1
1.1 Background.....	1
1.2 Game tree search.....	2
1.2.1 Min-Max Tree.....	2
1.2.2 Monte Carlo Tree Search.....	3
1.3 Combinatorial Game Theory.....	5
1.4 Framework of Research.....	7
1.5 Organization.....	8
Chapter 2 Triangular Nim.....	9
2.1 Introduction.....	9
2.2 Related Work.....	12
2.2.1 Backward Induction Approach.....	12
2.2.2 Retrograde Method.....	14
2.3 Solving Approach.....	15
2.3.1 Data Structures and Algorithms.....	15
2.3.2 Removing Redundancy.....	19
2.4 Experimental results.....	22
2.4.1 Eight Layer Triangular Nim.....	22
2.4.2 Nine Layer Triangular Nim.....	25
2.5 Conclusion.....	28
Chapter 3 XT Domineering: A New Combinatorial Game.....	30
3.1 Introduction.....	30
3.2 Combinatorial Games.....	31
3.2.1 Numbers.....	32
3.2.2 Nimbers.....	32
3.2.3 Sumbers.....	33
3.2.4 Infinitesimal and Atomic Weight.....	35

3.3	Domineering and XT Domineering.....	36
3.4	Game Values of 3×3 XT Domineering.....	38
3.5	Outcome of 3×3 XT Domineering	43
3.6	Conclusion And Further Consideration	46
Chapter 4	NoGo Endgame Analysis	48
4.1.	Introduction	48
4.2.	Classification of Moves.....	49
4.3.	Mean and Temperature	50
4.3.1	Definitions of Mean and Temperature	51
4.3.2	Thermograph.....	52
4.4.	Game Values of NoGo Positions	54
4.4.1	Game Values	55
4.4.2	Means and Temperatures	58
4.4.3	More Analysis	63
4.5.	NoGo Propositions.....	64
4.6.	Conclusion	72
Chapter 5	Conclusions	73
References	75
Appendix A	The Derivations for XT Domineering Games Values of Positions....	81
Appendix B	The proof for XT Domineering inequalities.....	85
Vita	91



List of Figures

Figure 1. An example of minimax Tree.	2
Figure 2. Monte Carlo Tree Search	5
Figure 3. The framework of research	7
Figure 4. A scenario of Triangular Nim with side size five.	10
Figure 5. A piece mapping of 5 layer Triangular Nim.	13
Figure 6. Six block representations for 8 layer Triangular Nim.	16
Figure 7. Inter-block updates of the block representations in Figure 6 (a) and (b), respectively.	19
Figure 8. Six rotated and mirrored blocks.	20
Figure 9. An isomorphic group with one, two and three distinct blocks only.	21
Figure 10. The moves to win in 8 layer normal Triangular Nim.	24
Figure 11. The moves to win in 8 layer misère Triangular Nim.	24
Figure 12. Three block representations with 13 BID pieces.	25
Figure 13. Three block representations with 15 BID pieces.	26
Figure 14. The moves to win in 9 layer normal Triangular Nim.	27
Figure 15. The moves to win in 9 layer misère Triangular Nim.	27
Figure 16. The ratio of the number of P-positions to that of N-positions.	28
Figure 17. Middle game of 6×6 Domineering.	36
Figure 18. Sub-positions of the graph in Figure 17.	37
Figure 19. Some game values in Domineering.	37
Figure 20. Some game values in XT Domineering.	42
Figure 21. Some game values in XT Domineering.	43
Figure 22. Some game values in XT Domineering.	46
Figure 23: The classification of moves in NoGo.	49
Figure 24: Thermographs of (a) a simple max function and (b) a simple min function.	53
Figure 25: The thermograph of $G = \{3 \{0 - 2\}\}$	54
Figure 26. A specific 5×5 Nogo position.	62
Figure 27. The temperature is greater than 2 in a 4×7 NoGo position.	63
Figure 28. The example of NoGo definitions.	65
Figure 29. Each No-Go has game value zero.	66
Figure 30. The example of B-Dragons and W-Dragons.	67
Figure 31. Three different walls in NoGo.	68

Figure 32. The board forms left and right independent regions.....69

Figure 33. The symbol of black triangle is represented for B-EGo71

Figure 34. The example of B-Ego and W-Ego.....72

Figure 35. Deriving both game values of C and E of Figure 18 in (a) and (b) respectively.81

Figure 36. Deriving the game values of $C + E$82

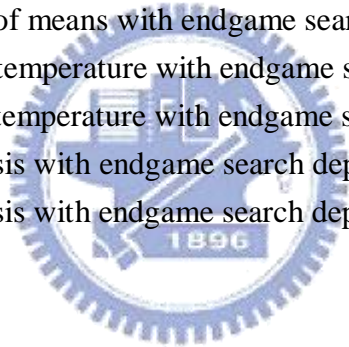
Figure 37. Deriving both game values of C and E of Figure 18 in (a) and (b) respectively.83

Figure 38. Deriving the game values of $C + E$84



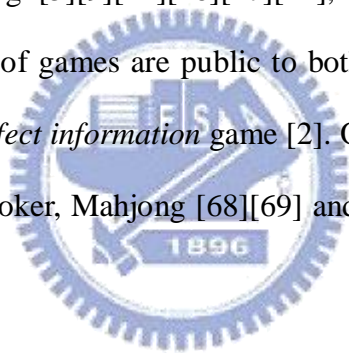
List of Tables

Table 1. Experimental results for the block representations without removing redundancy in Figure 6.	23
Table 2. Experimental results for three block representations with removing redundancy in Figure 6.	24
Table 3. Experimental results for the block representations in Figure 12.	25
Table 4. Experimental results for the block representations in Figure 13.	26
Table 5. The result of k layer Triangular Nim.	28
Table 6. Game values of 3×3 XT Domineering	41
Table 7. Minimum ups U required for $U + S_B + S_C > 0$	44
Table 8. The list of special game values of 4×4 NoGo.....	58
Table 9. The highest and lowest of means with endgame search depth 6 and 8 of 4×4 NoGo	59
Table 10. The lists of maximum temperature with endgame search depth 6 in 4×4 NoGo.	61
Table 11. The lists of maximum temperature with endgame search depth 8 in 4×4 NoGo.....	62
Table 12. The temperature analysis with endgame search depth 6 in 4×4 NoGo positions.	64
Table 13. The temperature analysis with endgame search depth 8 in 4×4 NoGo positions.	64



Chapter 1 Introduction

Combinatorial game theory (CGT) [7][20] has become the common fundamental mathematical model for the analysis of many intelligent games. In CGT, a game can be viewed as a tree where the branches are classified into left and right branches and the terminal nodes are numbers. A sum of games is a collection of trees where each player can choose one to move at a turn. Combinatorial games include many games like chess [23][49], Chinese chess [26], Go [6][53], Heap Go [41][42], Nim [8], Triangular Nim [3][14][33][47][57], Domineering [5][9][12][16][17][44], XT domineering [40][43], and NoGo [15][45]. If the positions of games are public to both players and their all available moves are also public, noted *perfect information* game [2]. Other games are called *imperfect information* game, like Bridge Poker, Mahjong [68][69] and so on. These games hide some information from other players.



1.1 Background

CGT is one of the applied mathematics which uses algebra characteristics to reduce the complexity of many intelligent games. Based on the theory, playing or solving many combinatorial games may simply become mathematical calculations, such as summation, instead of a complex tree search.

CGT studies two-player games with perfect information. The two players are assumed to take turns alternatively, and a game is considered as a sum of local positions, where each player can choose one local position to move at each turn.

1.2 Game tree search

The game search has been an important issue in computer science. From early minimax tree search to recently Monte Carlo Tree Search (MCTS), most of researches focused on improvement of tree search. In the following two subsections, we review minimax search and MCTS, respectively.

1.2.1 Minimax Tree

Figure 1 (below) shows a minimax tree. Square nodes usually represent Max nodes which get the maximum of the values of all children, and circular nodes represent Min nodes which get the minimum of the values of all children. Claude Shannon first estimated lower bound on the game-tree complexity of chess [58]. Recently, MCTS is a search paradigm that has been remarkably successful in computer games like Go. MCTS uses Monte-Carlo simulation to evaluate the values of nodes in a search tree. Currently MCTS is popular for game search method which also follows the original game-tree estimated.

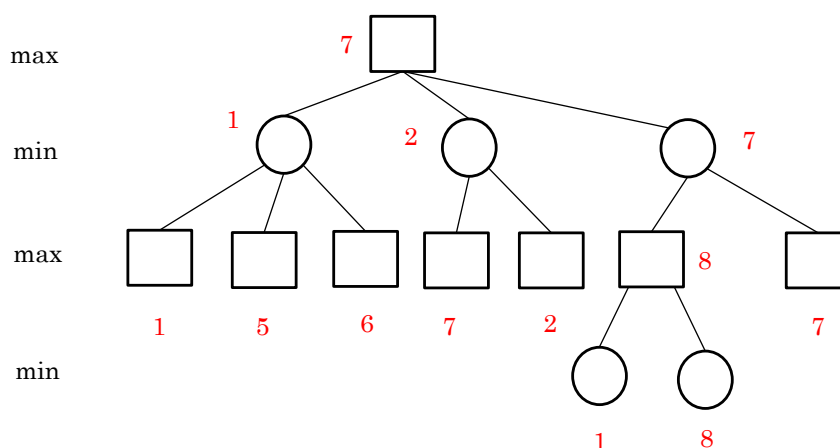


Figure 1. An example of minimax Tree.

1.2.2 Monte Carlo Tree Search

MCTS [21] is a search paradigm for two-player and zero-sum games. It has been applied to various computer games [4][11][25][29][48][56][61][62][66]. This subsection reviews the basic ideas of MCTS and its related policy improvements, following the definitions of [30].

Let S denote the state space of a game, and s_t be the state of a game at time t . Let A denote the space of actions and let $A(s)$ denotes the set of legal actions from state s . The two players alternate turns, at each turn t selecting an action a_t in $A(s_t)$. The game finishes upon reaching a terminal state with outcome z . One player's goal is to maximize z ; the other player's goal is to minimize z . A policy π is defined as a stochastic action selection strategy that determines the probability of selecting actions in any given state. $Q^\pi(s, a)$ is defined as the expected outcome after playing action a in state s , and then following policy π for both players until termination.


$$Q^\pi(s, a) = E_\pi[z \mid s_t = s, a_t = a] \quad (1)$$

The basic idea of MCTS is to evaluate the expected outcome online from simulated games. Each simulated game starts from a root state s_0 , and sequentially samples actions until the game terminates. At each step t of simulation, a simulation policy π is used to select an action a_t . The outcome z of each simulated game is used to update the Q -values encountered during that simulation. This update can be implemented incrementally by incrementing the state-action simulation count $N(s_t, a_t)$ and updating the Q -value towards the outcome z .

$$N(s_t, a_t) \leftarrow N(s_t, a_t) + 1 \quad (2)$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \frac{z - Q(s_t, a_t)}{N(s_t, a_t)} \quad (3)$$

There are two policy stages in the simulations. A tree policy is used to select actions on the state s_t represented in the search tree, while a default policy is used on those not in the tree, mainly for the playout simulations.

The second idea of MCTS is policy improvement. The values in the search tree are used as references to select the actions during subsequent simulations. As the number of simulations increases, the policy π continues to improve. Eventually, with sufficient simulations, the policy will reach the optimal strategy.

UCT [28] is an example of policy improvement scheme. It selects actions by using the UCB algorithm which maximizes an upper confidence bound on the value of actions. Specifically, the Q -value is augmented by an exploration bonus that is high for rarely visited actions, and the policy selects the action a^* maximizing the augmented value.

$$Q^{UCB}(s, a) = Q(s, a) + c \sqrt{\frac{\log N(s)}{N(s, a)}} \quad (4)$$

$$a^* = \arg \max_a Q^{UCB}(s, a) \quad (5)$$

where c is a scalar exploration constant, $N(s)$ is the number of visits to state s , and \log is the natural logarithm. The underlying idea of UCT is to provide a balance between exploitation of the current best action and exploration of other potential better actions.

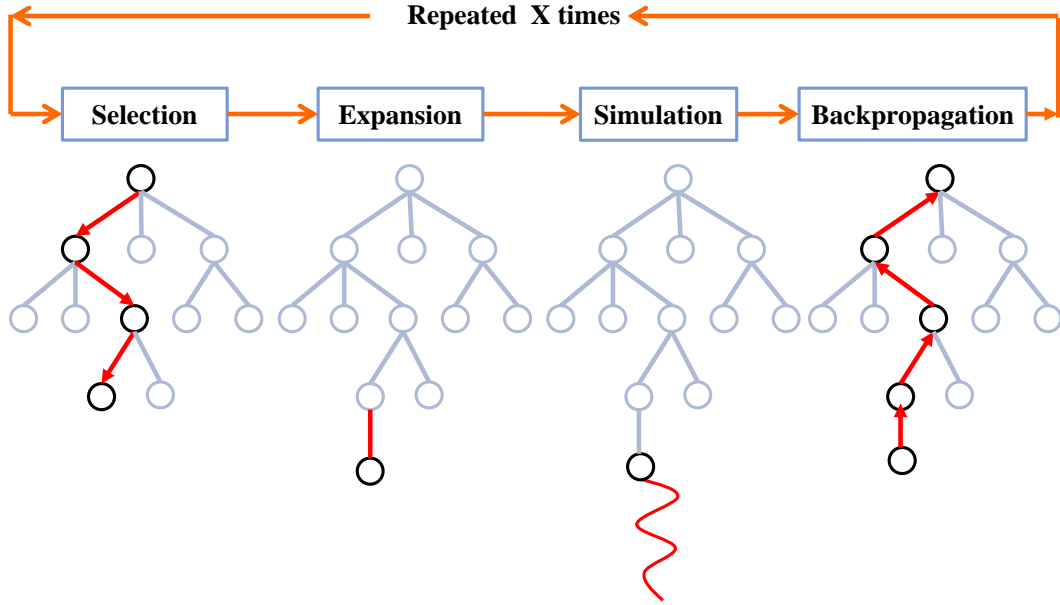


Figure 2. Monte Carlo Tree Search

Figure 2 shows four steps in MCTS including selection, expansion, simulation and back propagation in loop while.

1.3 Combinatorial Game Theory

Combinatorial game theory [7][20] starts from a simple definition of game: A game is an ordered pair of sets of games. Conventionally, a game G is denoted as:

$$G = \{G^L \mid G^R\}, \quad (6)$$

where G^L and G^R are sets of games. A special game is named 0, when both G^L and G^R are empty sets, \emptyset .

Negation, addition and comparisons are defined as follows.

$$-G = \{-G^R \mid -G^L\}. \quad (7)$$

$$G + H = \{G^L + H, G + H^L \mid G^R + H, G + H^R\} \quad (8)$$

$$G \geq 0, \text{ if and only if there is no element in } G^R \leq 0, \quad (9)$$

$$G \leq 0, \text{ if and only if } -G \geq 0, \quad (10)$$

$$G \geq H, \text{ if and only if } G - H \geq 0. \quad (11)$$

When neither $G \geq H$ nor $G \leq H$, it is said G *confused* with H , denoted by $G \parallel H$. $G <| H$ denotes either $G < H$ or $G \parallel H$, and similarly for $G >| H$. Furthermore, an equivalence relation on the sets of games is defined as follows.

$$G \equiv H, \text{ if and only if } G \geq H \text{ and } G \leq H. \quad (12)$$

The equivalence classes of games form an algebraic group, which can be used to describe the positions of many intelligent games as follows.

- There are two players (say Left and Right) move alternatively.
- The game is a sum of positions; each position has two sets of next positions; one for each player.
- On each player's turn, the player can choose one position and move the position to one of its next positions.
- The player who cannot find a move is the loser.

For each game G , there are 4 types of possible outcomes. The corresponding relations between G and 0 are described as below:

- $G \equiv 0$: The first player cannot win the game.
- $G < 0$: Left cannot win the game.
- $G > 0$: Right cannot win the game.
- $G \parallel 0$: The first player can win the game.

In general, players are concerned with who can win a given game G . Mathematically speaking, the question is equivalent to determining one of the above four relations between G and 0.

1.4 Framework of Research

In this thesis, we study to solve games and improve search performances with embedded combinatorial game knowledge. For this study, we investigate three combinatorial games, including Triangular Nim, XT Domineering and NoGo. From the viewpoint of CGT, Triangular Nim and XT Domineering belong to nimbers and infinitesimals, respectively. But the third game NoGo, which was introduced in 2011, contains numbers, infinitesimals and hot games. Figure 3 shows the framework of research in this thesis.

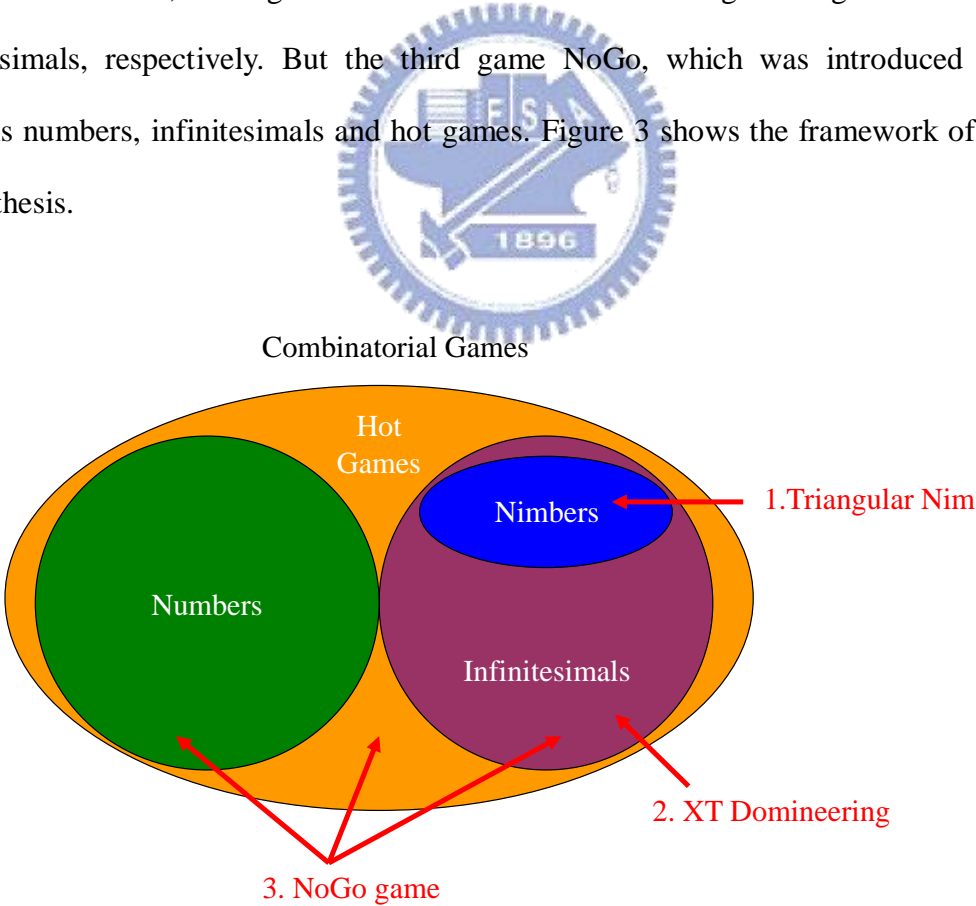


Figure 3. The framework of research

1.5 Organization

Chapter 2 presents the solving of nine layer Triangular Nim. We propose some methods to improve the performance, such as designing data structures in blocks, using the retrograde method, removing redundancy and selecting the block representation with the less number of inter-block updates.

Chapter 3 introduces a new game: XT domineering. We present a mathematical approach to solve sums of 3×3 XT Domineering and find several infinitesimal games with interesting properties. Chapter 4 is NoGo game analysis and derives three important propositions. Chapter 5 concludes this thesis.



Chapter 2 Triangular Nim

This chapter introduces the game of Triangular Nim. We propose a solving approach to strong solve the nine layer Triangular Nim and use isomorphism of Triangular Nim to reduce storage requirement and improve the performance.

2.1 Introduction

Triangular Nim [10][65], which is a common two-player game in Taiwan and China, is one variant of the game Nim. Nim is a two-player mathematical game of strategy in which players take turns removing pieces from distinct heaps, each containing a set of pieces. Two players alternatively remove one or more pieces from the same heaps. The game won by the player who removes the last piece is called a normal play game. In contrast, the game won by the other is called a misère play game, or simply a misère game [7][20].

Many interesting and useful theories were developed for Nim in [7][8][20]. Obviously, Nim games are never drawn. Nim games are also called impartial games since from all positions¹ the moves available to move by either player are exactly the same. Sprague [60] and Grundy [31] also gave some useful theoretical analysis, such as Sprague–Grundy theorem.

Since Nim games are never drawn, all positions are either winning or losing to the player to move. The positions that the player is to move wins are commonly called N-positions, while the others are called P-positions [7]. The positions that have no

¹ In this thesis, *positions* for Triangular Nim include the information of the remaining pieces, but exclude the information of the player turn.

subsequent moves are called terminal positions. Apparently, all terminal positions are P-positions in the normal play game, while they are N-positions in the misère play game. In addition, it is clear to see the following two rules: [24]

For each N-position, there is at least one move to a P-position.

For each P-position, every move is to an N-position.

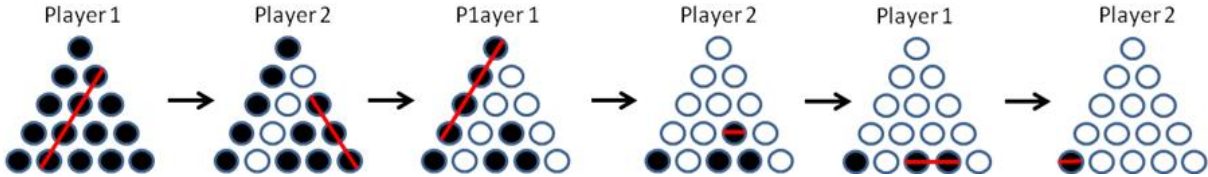


Figure 4. A scenario of Triangular Nim with side size five.

In the game of Triangular Nim, all pieces are placed in an equilateral triangle as shown in Figure 4. Two players alternatively remove one or more consecutive pieces from one row or diagonal in the triangle of pieces. Triangular Nim shares some properties of Nim. For example, the games are never drawn; and the games are also impartial games. Similarly, the game won by the player who removes the last piece is called a normal Triangular Nim, while the game won by the other is called a misère Triangular Nim. Illustrated in Figure 4, player 2 who removes the last piece loses the game in the misère. The version commonly played in Taiwan and China is the misère Triangular Nim.

A Triangular Nim is called a k layer Triangular Nim, if the equilateral triangle is of side size k . For example, Figure 4 shows a 5 layer Triangular Nim which has 15 pieces in total. The position with all the 15 pieces is the initial position of a 5 layer Triangular Nim.

In the past, Hsu [33] solved 7 layer Triangular Nim by a brute force approach, called a *backward induction approach* in [24]. This approach tends to verify exhaustively all game positions to win or lose, from those with fewer pieces to more pieces. For the position

without any pieces, set it to P-position or N-position depending on normal or misère, initially. Then, evaluate positions from those with fewer pieces to those with more pieces. For each position P to be evaluated, set it to P-position or N-position from all legal moves, according to the above two rules. Since all the moves will make the number of pieces fewer, the new position, say P' , must have been evaluated earlier than P . Thus, we can quickly derive the result of P from all P' . The method checking all legal moves for each position is called *forward checking method* in this thesis.

Recently, both research teams [3] and [47] solved 8 layer Triangular Nim by different methods independently. The researchers in [3] used the forward checking method by Hsu, while the researchers in [47] used a faster method to improve the performance. This method is a kind of *retrograde methods*.

In the past, the retrograde methods were successfully used by researchers in [34][63][64][70]. In retrograde methods, whenever the game-theoretical value (or just value) of a position, the win/loss status of a position, is derived to be a loss, we update the values of all its parents to be wins. Note that this thesis follows the definitions of the terms in [35], such as strongly solved and game-theoretical values of positions.

The retrograde methods were shown to be very efficient, since the loss ratio (the number of losing positions over the number of all positions) is usually low. Thus, the update operations are not done often. In this chapter, we will also show that the loss ratio is very low in Triangular Nim.

This thesis follows, in principle, the retrograde method used in [47] to strongly solve 9 layer Triangular Nim and obtains the results that the first player wins in both normal and misère. However, strongly solving 9 layer Triangular Nim requires huge amount of memory. Our first version with the retrograde method requires 4 terabytes of data in memory. Since it is impossible to store all the data into RAM in most computer systems, the hard disks are

used. Hence, it also increases much more computation time to solve.

For this problem, this section proposes the block representation to decrease I/O frequencies and allow parallel processing to speed up the performance. Thus, it takes about 32.31 days on a machine with four cores and 129.21 days on one core aggregately.

Furthermore, we also improve the performance by making use of the characteristics of rotation and mirroring. The second version with the improvement efficiently reduces the memory by a factor of 5.72 and the computation time by a factor of 4.62.

2.2 Related Work

This section reviews the research for Triangular Nim in the past. Subsection 2.2.1 introduces the backward induction approach and described the forward checking method used in [3][33]. Subsection 2.2.2 described the retrograde method used in [47].

2.2.1 Backward Induction Approach

As described above, the backward induction approach is to evaluate the values of all game positions, from those with fewer pieces to more pieces. Since either player wins in Triangular Nim, the value of a given position is either a win or a loss, and therefore can be represented by one bit of data. Without loss of generality, let 1 indicate an N-position where the player is to move win; and let 0 indicate a P-position, otherwise.

For k layer Triangular Nim, since it has $k(k+1)/2$ pieces, the total number of positions is $2^{k(k+1)/2}$. Thus, the space complexity is $O(2^{k(k+1)/2})$. From above, we can use $2^{k(k+1)/2}$ bits to represent the values of all the positions. For example, 5 layer Triangular Nim requires 2^{15} bits to represent all the values, while 9 layer Triangular Nim requires 2^{45} bits, equivalent to 2^{42} bytes, about 4 terabytes.

In order to identify the value of a given position in all the $2^{k(k+1)/2}$ bits, Hsu [33] simply used a $k(k+1)/2$ -bit binary number as a *position identifier*. In the position identifier, each bit indicates whether a corresponding piece exists. The corresponding pieces to bits are designated by a *piece mapping* in advance. For example, in Figure 5, a piece mapping is shown, and a position of 5 layer Triangular Nim is identified as 011010111001011, equivalent to 13,771 in decimal, by a piece mapping. The initial position of 5 layer Triangular Nim is 111111111111111, all 1s. The initial positions for other layer Triangular Nim are all 1s, too.

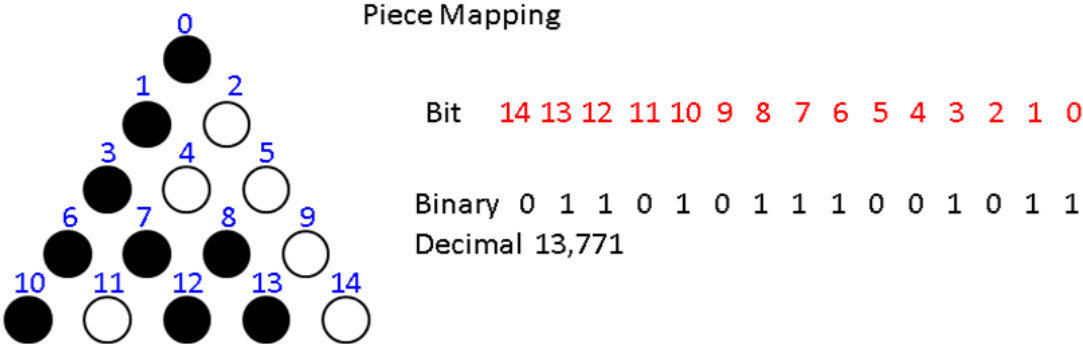


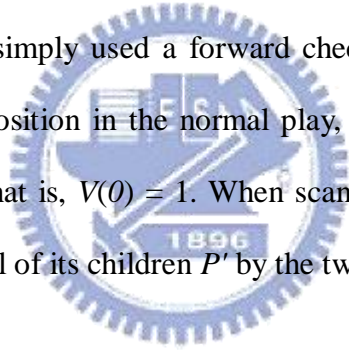
Figure 5. A piece mapping of 5 layer Triangular Nim.

For a piece mapping, a legal move, removing some consecutive pieces, can also be represented by a $k(k+1)/2$ -bit binary number corresponding to these removed pieces. For example, for the piece mapping in Figure 5, 000000000001011 is a legal move that removes the top three pieces at 0, 1 and 3. Since a legal move must remove consecutive pieces in a row or diagonal, the move at 1, 3 and 7 and the one at 1 and 8 are illegal. In [33], Hsu derived that the total number of moves is $k^2(k+1)/2$ in k layer Triangular Nim. Hence, the number of legal moves is 75 for 5 layer Triangular Nim, while the number is 405 for 9 layer Triangular Nim. Thus, the time complexity for k layer Triangular Nim is roughly $O(k^3 2^{k(k+1)/2})$.

For a given position P , assume that a move M is legal in P . The bits with 1 in M must be 1 in P , too. We can use the following simple bit-wise operation to detect whether M is a legal move in P : The bit-wise AND operation on P and M is still M . In addition, we can also easily derive the position after the move M , by using bit-wise operations to remove all 1s of M from P . Namely, the new position P' is $P \oplus M$, where \oplus is a bit-wise exclusive OR operation. For simplicity of discussion, P' is called a child of P , and P is the parent of P' .

For Triangular Nim, the backward induction approach is to derive $V(P)$ from $P = 0$ to $2^{k(k+1)/2} - 1$. $V(P)$ denotes the value of a given position P . An important property is: The values $V(P')$ for all children P' of P must have been evaluated, before the value $V(P)$ is evaluated.

In [3][33] the researchers simply used a forward checking method to evaluate $V(P)$. Initially, the position 0 is a P-position in the normal play, that is, $V(0) = 0$, while it is an N-position in the misère play, that is, $V(0) = 1$. When scanning to the position P , evaluate $V(P)$ from the values $V(P')$ for all of its children P' by the two rules in Section 2.1.



2.2.2 Retrograde Method

The research in [47] used a more efficient method, a kind of retrograde method, which is also used in this thesis. Initially, the value $V(0)$ is initialized as the forward checking method. In addition, all the other $V(P)$ are initialized to 0. Then, the retrograde method also follows the backward induction approach to visit position P from position 0 to $2^{k(k+1)/2} - 1$ and perform the following update operation when visiting P .

- If $V(P) = 0$, then set $V(P_{par}) = 1$ for all of its parents P_{par} .

By induction, we can show that the above operation satisfies the following property: When position P is visited, the value $V(P)$ indicates the correct value. Assume that the properties are satisfied for all of its children. Then, if some child P' is a P-position, $V(P)$ is set to 1 when visiting P' based on the above operation. This is correct since the position P is indeed an N-position based on the first rule in Section 2.1. If all children are N-positions, $V(P)$ remains 0. This is also correct since P is indeed a P-position based on the second rule. Thus, it is concluded that the above property is satisfied.

The retrograde method is very efficient for the following reason. The above update operation is done only when the position P is a P-position (a loss to the next player to move). Most importantly, the loss rate, the number of P-positions to that of all positions, is normally low. The loss rate is only 6.0% for 8 layer Triangular Nim according to [47], while the loss rate is only 5.0% for 9 layer Triangular Nim according to the experiments in Section 2.4. That is, the computation times can be reduced by a factor of 20 or so.

2.3 Solving Approach

This section presents our approach to strongly solve 9 layer Triangular Nim. In our approach, we use the retrograde method and also propose some new methods for efficient data structures and removing redundancies, respectively described in Subsections 2.3.1 and 2.3.2.

2.3.1 Data Structures and Algorithms

Since 9 layer Triangular Nim requires a huge amount of memory (4 terabytes for the design described in Subsection 2.2.1), the memory space must be cut into disjoint blocks to make it feasible to process. To facilitate identifying the blocks, we let the most significant

bits of position identifiers be the block identifier. For simplicity of discussion, the block representation in 8 layer Triangular Nim is illustrated in the rest of this subsection.

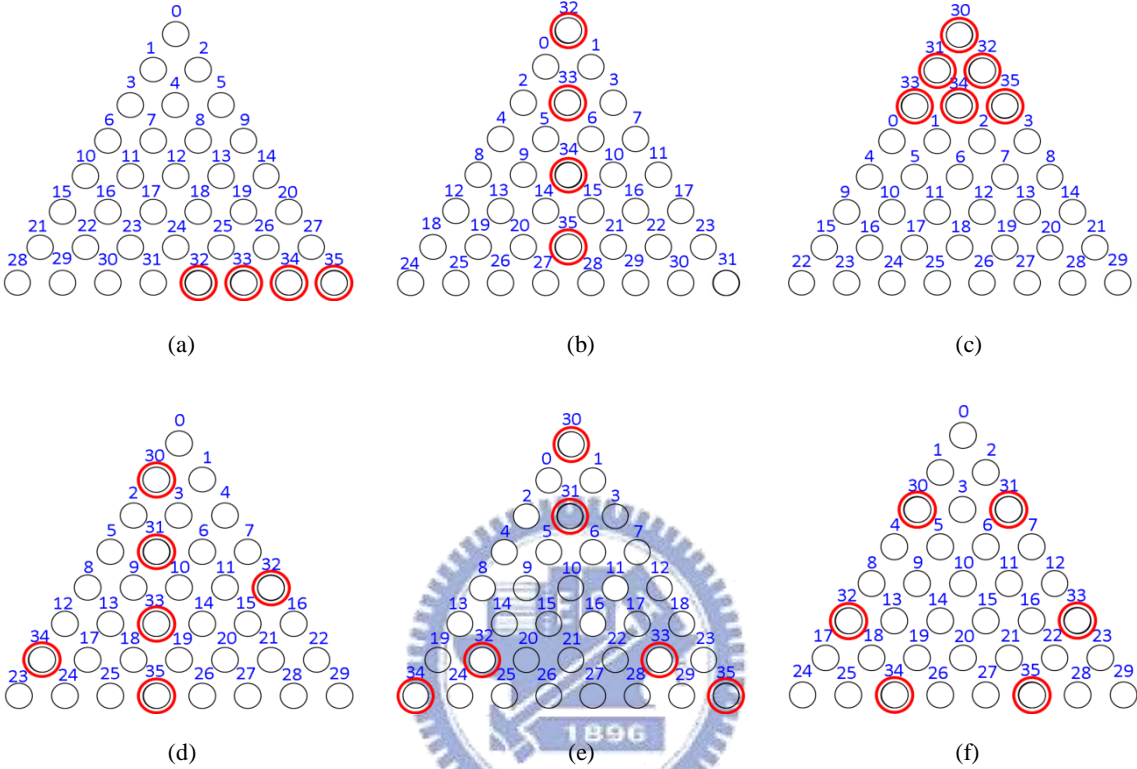


Figure 6. Six block representations for 8 layer Triangular Nim.

As described in Subsection 2.2.1, 8 layer Triangular Nim requires 2^{36} bits, equivalent to 2^{33} bytes or 8 gigabytes. Assume to cut the memory space into 16 blocks, each with 512 megabytes. Then, the most four significant bits in the position identifier are used to indicate the *block identifier* (*Block ID* or *BID*), 0 to 15. The 16 blocks are denoted by B_0, \dots, B_{15} . From the piece mapping described in Subsection 2.2.1, four pieces are designated as the bits of block identifier. The four pieces are called *Block-ID pieces* or *BID pieces*. For example, two such block representations using different BID pieces, marked with double cycles, are shown in Figure 6 (a) and (b). Other block representations using six BID pieces are shown in the rest of Figure 6.

Following the backward induction approach (in Subsection 2.2.1), we visit positions from B_0 to B_{15} . Now, the next question is how to evaluate each block. Let us still follow the approach as the retrograde method. We need to update the values in B_{i+1}, \dots, B_{15} , when visiting B_i . However, since the system memory (like RAM) may not be able to include all blocks, we use the following operations when visiting the block B_i .

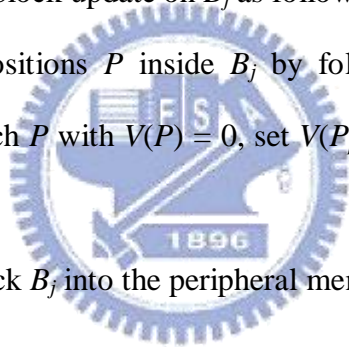
1. Load the block B_i (into the system memory).
2. Evaluate the values of B_i as follows.
 - a. Evaluate all positions P inside B_i by following the retrograde method. Namely, for each P with $V(P) = 0$, set $V(P_{par}) = 1$ for all of its parents P_{par} inside B_i .
 - b. Restore the block B_i into the peripheral memory like hard disks.
3. For all $j > i$, repeatedly perform the following.
 - a. Load the block B_j .
 - b. For each position P in B_i with $V(P) = 0$, set $V(P_{par}) = 1$ for all of its parents P_{par} in B_j .
 - c. Restore the block B_j .

The above operations can be separated into different jobs. The operations 1 and 2 together, called *intra-block update* on B_i , is to update B_i itself. The operations 1 and 3 together for each j , called *inter-block update* from B_i to B_j , is to update B_j from B_i . For inter-block updates from B_i to B_j , the block B_j is called a parent of B_i , or B_i is the child of B_j .

The above operations can be further improved by saving the restore operations in the inter-block updates (at Step 3.c), if we use a forward-checking method in the block level,

instead of a retrograde method in the block level. Namely, all inter-block updates from B_i to B_j are done when visiting block B_j , not when visiting block B_i . The improved algorithm is modified as follows.

1. Initialize the block B_j .
2. For all $i = 0$ to $j-1$, repeatedly perform the following inter-block updates.
 - a. Load the block B_i .
 - b. For each position P in B_i with $V(P) = 0$, set $V(P_{par}) = 1$ for all of its parents P_{par} in B_j .
3. Process the intra-block update on B_j as follows.
 - a. Evaluate all positions P inside B_j by following the retrograde method. Namely, for each P with $V(P) = 0$, set $V(P_{par}) = 1$ for all of its parents P_{par} inside B_j .
 - b. Restore the block B_j into the peripheral memory like hard disks.



The above operation for 8 layer Triangular Nim requires 16 intra-block updates and 120 ($=15 \times 16 / 2$) inter-block updates. In fact, the number of inter-block updates can be far below the above number. For example, for the block representation in Figure 6(a), no operations are required for the inter-block update from B_0 to B_5 , since it is impossible to have a move remove pieces 32 and 34 without removing 33.

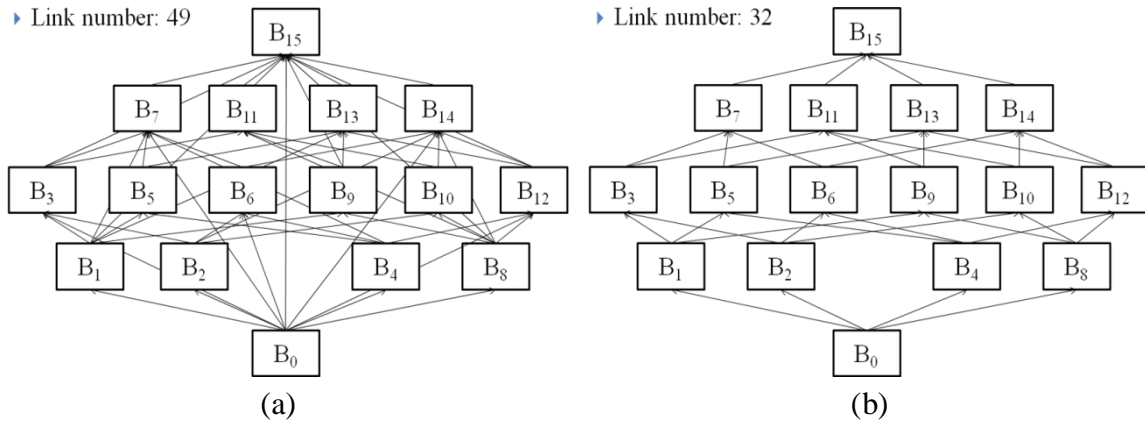


Figure 7. Inter-block updates of the block representations in Figure 6 (a) and (b), respectively.

The number of required inter-block updates is usually small if the BID pieces are distributed sparsely, and large if the BID pieces are grouped. After our analysis, the number of required inter-block updates is 49 for the one in Figure 6(a), and 32 for the one in Figure 6(b). Figure 7 shows the inter-block updates for the two block representations, respectively. Usually, the smaller number of required inter-block updates is, the better performance is. This is also shown in our experiment in Section 2.4.

2.3.2 Removing Redundancy

Due to symmetry and rotation, we do not have to evaluate all the positions. However, it is hard to save space for all symmetric and rotated positions directly, since it is hard to identify saved bits in our data structure shown in Subsection 2.3.1. Fortunately, it becomes easier to save space on blocks.

Let us illustrate the block representation with six BID pieces shown in Figure 6(f) and the blocks, B_1 , B_2 , B_4 , B_8 , B_{16} and B_{32} , shown in Figure 8 (below). Due to symmetry and rotation, all the position values in one block, say B_2 , can be found from those in another, say

B_1 , by finding the corresponding mirrored or rotated position values, and *vice versa*. Here, B_1 is said to be an isomorphic block of B_2 , and *vice versa*.

All of the six blocks shown in Figure 8 are actually isomorphic. Thus, it is sufficient to evaluate the values of one block, say B_1 , only. And, we can get the values of other blocks from the values of B_1 . These blocks form an *isomorphic group of blocks*.

In an isomorphic group of blocks, we can simply choose to evaluate the block with smallest BID for simplicity. For example, in the group shown in Figure 8, we choose B_1 to evaluate. For the improved algorithm in the previous subsection, we add the following rules.

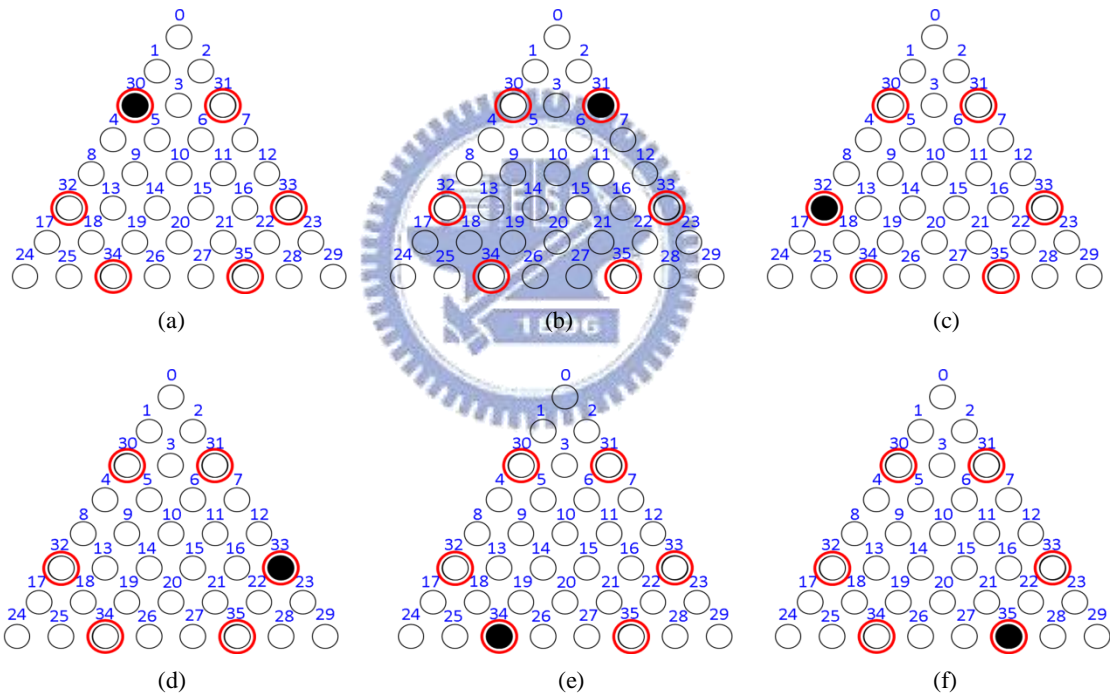


Figure 8. Six rotated and mirrored blocks.

1. When visiting Block B_i , skip it if there exists another isomorphic block with smaller BID.
2. When making an inter-block update from B_i to B_j , access $B_{i'}$ instead of B_i , if $B_{i'}$ is the block with smallest BID among all the isomorphic blocks of B_i .

In the second rule, when accessing block B_i , the block $B_{i'}$ (with the smallest BID) must be available, since we evaluate blocks from the lower BID to higher. In the rule, when accessing positions in B_i , access the corresponding positions in $B_{i'}$ according to the directions of rotation and mirroring.

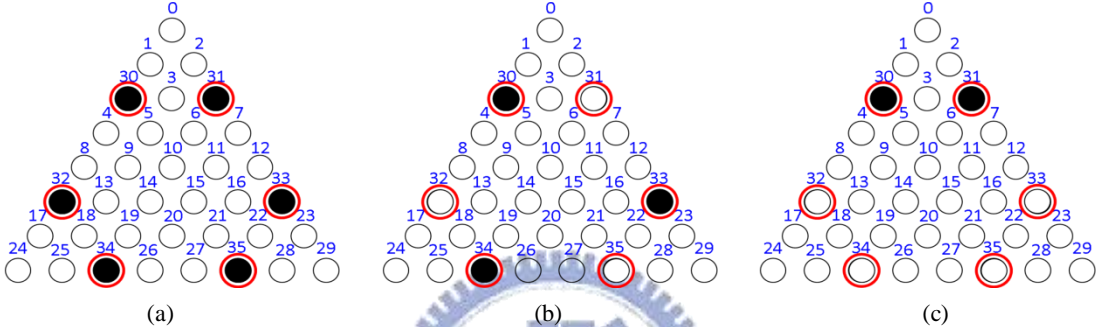


Figure 9. An isomorphic group with one, two and three distinct blocks only.

An isomorphic group may contain one, two, three, six distinct blocks, as illustrated in Figure 9(a), Figure 9(b), Figure 9(c), and Figure 8(a), respectively. Since an isomorphic group contains at most six distinct blocks, we can reduce the space by a factor of at most six. For the block representation (in both Figure 8 and Figure 9), the number of isomorphic groups with one distinct block is 2, that with two is 1, that with three is 6, and that with six is 7. Thus, for this block representation, we need to evaluate 16 blocks, among all the 64 blocks. Thus, we can save the space by a factor of four in this case.

The space saving rate grows higher and approximates to 6, if the block representation is well designed with large number of BID pieces. For example, for the block representation for 9 layer Triangular Nim shown in Figure 13(b), the number of isomorphic groups with one distinct block is 16, that with two is 8, that with three is 496, and that with six is 5,208.

Thus, for this block representation, we need to evaluate 5,728 blocks only, among all the 32,768 blocks. The space is reduced by a factor of 5.72, approximate to 6. More specifically, we can reduce the space from 4 terabytes to 716 gigabytes.

2.4 Experimental results

In this section, all experiments were done on a 4 cores personal computer equipped with Intel® Core™ 2 Quad CPU Q6600, 2.4GHz CPU, 8 gigabytes RAM and 6 terabytes hard disks with RAID, which had been used in the desk-top system [67]. We used a simple parallel method to exploiting the parallelism of the four cores as follows. Whenever a core is idle, we choose a block B_i which has not been evaluated yet and whose children have been evaluated. However, since parallelism is not the goal of our destination, we used the total times aggregated from the four cores.

Since it takes a huge amount of computation time for solving 9 layer Triangular Nim, we first investigate 8 layer Triangular Nim in Subsection 2.4.1. Then, based on the experiences for 8 layer Triangular Nim, we solve 9 layer Triangular Nim in Subsection 2.4.2.

2.4.1 Eight Layer Triangular Nim

This subsection investigates how to solve 8 layer misère Triangular Nim. We used the six block representations in Figure 6 as our testing cases. We used (a), (b), (c), (d), (e) and (f) to indicate the six representations respectively.

Table 1 shows the experimental results, using the method without removing redundancy as described in Subsection 2.3.1. Without removing redundancy, the total memory space requires 8 gigabytes for all cases. Table 1 clearly shows that the block

representations with the less numbers of inter-block updates are more efficient, since the overhead for inter-block updates becomes smaller. Due to the overhead for inter-block updates, the block representations with larger block sizes tend to run faster. On the other hand, the block size cannot exceed the physical or virtual memory size of the operating systems. Thus, this is a tradeoff for the block size.

Table 2 shows the experimental results with removing redundancy. This table does not include the cases (a), (b) and (d), since the results are the same as Table 1. Both cases (a) and (d) are not symmetric and rotatable. Although the case (b) is symmetric, all the isomorphic groups contain one block only, and therefore the results are the same as Table 1.

In Table 2, both cases (e) and (f) perform apparently more efficiently than the case (c), since the case (c) does not remove the redundancy caused by rotation. The case (f) performs more efficiently than the case (e), since both the number of blocks and the number of inter-block updates are smaller.

From the above experiments, we observe that the most important factor for high performance is to remove redundancy, and then to reduce the number of blocks or the number of inter-block updates. From the observation, we solve 9 layer Triangular Nim.

	Inter-block Number	Block Number	Block Size	Total Space	Aggregated Time
a	49	16	512MB	8GB	8,992 sec
b	32	16	512MB	8GB	8,472 sec
c	360	64	128MB	8GB	13,453 sec
d	208	64	128MB	8GB	11,042 sec
e	288	64	128MB	8GB	12,339 sec
f	336	64	128MB	8GB	13,128 sec

Table 1. Experimental results for the block representations without removing redundancy in Figure 6.

	Inter-block Number	Block Number	Block Size	Total Space	Aggregated Time
c	228	40	128MB	5GB	8,423 sec
e	96	20	128MB	2.5GB	4,161 sec
f	87	16	128MB	2GB	3,705 sec

Table 2. Experimental results for three block representations with removing redundancy in Figure 6.

From the above experiments, we strongly solved all positions of 8 layer Triangular Nim with both normal and misère play. The results show that both initial positions for both normal and misère play are N-positions. The first player wins both by making the moves shown in Figure 10 for the normal play and those in Figure 11 for the misère play.

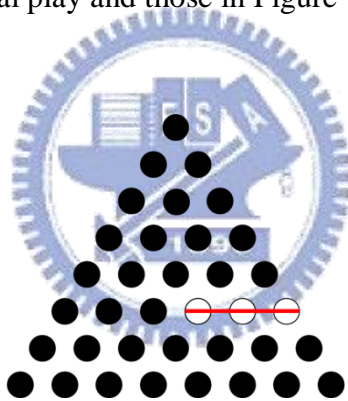


Figure 10. The moves to win in 8 layer normal Triangular Nim.

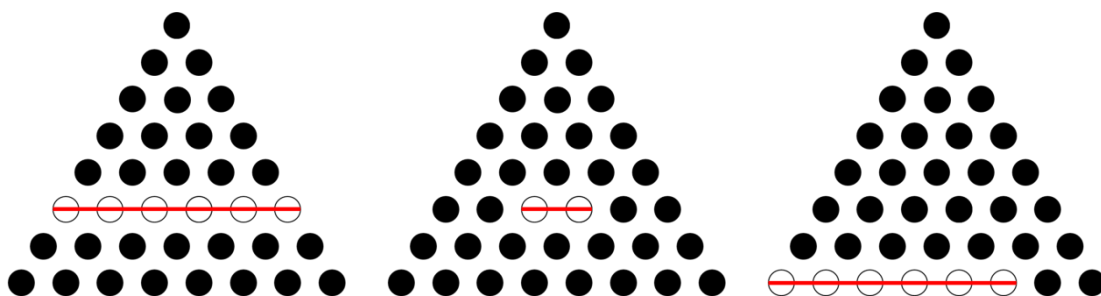


Figure 11. The moves to win in 8 layer misère Triangular Nim.

2.4.2 Nine Layer Triangular Nim

In this subsection, we first investigated the block representations with 13 BID pieces, three of them shown in Figure 12 (below). Although each block requires a large block size, 512 megabytes, these BID pieces cannot form symmetry or rotation so that we cannot remove redundant blocks as described in Subsection 2.3.2. Among the three in Figure 12, the one in Figure 12(c) has the least number of inter-block updates. Unfortunately, since it is still not symmetric and rotatable, it took 32.31 days on four cores (129.21 days on one core) to finish the whole computation in Table 3.

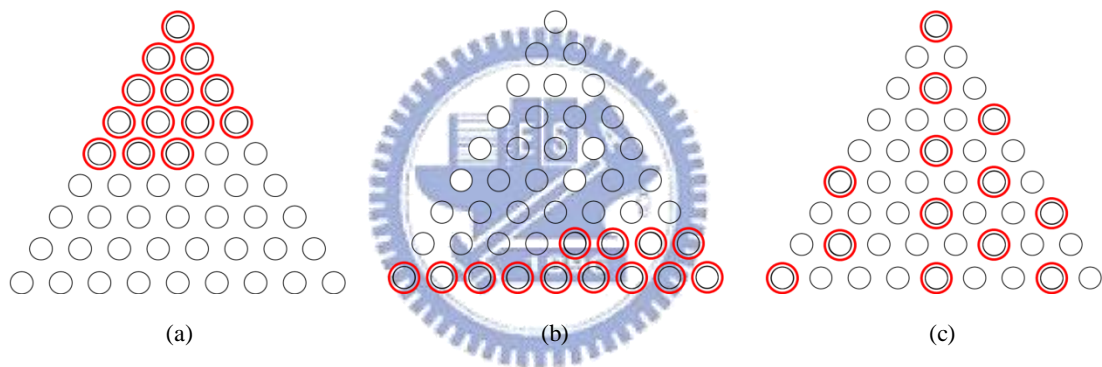


Figure 12. Three block representations with 13 BID pieces.

	Inter-block Number	Blocks Number	Block Size	Total Space	Parallel Computation Time	Aggregated Time
a	121,600	8,192	512MB	4TB	N/A	N/A
b	107,024	8,192	512MB	4TB	N/A	N/A
c	88,064	8,192	512MB	4TB	32.31 days	129.21 days

Table 3. Experimental results for the block representations in Figure 12.

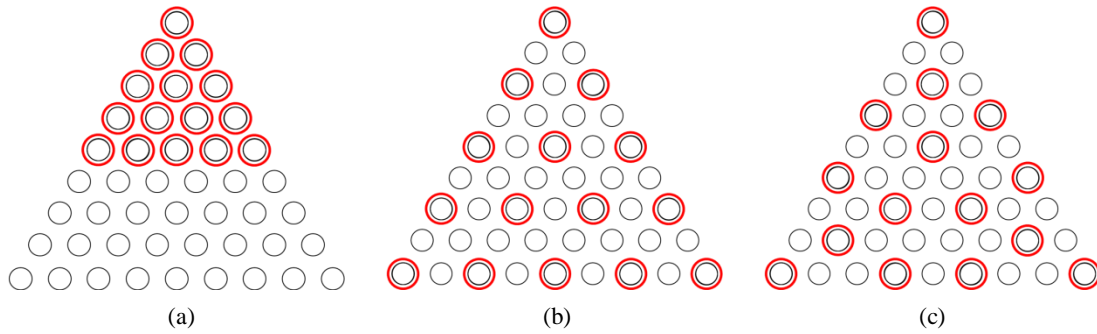


Figure 13. Three block representations with 15 BID pieces.

	Inter-block Number	Blocks Number	Block Size	Total Space	Parallel Computation Time	Aggregated Time
a	298,320	16,640	128MB	2,080GB	N/A	N/A
b	102,950	5,728	128MB	716GB	8.32 days	33.23 days
c	78,852	5,728	128MB	716GB	6.99 days	27.93 days

Table 4. Experimental results for the block representations in Figure 13.

This subsection also investigates the block representations with 15 BID pieces, three of them shown in Figure 13. Apparently, the one in Figure 13(a) is symmetric, while the other two are symmetric and rotatable. Due to symmetry and rotation, the number of blocks can be reduced by a factor of near 2 for the first and near 6 for the other two. The results showed that the computation times for the second and third are reduced to 8.32 days and 6.99 days as shown in Table 4, respectively. The third is slightly better than the second, since the number of inter-block updates is smaller as shown in Table 4. In brief, from the one in Figure 12(c) to that in Figure 13(c), we reduce the memory by a factor of 5.72 and the computation time by a factor of 4.62.

From the above experiments, we strongly solved all positions of 9 layer Triangular Nim in both normal and misère play. The results also show that both initial positions in both normal and misère play are N-positions. The first player wins both by making the moves

shown in Figure 14 for the normal play and those in Figure 15 for the misère play.

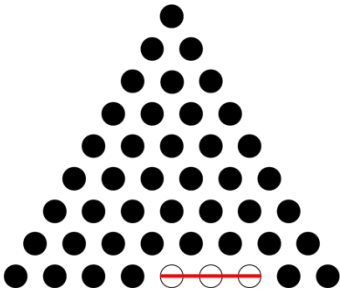


Figure 14. The moves to win in 9 layer normal Triangular Nim.

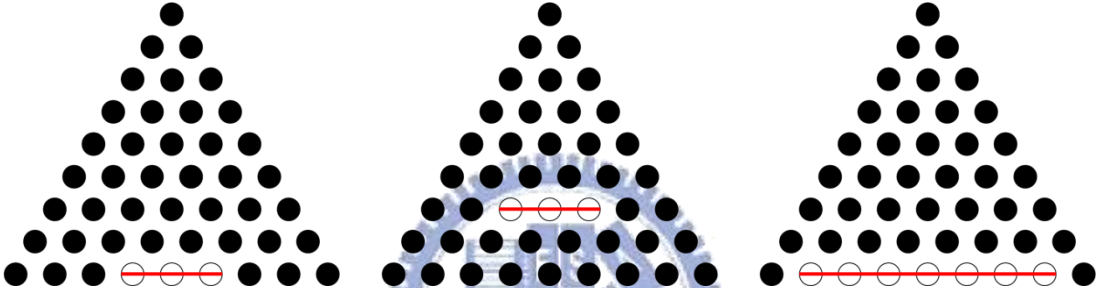


Figure 15. The moves to win in 9 layer misère Triangular Nim.

Table 5 lists all the values of the initial positions of k layer Triangular Nim in both normal and misère, where k is 1 to 9. This table shows that the first player tends to win or that the initial positions tend to be N-positions. This is because the ratio of the number of P-positions to that of N-positions tends to be low, that is, most positions are N-positions.

We also analyzed the ratio in both normal and misère in Figure 16 for all k layer Triangular Nim, where k is 1 to 9. From the Figure 16, the ratios get smaller for large k . For 9 layer Triangular Nim, the ratio is 5.0% only. This explains why most of initial positions are N-positions. In addition, since most positions are N-positions, this shows that the retrograde method is clearly superior.

k	k Layer Normal Triangular Nim	k Layer Misère Triangular Nim
1	Win	Loss
2	Loss	Win
3	Win	Loss
4	Win	Win
5	Win	Loss
6	Win	Win
7	Win	Win
8	Win	Win
9	Win	Win

Table 5. The result of k layer Triangular Nim.

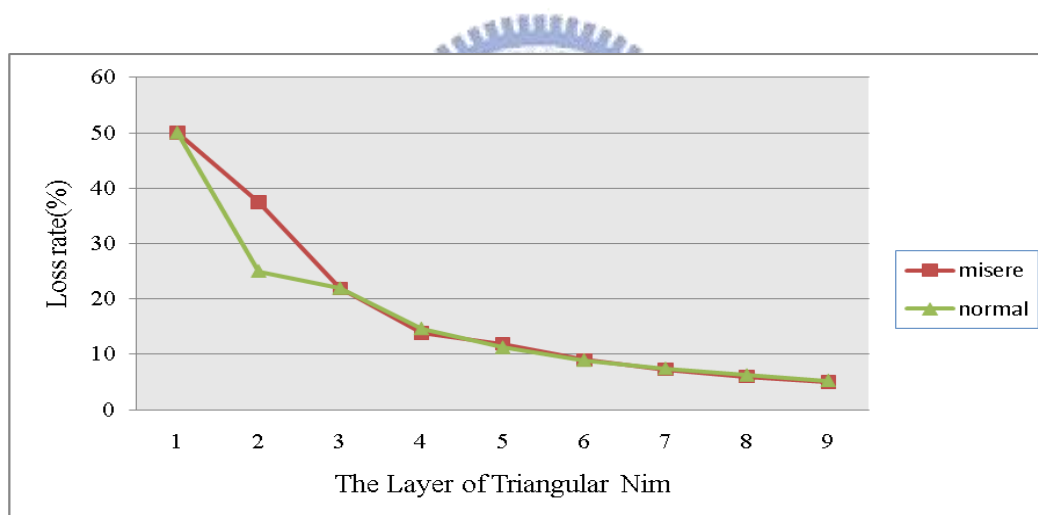


Figure 16. The ratio of the number of P-positions to that of N-positions.

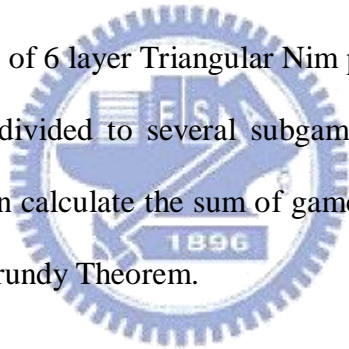
2.5 Conclusion

We summary our contributions in this chapter as following lists.

- Using the retrograde methods, this thesis strongly solves 9 layer Triangular Nim.

- This thesis also proposes some methods to improve the performance, such as designing data structures in blocks, using the retrograde method, removing redundancy and selecting the block representation with the less number of inter-block updates. Especially, by removing redundancy, we reduce the memory by a factor of 5.72 and the computation time by a factor of 4.62 using personal computer equipped with Intel® Core™ 2 Quad CPU Q6600, 2.4GHz CPU, 8 gigabytes RAM.
- Our experiment result also shows that the ratio of the number of P-positions to that of N-positions is low, 5.0% for 9 layer Triangular Nim. Due to the low ratio, the retrograde method does perform well when compared with the traditional forward checking.

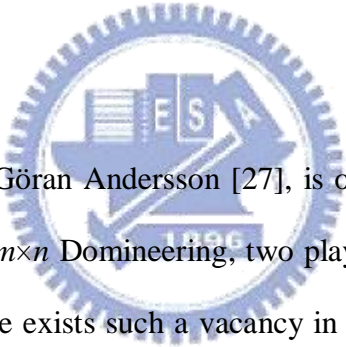
We also got all game values of 6 layer Triangular Nim positions in normal play. During the game play, if the board is divided to several subgames which are all under 6 layer Triangular Nim positions, we can calculate the sum of game values of subgames to quickly judge who to win by Sprague–Grundy Theorem.



Chapter 3 XT Domineering: A New Combinatorial Game

This chapter introduces a new combinatorial game of XT Domineering. We use combinatorial game theory (CGT) to calculate the game values of all sub-graphs of 3×3 squares and show that each sub-graph of 3×3 squares is a linear combination of 8 elementary infinitesimals.

3.1 Introduction



Domineering, designed by Göran Andersson [27], is one of combinatorial games that were based on the model. In an $m \times n$ Domineering, two players alternatively place 1×2 and 2×1 domino at a position, if there exists such a vacancy in a board with $m \times n$ squares. One player is allowed to place 1×2 domino only, while the other is 2×1 domino only. The one who cannot place domino loses.

In the past, many Domineering problems were solved. The general Domineering problem of $2 \times n$ board for all odd n was solved by Berlekamp [5]. The researchers in [9] used the technique of transposition tables to solve the 8×8 board. Subsequently, the researchers in [44] found out the results for boards of width 2, 3, 5, and 7 and some specific cases. Recently, Bullock solved the 10×10 board Domineering [12]. Furthermore, Cincotti developed three players Domineering on a three dimensional board [16][17].

This chapter introduces a new game named *XT Domineering* (named from *eXTended Domineering*). *XT Domineering*, modified from the *Domineering* game, allows players to place a 1×1 domino on an empty square s while unable to place a 1×2 or 2×1 domino in the connected group of empty squares that includes s . A connected group of empty squares is called an *active* group. After modifying the rule, players are allowed to place 1×1 domino on any square of an active group on which players are not allowed to place any dominos in the original *Domineering* game. For example, in *XT Domineering*, all 1×1 isolated vacancies in the board are allowed to be placed by more dominos. Thus, the move lengths in the new game are normally longer than those in *Domineering*. Thus, the game has higher game-tree complexity, based on the definition in [35].

This chapter also introduces the mathematical analysis of *XT Domineering*. In *XT Domineering*, each game position is actually an infinitesimal (as described in Section 3.4). In this chapter, we study several interesting infinitesimals in *XT Domineering*. This chapter calculates the game values of all sub-graphs of 3×3 squares and presents a rule to determine the outcome of any sum of these positions.

Section 3.2 reviews the combinational games including three subgroups of games. Section 3.3 reviews the game *Domineering* and introduces the new game, *XT Domineering*. Section 3.4 derives the game values of 3×3 *XT Domineering*, while Section 3.5 derives the outcomes of sums of 3×3 *XT Domineering*. Section 3.6 concludes this chapter.

3.2 Combinatorial Games

CGT [7] starts from a simple definition of game: A game is an ordered pair of sets of games. Since we are dealing with equivalence classes, for simplicity, we shall use the symbol $=$ to replace \equiv in the following context.

There are several subgroups of combinatorial games whose addition and outcome properties are well-studied. Some of them are reviewed in the following subsections.

3.2.1 Numbers

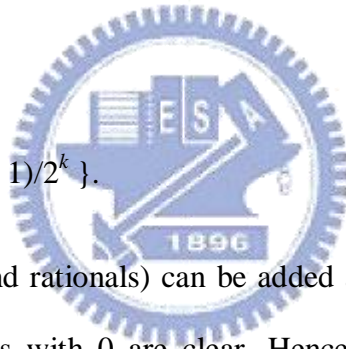
A game G is called a *number* [7][20] if all the elements in G^L and G^R are numbers and there is no element in G^L greater than or equal to any element in G^R . Some numbers are illustrated as follows:

$$1 = \{0 \mid \emptyset\}, \dots$$

$$n = \{n-1 \mid \emptyset\}, \dots \tag{13}$$

$$1/2 = \{0 \mid 1\}, \dots$$

$$m/2^k = \{ (m-1)/2^k \mid (m+1)/2^k \}. \tag{14}$$



These numbers (integers and rationals) can be added as the usual ways. Numbers are well ordered, and their relations with 0 are clear. Hence, one can easily determine the outcome for any sum of numbers.

3.2.2 Nimbers

A game G is a *nimber* [8][31] if all the elements in G^L and G^R are nimbers and $G^L = G^R$.

Nimbers are defined as:

$$*1 = \{0 \mid 0\},$$

$$*2 = \{0, *1 \mid 0, *1\},$$

...

$$*n = \{0, *1, *2, \dots, *(n-1) \mid 0, *1, *2, \dots, *(n-1)\}. \quad (15)$$

For simplicity, $*1$ is also denoted as $*$ and named *star*. The special number with infinite options:

$$\star = \{0, *1, *2, \dots \mid 0, *1, *2, \dots\}. \quad (16)$$

is named *remote star*.

For each non-zero number, the first player can win a game. That is, each non-zero number is confused with 0. Hence one can easily determine the outcome of any sum of numbers [31][60]. From this, two well-known properties are (1) $*n + *n = 0$, and (2) $\{*n \mid *n\} = 0$.

3.2.3 Sumbers



For each number d , there is a corresponding *up* defined as [7][20][36].

$$\uparrow(d) = \{\uparrow(d^L), * \mid \uparrow(d^R), *\}. \quad (17)$$

The negation of up is called *down*.

$$\downarrow(d) = -\uparrow(d). \quad (18)$$

A property between all ups and stars [7][20] is: for all numbers $d > 0$ and $n > 1$, we have

$$\uparrow(d) > *n \text{ and } \uparrow(d) > \star; \quad (19)$$

and, for all numbers d , we have

$$\uparrow(d) \parallel *1 \text{ (or } \uparrow(d) \parallel *). \quad (20)$$

We use the notation $m.\uparrow(d)$ to denote the sum of m copies of $\uparrow(d)$. A *sumber* S (cf. [39]) is a sum of ups, downs and stars(*).

$$S = \sum_{k=1,n} a_k.\uparrow(d_k) + a_0.*. \quad (21)$$

where a_k are integers and d_k are numbers, $0 < k \leq n$. Without loss of generality, in (21), we assume $0 < d_1 < d_2 < \dots < d_n$ and $a_0=0$ or 1 . Clearly, sumbers are closed under addition. We use the notation $G \ll H$ to denote that the sum of any number of copies of G is less than H . The sumbers have the following properties:

$$0 < \uparrow(d_1) < \uparrow(d_2), \quad (22)$$

$$0 \ll \uparrow(d_{n+1}) - \uparrow(d_n) \ll \uparrow(d_n) - \uparrow(d_{n-1}), \quad (23)$$

$$\uparrow(d_{n+1}) + \uparrow(d_{n+1}) - \uparrow(d_n) > *, \quad (24)$$

where $0 < d_1 < d_2 < \dots < d_n < d_{n+1} < \dots$. These properties are *sufficient* to determine the outcome of any sum of sumbers. The research in [39] provides a simple rule to determine the outcome of (21):

$$S > 0 \text{ if and only if } \sum_{k=1,n} a_k > a_0 \text{ or } (\sum_{k=1,n} a_k = a_0, \text{ and } a_1 < 0),$$

where a_0 is either 1 or 0. Note that the net number of ups is greater than the net number of *, or the net number of ups equals the net number of * and the smallest up has a negative coefficient.

For example, consider $S_A = -\uparrow(3) + 3.\uparrow(1) + *$. In S_A , the net number of ups (=2) is greater than the net number of * (=1), thus $S_A > 0$. Consider $S_B = -\uparrow(3) + 3.\uparrow(2) - \uparrow(1) + *$. In S_B , the net number of ups (=1) equals the net number of * (=1), and the smallest up (=1)

has a negative coefficient ($= -1$), thus $S_B > 0$. Consider $S_C = \uparrow(3) - 2.\uparrow(2) + 2.\uparrow(1) + *$. In S_C , the net number of ups ($=1$) equals the net number of $*$ ($=1$), but the smallest up ($=\uparrow(1)$) has a positive coefficient ($=2$), thus $S_C \not> 0$. Let “ $\not>$ ” denote “not greater than”.

3.2.4 Infinitesimal and Atomic Weight

A game G is called an *infinitesimal* if and only if G is less than any positive numbers and greater than any negative numbers. Nimbers and sumbers are all infinitesimals. Researchers in [7][20] introduced the definition of atomic weight. If $G = \{G_a, G_b, G_c, \dots | G_d, G_e, G_f, \dots\}$ where $G_a, G_b, G_c, G_d, G_e, G_f, \dots$ have atomic weight a, b, c, d, e, f, \dots , then the atomic weight of G is

$$G_0 = \{a - 2, b - 2, c - 2, \dots | d + 2, e + 2, f + 2, \dots\}$$

unless G_0 is an integer and either $G > \star$ or $G < \star$. In these exceptional cases, if $G > \star$ then the atomic weight of G is the largest integer $< | d + 2, e + 2, f + 2, \dots$, and if $G < \star$ then the atomic weight of G is the least integer $> a - 2, b - 2, c - 2, \dots$

According to the above definition, each nimber has atomic weight 0; each up has atomic weight 1.

Two important properties [7][20] about atomic weights are described as follows.

1. The atomic weight of a sum of games equals to the sum of the atomic weights of the games.
2. If the atomic weight of a game is greater than or equals to 2, then Left wins the game. On the other hand, if it is less than or equals to -2 , then Right wins the game. However, there are no general rules when the atomic weight is between -2 and 2.

For example, $\uparrow + \uparrow(2)$ has atomic weight 2, hence Left can win the game; $\downarrow + \uparrow(2) + \downarrow(3) + \downarrow(4) - \star + * - *(3)$ has atomic weight -2 , hence Right can win the game. Thus, for some games, we can determine the winners by computing the atomic weight of sub-games, instead of searching complex trees.

3.3 Domineering and XT Domineering

Domineering (also called Stop-Gate or Crosscram) [27] is a mathematical game played on a board with $n \times n$ squares. Two players have a collection of 1×2 and 2×1 dominos which they place on the grid in turn, covering up squares. One player, Left, plays first and places domino vertically (1×2), while the other, Right, places horizontally (2×1). The first player who cannot place a domino loses the game.

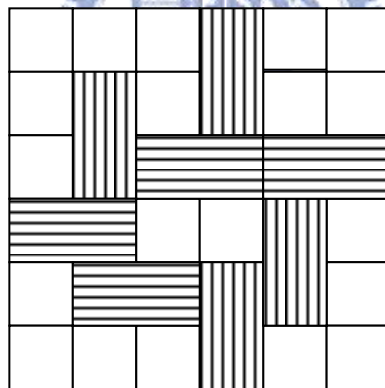


Figure 17. Middle game of 6×6 Domineering.

As the game progresses, the original $n \times n$ squares may be partitioned into a set of disjoint sub-positions. Figure 17 shows a graph in the middle of a 6×6 Domineering. It contains 5 disjoint sub-positions shown in Figure 18.

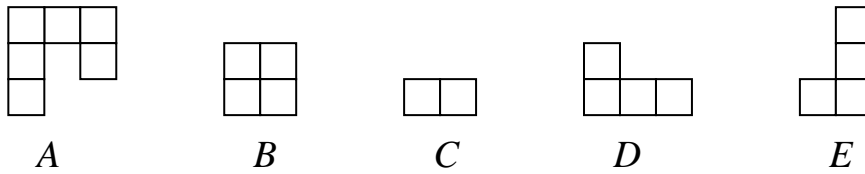


Figure 18. Sub-positions of the graph in Figure 17

In terms of CGT, the game G in Figure 17 is a sum of sub-positions A , B , C , D , and E , i.e., $G = A + B + C + D + E$. Note that by rotating position D 90° counter clockwise, one can get position E . In general, rotating a Domineering position 90° (either clockwise or counter clockwise) will result a negation of the original position, and reflecting a Domineering position with respect to a vertical axis or horizontal axis will not change the game value of the position. Hence, $E = -D$, and $G = A + B + C$.

Domineering attracted many combinatorial game researchers because the game contains many numbers, switches of numbers, and complicated hot positions.

Figure 19 (below) shows the game values of the positions in Figure 18. Note that the derivations are based on [5] and the details of derivations are therefore omitted in this thesis. By summing up the values, we have $G = 3/4 + \{1 \mid -1\} - 1 = -1/4 + \{1 \mid -1\} = \{3/4 \mid -5/4\}$, thus the first player can win the game. This illustrates the power of using combinatorial theory, since we can derive the result without tree search as many board games do. A simpler example is illustrated in Appendix A.

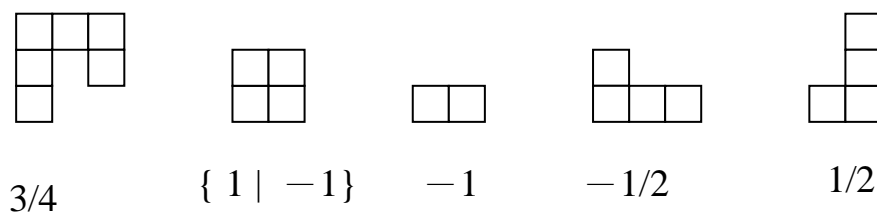


Figure 19. Some game values in Domineering.

XT Domineering is modified from the Domineering game by changing the rule to allow a player placing a small (1×1) domino on a sub-position while unable to place his big domino (1×2 or 2×1) in the sub-position in the original Domineering game. For example, consider sub-position C in Figure 18. In Domineering, Left cannot place a domino vertically (1×2) at sub-position C , while in XT Domineering, Left is allowed to place a 1×1 domino at sub-position C . More specifically, sub-position C has the value $\{\emptyset \mid 0\} = -1$ in Domineering, and $\{\{0 \mid 0\} \parallel 0\} = \{*\mid 0\} = \downarrow$ in XT Domineering. Note that Left is not allowed to place a 1×1 domino at a position while he is able to place a 1×2 domino at that position and Right is not allowed to place a 1×1 domino at a position while he is able to place a 2×1 domino at that position. For example, both players are not allowed to place 1×1 domino at positions A , B , D and E in Figure 18.

Since XT Domineering has at least the same number of options as Domineering and allows more moves (e.g., on 1×1 vacancies), XT Domineering has higher game-tree complexity [35].

Note that each player has at least one option at any non-empty position in XT Domineering. This nature prevents the occurrence of non-zero numbers and ensures that each position in XT Domineering is an infinitesimal. One of the major motivations of this thesis is to see what kind of infinitesimals may be shown up in this game.

3.4 Game Values of 3×3 XT Domineering

For XT Domineering with $1 \times n$ squares, the games have periodic values with period length 8, $\{0, *, \downarrow, \uparrow, *, 0, \uparrow^*, \downarrow^*\}$ [38]. This is in fact a partisan octal game [51]. In this section, we investigate a total of 2^9 sub-graphs of 3×3 squares in XT Domineering.

After excluding non-connected sub-graphs, rotated negation sub-graphs, or reflected equivalence sub-graphs, there are 34 distinct positions. The game values of these distinct positions are derived based on the above inequalities (6) to (24), and shown in Table 6.

Each position in Table 6 is a linear combination of the following 8 elementary games:

$$* = \{0 \mid 0\}, \quad (25)$$

$$\uparrow = \{0 \mid *\}, \quad (26)$$

$$\uparrow^+ = \{\uparrow \mid *\}, \quad (27)$$

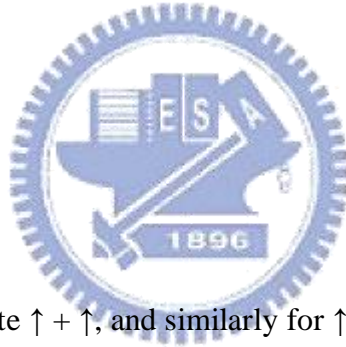
$$\uparrow/2 = \{\uparrow\uparrow^* \mid \downarrow^*\}, \quad (28)$$

$$\star = \{0, \uparrow^* \mid \downarrow^*, 0\}, \quad (29)$$

$$*/2 = \{\uparrow\uparrow \mid \downarrow\downarrow^*\}, \quad (30)$$

$$(* / 2)^+ = \{\uparrow\uparrow, \uparrow\uparrow^* \mid \downarrow\downarrow^*\}, \quad (31)$$

$$\diamond = \{\uparrow\uparrow\uparrow^* \mid \downarrow\downarrow\downarrow^*\}. \quad (32)$$



For simplicity, let $\uparrow\uparrow$ indicate $\uparrow + \uparrow$, and similarly for $\uparrow\uparrow^*$, $\uparrow\uparrow\uparrow^*$, etc.

The games $*$, \uparrow and \uparrow^+ ($=\uparrow(2)$) have been introduced in Section 3.3. $*$ has atomic weight 0 (as described in Subsection 3.2.4), while \uparrow and \uparrow^+ have atomic weight 1 each. We use the symbol \uparrow^2 to denote $\uparrow^+ - \uparrow$.

$$\uparrow^2 = \uparrow^+ - \uparrow. \quad (33)$$

From inequality (23), we have

$$\uparrow \gg \uparrow^2 > 0. \quad (34)$$

The game $\uparrow/2$ (*half up*), as the name suggested, has atomic weight 1/2 and the following properties:

$$\uparrow/2 + \uparrow/2 = \uparrow. \quad (35)$$

$$\uparrow/2 > \uparrow^2. \quad (36)$$

The game \star (*black star*) has atomic weight 0 and with property similar to numbers:

$$\star + \star = 0, \quad (37)$$

$$\star \parallel *(n), \text{ for integer } n > 0 \quad (38)$$

The game $*/2$ (*half star*), as the name suggested, has the following property:

$$*/2 + */2 = *. \quad (39)$$

The game $*/2$ has atomic weight $\{0 \mid 0\} = *$, since the atomic weight of $\uparrow\uparrow$ is + 2 and that of $\downarrow\downarrow*$ is -2.

The game $(*/2)^+$ (*half star plus*), as the name suggested, is just slightly greater than $*/2$ and has atomic weight $\{0, 0 \mid 0\} = *$. The difference between $(*/2)^+$ and $*/2$ is named \triangle_* :

$$\triangle_* = (*/2)^+ - */2 > 0. \quad (40)$$

Since the atomic weight of both $(*/2)^+$ and $*/2$ are $*$, the atomic weight of \triangle_* equals $* - * = 0$.

The game \diamond (*diamond*) has atomic weight $\{1 \mid -1\}$. Since the incentive of \diamond (*diamond*) is greater than the ones of all the other 7 elementary games, \diamond should always be played first among the 8 elementary games. Diamond also has the property below:

$$\diamond + \diamond = 0 \quad (41)$$

No.	Position	Value	No.	Position	Value
P_{1-1}		*	P_{6-1}		$*/2$
P_{2-1}		↑	P_{6-2}		$*/2$
P_{3-1}		0	P_{6-3}		↑↑*
P_{3-2}		↓	P_{6-4}		↑ ⁺
P_{4-1}		*	P_{6-5}		↑/2
P_{4-2}		*	P_{6-6}		★ + *
P_{4-3}		↑↑*	P_{6-7}		0
P_{4-4}		*	P_{6-8}		0
P_{5-1}		★	P_{7-1}		*
P_{5-2}		*	P_{7-2}		↑*
P_{5-3}		↑↑	P_{7-3}		↑
P_{5-4}		★	P_{7-4}		↑*
P_{5-5}		↑↑	P_{7-5}		*
P_{5-6}		↑	P_{7-6}		0
P_{5-7}		0	P_{7-7}		$* + (* / 2)^+$
P_{8-1}		*	P_{8-3}		0
P_{8-2}		◇	P_{9-1}		0

Table 6. Game values of 3x3 XT Domineering

The calculation for the values of positions in Table 6 is a tedious process. In general, one first derives a position expression according to the rule and then simplifies the expression by removing the *dominated* options and replacing with the *reversible* options (c.f. [7] and [20]). For example, considering P_{4-3} , according the rule $P_{4-3} = \{0, \downarrow \mid \uparrow\}$. After eliminating the dominated option \downarrow ($\downarrow < 0$), one can get $P_{4-3} = \{0 \mid \uparrow\}$. Considering P_{5-7} , according the rule $P_{5-7} = \{*\mid*\}$. After replacing P_{5-7} with reversible option ($P_{5-7}^{LR} = 0$), one can get $P_{5-7} = 0$. After simplifying a position, one needs to check whether the position can be represented as a sum of simpler game. For example, $P_{4-3} = \{0 \mid \uparrow\} = \uparrow\uparrow*$. The research in [39] provided an algorithm to simplify switches of up sums into up sums whenever possible. The game values in Table 6 have also been verified in CgSuite [59], a useful tool for deriving game values.

Figure 20 (below) shows the corresponding XT Domineering games values of positions in Figure 18. The derivations for C, E, and E+* are illustrated in Appendix A.

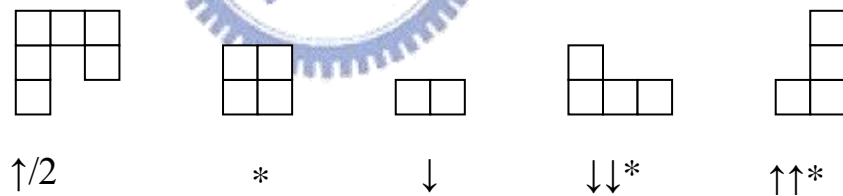


Figure 20. Some game values in XT Domineering.

The sum in Figure 20 is $\uparrow/2 + * + \downarrow + \downarrow\downarrow* + \uparrow\uparrow* = \downarrow/2 + * = \{\uparrow \mid \downarrow\downarrow\}$. Hence the first player can win the game.

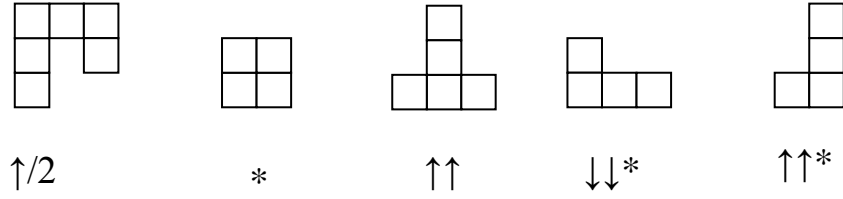


Figure 21. Some game values in XT Domineering.

Assume that sub-position C is changed as shown in Figure 21. Then, the sum in Figure 21 becomes $\uparrow/2 + * + \uparrow\uparrow + \downarrow\downarrow* + \uparrow\uparrow* = \uparrow\uparrow + \uparrow/2 + *$. Since the atomic weight of the above sum is $2 + 1/2$, over 2, Left wins the game. From above examples, Table 6 becomes an important knowledge base for playing the game of XT Domineering.

3.5 Outcome of 3×3 XT Domineering

In the previous section, we derive the values of positions in Table 6. Then, we can easily determine the outcome of sums, if the atomic weights are at least 2 or at most -2 . However, there are no simple rules when the atomic weights are between -2 and 2.

This section discusses the approach to determine the outcome of sums of 3×3 XT Domineering, even when the atomic weights are between -2 and 2. Since the game \diamond will always be played before any other games in Table 6, we may only focus on the analysis of sums of the other 7 elementary games. Without loss of generality, a sum S of any positions in Table 6 can be written as:

$$S = S_A + S_B + S_C, \tag{42}$$

where S_A is a linear combination of \uparrow^+ , \uparrow and $\uparrow/2$,

S_B is a linear combination of $*$ and \star , and

S_C is a linear combination of $*/2$ and $\triangle*$.

S_A measures the *up-ness* (or advantage for Left) of S ; S_B is a sum that neither player has advantage; S_C consists of games with atomic weight $*$. There are only 4 possible cases of S_B , as shown in the column subhead of Table 7, and 9 possible cases of S_C , as shown in the row subhead of Table 7. Note that the atomic weight of S_C is 0 in row 1, 2, 3 and 4, and $*$ in row 5, 6, 7, 8 and 9.

$S_C \setminus S_B$	1	2	3	4
$(n > 0)$	0	★	*	★ + *
1 $(n+1).\triangle*$	0	\uparrow^2	0	\uparrow^2
2 $\triangle*$	0	\uparrow^2	$\uparrow+2.\uparrow^2$	$\uparrow+\uparrow^2$
3 0	$\uparrow^2,$ $\uparrow/2-\uparrow^2$	\uparrow^2	$\uparrow+2.\uparrow^2$	$\uparrow+\uparrow^2$
4 $-n.\triangle*$	$\uparrow+2.\uparrow^2$	$\uparrow+\uparrow^2$	$\uparrow+2.\uparrow^2$	$\uparrow+\uparrow^2$
5 $*/2+(n+2).\triangle*$	$\uparrow/2$	$\uparrow/2+\uparrow^2$	$\uparrow/2$	$\uparrow/2+\uparrow^2$
6 $*/2+2.\triangle*$	$\uparrow/2$	$\uparrow/2+\uparrow^2$	$\uparrow-\uparrow^2,$ $\uparrow/2+2.\uparrow^2$	$\uparrow/2+\uparrow^2$
7 $*/2+\triangle*$	$\uparrow/2$	$\uparrow/2+\uparrow^2$	$\uparrow/2+\uparrow+2.\uparrow^2$	$\uparrow/2+\uparrow+\uparrow^2$
8 $*/2$	$\uparrow-\uparrow^2,$ $\uparrow/2+2.\uparrow^2$	$\uparrow/2+\uparrow^2$	$\uparrow/2+\uparrow+2.\uparrow^2$	$\uparrow/2+\uparrow+\uparrow^2$
9 $*/2-n.\triangle*$	$\uparrow/2+\uparrow+2.\uparrow^2$	$\uparrow/2+\uparrow+\uparrow^2$	$\uparrow/2+\uparrow+2.\uparrow^2$	$\uparrow/2+\uparrow+\uparrow^2$

Table 7. Minimum ups U required for $U + S_B + S_C > 0$

Table 7 is a set of 39 inequalities (note that there are two values in each of $grid(3,1)$, $grid(8,1)$ and $grid(6,3)$), $1 \leq i \leq 9$, $1 \leq j \leq 4$,

$$grid(i, j) + row(i) + col(j) > 0. \quad (43)$$

The proof for these inequalities is given in Appendix B. Let us illustrate by some example. The ups in $grid(9, 2)$ is $\uparrow/2 + \uparrow + \uparrow^2$, it corresponds to the inequality:

$$\uparrow/2 + \uparrow + \uparrow^2 + */2 - n.\triangle_* + \star > 0, \text{ for } n > 0.$$

$Grid(3, 1)$ represents 2 inequalities: $\uparrow^2 > 0$ and $\uparrow/2 - \uparrow^2 > 0$; $grid(8, 1)$ represents 2 inequalities: $\uparrow - \uparrow^2 + */2 > 0$ and $\uparrow/2 + 2.\uparrow^2 + */2 > 0$. These inequalities are sufficient to determine the outcome of any sum of the 8 elementary games. The general steps to determine the outcome of a sum S of 3×3 XT Domineering is described as follows:

1. Check the game value of each of S 's position from Table 6.
2. If there is any \diamond in the sum, play it out first.
3. Denote the sum $S_A + S_B + S_C$, (42) by S , and determine the value of S_A , S_B and S_C .
4. Use S_B and S_C to lookup Table 7 for the minimum ups U required.
5. Determine whether $S_A \geq U$ or not. Inequalities (34), (35) and (36) can help the determining process.
6. $S > 0$ if and only if $S_A \geq U$.
7. To determine whether $S < 0$ or not, it is equivalent to determining whether $-S > 0$ or not. Apply the above steps to $-S$.

For example, consider the sum S of the sub-positions as shown in Figure 22 (below) and who wins the game.

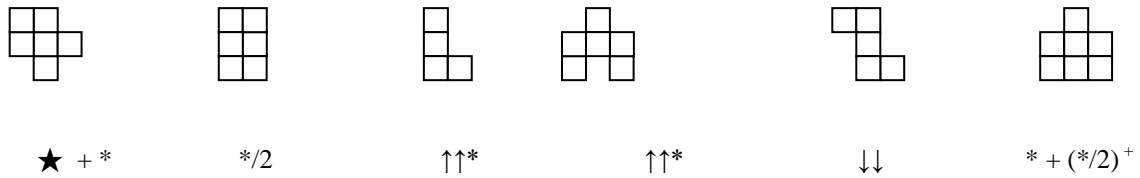


Figure 22. Some game values in XT Domineering.

The sum S can be simplified as:

$$\begin{aligned}
 S &= \star + * + */2 + \uparrow\uparrow* + \uparrow\uparrow* + \downarrow\downarrow + * + (* / 2)^+ \\
 &= \star + * + */2 + \uparrow\uparrow + * + (* / 2)^+ \\
 &= \star - */2 + \uparrow\uparrow + * + (* / 2)^+ \\
 &= \star + \uparrow\uparrow* + \triangle*
 \end{aligned}$$

and,

$$S_A = \uparrow\uparrow,$$

$$S_B = \star + *,$$

$$S_C = \triangle*.$$

Using S_B and S_C to lookup Table 7, we get $U = \uparrow + \uparrow^2$. Since $S_A = \uparrow\uparrow > \uparrow + \uparrow^2 = U$, we conclude $S > 0$. Hence the game is a win for Left, no matter who moves first.

3.6 Conclusion And Further Consideration

This chapter has the following three major contributions. First, we present a new game, XT Domineering, which has higher game-tree complexity [35] than Domineering.

Second, we also have presented a mathematical approach to solve sums of 3×3 XT Domineering. Again, this success demonstrates the potential of applying CGT to solving more of other intelligent games.

After solving 3×3 XT Domineering, it is natural to think of 3×4 , 4×4 , or even larger size XT Domineering. According to our preliminary study, there seems to be no simple close form equation that can relate a given position to its game value. Thus a lookup table is required to store the values of all the positions. CgSuite [59] is a useful tool to derive the values. After deriving the canonical form of the game values, one still needs to check whether a game can be decomposed as a sum of simpler elementary games. Unfortunately, there are too many sub-positions in 3×4 , 4×4 , or even larger size XT Domineering, we cannot afford to examine all the positions and check whether they can be decomposed as simpler elementary games. An automated game decomposition procedure is in need and deserves further research in the future.

Third, we find several infinitesimal games with interesting properties, including \star , $\ast/2$, $(\ast/2)^+$ and $\uparrow/2$. It is worth further research to find more other interesting infinitesimal games. The game of XT Domineering is a rich source of infinitesimal games.

Chapter 4 NoGo Endgame Analysis

This chapter investigates a combinatorial game named NoGo. We use combinatorial game theory (CGT) to calculate the mean and temperature values of NoGo positions. In the rest of this chapter, Section 4.1 introduces the game, Section 4.2 classifies the moves, Section 4.3 defines means and temperatures of games, Section 4.4 analyzes means and temperatures of NoGo positions, Section 4.5 proposes some more propositions for NoGo, and Section 4.6 makes conclusion.

4.1. Introduction

NoGo is a young game which was first called Anti-Atari Go in 2005, by John Moore [55]. Later, the game was also introduced by the organizers of the BIRS workshop, Combinatorial Game Theory 2011, for being a completely new combinatorial game [15] and named NoGo. NoGo is a 2-player and perfect-information board game whose rules are almost the same as Go, but stones cannot be removed after placed. Namely, the rules different from Go are summarized as follows. First, the moves to capture stones like stone capturing in Go are prohibited. Note that suicide moves prohibited in Go are also prohibited in NoGo. Second, the move pass is not allowed in NoGo, and then the player who has no moves to play loses the game.

The first NoGo competition was also held in the BIRS workshop on Combinatorial Game Theory 2011. Since that time, several NoGo computer game competitions have been held, such as the 2011 and 2013 Computer Olympiads in Netherlands and Japan, respectively, and 2012 TAAI, 2012 TCGA, and 2013 TCGA in Taiwan.

NoGo is a PSPACE problem as shown in [13], but it is not ensured to be PSPACE completeness. In this chapter, we use CGT to analyze the game positions to understand the characteristics of NoGo.

4.2. Classification of Moves

All empty grids in the board of NoGo are classified in the following categories.

1. The grid which both players can place a stone on is called *Both-Go* or *2-Go*.
2. The grid which only one player can place a stone on is called *One-Go* or *1-Go*. If only Black can place a stone on, the grid is called *Black-Go* or *B-Go*, and otherwise called *White-Go* or *W-Go*. A property is that a player can no longer place a stone on a grid once the player cannot place a stone on it. That is, when a grid becomes 1-Go, it will not become a 2-Go again.
3. The grid which neither player can place a stone on is called *No-Go* or *0-Go*. Similarly, a property is: when a grid becomes a 0-Go, it will not become a 1-Go or 2-Go again.

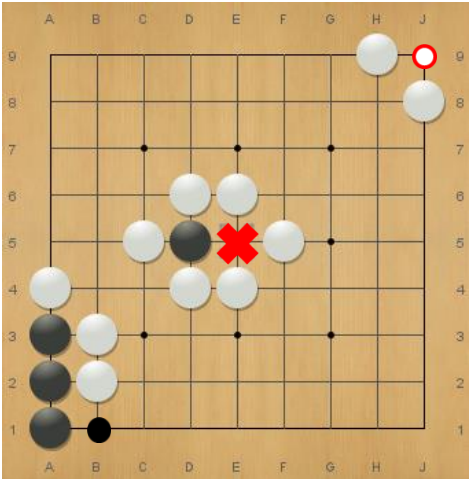
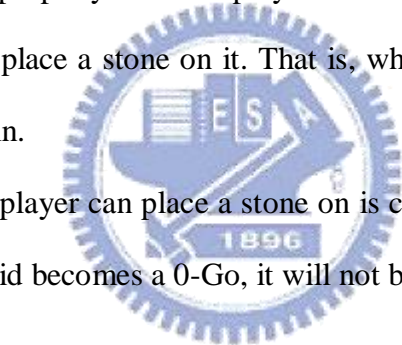


Figure 23: The classification of moves in NoGo.

Figure 23 illustrates the classification. B1, marked with the black circle is a B-Go

which only Black can place a stone on it. If White places a stone on the grid, it will capture the Black's stones. Similarly, J9, marked with the white circle is a W-Go which only White can place a stone on it. If Black places a stone on the grid, the stone commits suicide due to the surrounding of White. E5, marked with an X symbol is a No-Go which neither Black nor White can place any stone on, since Black cannot place on it due to the prohibition of suicide and White cannot due to the prohibition of capturing Black's stone.

4.3. Mean and Temperature

For each combinatorial game, there are two important values, *mean* and *temperature*. Roughly speaking, mean is a measure of the average outcome and temperature is a measure of the move size of a game. The existence of mean values of games was first raised and proved by Milnor (1953) [52] and Hanner (1959) [32].

A constructive algorithm, named *thermograph*, for mean and temperature was due to Berlekamp *et al.* (1982) [7] and Conway (1976) [20]. An approach to calculating mean and temperature with partial information of a single branch game was proposed by Kao (1998) [37]. Müller *et al.* (2004) [54] proposed to use a coupon stack CS with decreasing temperatures to simulate the environment and calculating the temperature of a game G by tracing the move sequence of the sum $G + CS$. Lew and Coulom (2010) [46] proposed to estimate the mean and temperature of a game from its left and right stops and calculating these stops by temporal difference learning.

In the rest of this section, Subsection 4.3.1 introduces definitions of mean and temperature. Then, a method of calculation of mean and temperature presents in Subsection 4.3.2.

4.3.1 Definitions of Mean and Temperature

In this subsection, we review the definitions of the mean and temperature of a game G , denoted as $m(G)$ and $t(G)$, respectively [7][20][42].

Let G be a game and t be a number, define

$$L(G, t) = \max_{\{x \in G^L\}} \begin{cases} m(G) \\ R(x, t) - t \end{cases} \quad (44)$$

$$R(G, t) = \min_{\{y \in G^R\}} \begin{cases} m(G) \\ L(y, t) + t \end{cases} \quad (45)$$

$L(G, t)$ is called the *left wall* and $R(G, t)$ the *right wall* of G . $L(G, t)$ ($R(G, t)$) is the min-max optimal outcome of the game G when L (R) moves first and subject to the constraint that L has the right either to accept the mean of G as the outcome or to make a move at G and pay a tax t . When G is a number (terminal position), both players will accept the number (equals its mean) as the outcome value and make no more move. Note that, for non-number games,

1. When the tax t is low, the players may prefer to make a move and pay the tax t than accept the mean as the outcome.
2. When the tax t is too high, the players may prefer to accept the mean as the outcome than make a move and pay the tax t .
3. $L(G, t)$ is monotonically decreasing with respect to t . The higher the tax t , the lower the optimal outcome value when L moves first.
4. $R(G, t)$ is monotonically increasing with respect to t . The higher the tax t , the higher the optimal outcome value when R moves first.

5. When the tax t is low, we have $L(G, t) > m(G) > R(G, t)$. When the tax t reaches or exceeds some critical value, we have $L(G, t) = m(G) = R(G, t)$.

Thus, finding the mean and temperature of a game G is indeed a task of solving the min-max equation below.

$$\max_{\{x \in G^L\}} R(x, t) - t = \min_{\{y \in G^R\}} L(y, t) + t \quad (46)$$

There might be more than one solution of t for the above equation. The minimum solution of t is $t(G)$. When $t = t(G)$, the solution of the min-max equation equals $m(G)$. Mean and temperature of a game have the following properties. Let G and H be two games. We have

$$m(G + H) = m(G) + m(H) \quad (47)$$

$$t(G + H) \leq \max\{t(G), t(H)\} \quad (48)$$

Hence knowing the mean and temperature of games in a sum can help the estimation of the mean and temperature of the sum. Another important feature of mean and temperature is that they can be used to estimate the range of the optimal outcome of a game. The inequality below shows the bounds of the optimal outcomes $L(G, t)$ and $R(G, t)$.

$$m(G) - t(G) \leq R(G, t) \leq m(G) \leq L(G, t) \leq m(G) + t(G) \quad (49)$$

4.3.2 Thermograph

This subsection reviews the thermograph approach to calculating the mean and temperature of a game.

A function $f(t)$ is called *simple max* if it can be written as:

$$f(t) = \max\{c_1, \min\{c_2 - t, \dots \max\{c_{2k-1}, \min\{c_{2k} - t, \dots\}\}\}\} \quad (50)$$

where $c_{2k} > c_{2k+2}, c_{2i} > c_{2k+1} > c_{2k-1}$.

Similarly, $g(t)$ is called *simple min* if it can be written as:

$$g(t) = \min\{c_1, \max\{c_2 + t, \dots \min\{c_{2k-1}, \max\{c_{2k} + t, \dots\}\}\}\} \quad (51)$$

where $c_{2k-1} > c_{2k+1} > c_{2i}, c_{2k+2} > c_{2k}$.

Each simple max(min) function can be represented as a sequence $[c_1, \dots, c_n]$ of constants. The graph of a simple max(min) function is a folded line. Figure 24(a) and (b) show the graph of simple max and min functions.

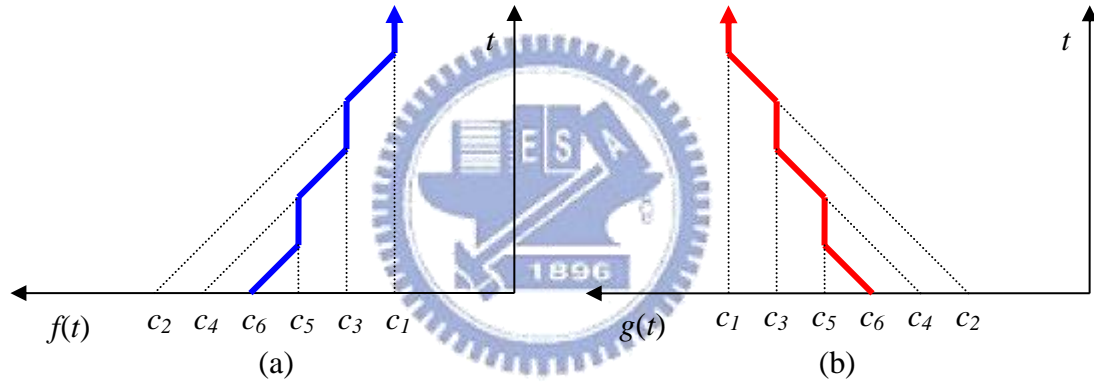


Figure 24: Thermographs of (a) a simple max function and (b) a simple min function.

It should be clear that the max(min) of two simple max(min) functions is again a simple max(min) function. If $f(t)$ is a simple max function and c is a number, then $\min\{c, f(t) + t\}$ is a simple min function. If $g(t)$ is a simple min function and c is a number, then $\max\{c, g(t) - t\}$ is a simple max function. Thus, the left wall of a game is a simple max function and the right wall is a simple min function. Figure 25 illustrates the thermograph of $G = \{3|\{0|-2\}\}$.

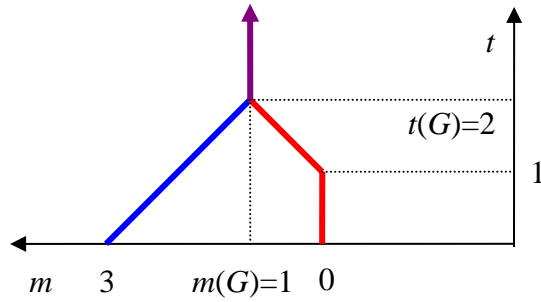


Figure 25: The thermograph of $G = \{3|\{0|-2\}\}$.

The general procedure to calculate the left wall (LW) and the right wall (RW) of a game G is as follows.

1. Calculate the walls of all G 's children.
2. Find the max of the RW s of G 's left children. Store the result as $R(t)$.
3. Find the min of the LW s of G 's right children. Store the result as $L(t)$.
4. Calculate $m(G)$. That is to solve the equation $R(t) - t = L(t) + t$.
5. $LW = \max\{m(G), R(t) - t\}$
6. $RW = \min\{m(G), L(t) + t\}$

The above procedure is recursive. To calculate the walls of G , one needs to calculate the walls of all G 's children first (Step 1). Eventually, the walls of all the offspring of G must be calculated in order to calculate the walls of G .

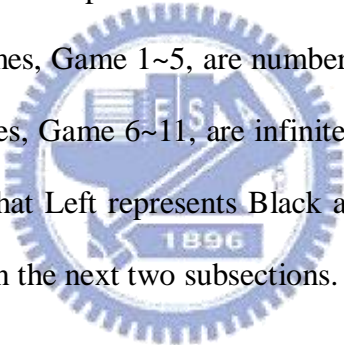
4.4. Game Values of NoGo Positions

Since 4×4 NoGo positions include many interesting game values, this section only investigates the game values of 4×4 NoGo positions. Especially, we calculate game values for 4×4 NoGo positions with endgame search depth 6 and 8. A position is called to have

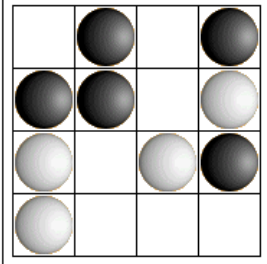
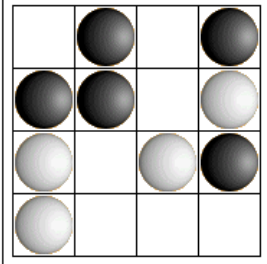
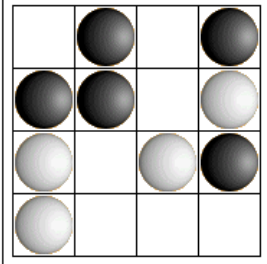
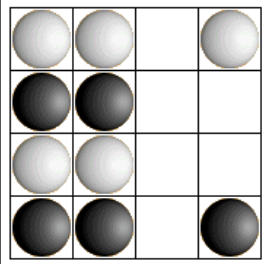
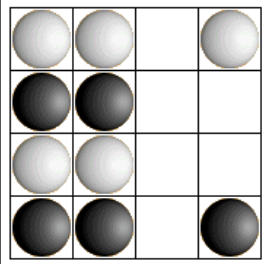
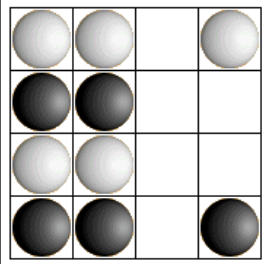
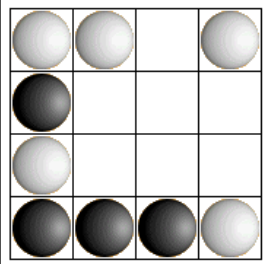
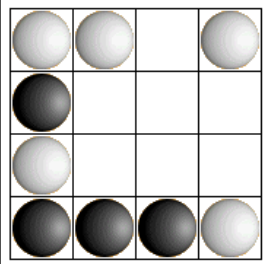
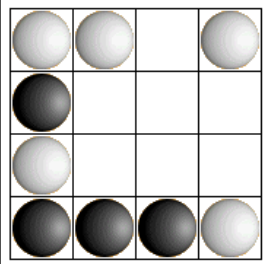
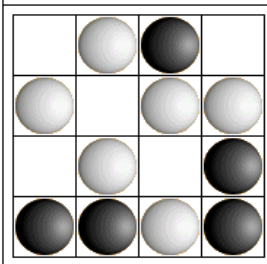
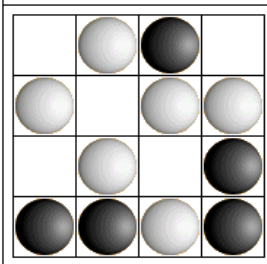
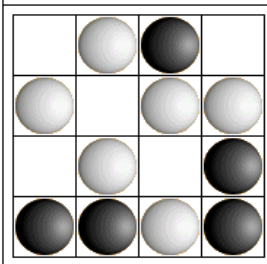
endgame search depth d , if at most d moves are required to end the game and there exists some case where d moves are required to end the game. From the game values, we can also obtain their means, temperatures and atomic weights. We investigate, in total, 89,964 and 21,454 4×4 NoGo positions with endgame search depth 6 and 8, respectively. All of 4×4 NoGo game values were derived by CgSuite [59]. Since the computation times by CgSuite are high for positions with high endgame search depth, we analyze the positions with the depth 6 and 8, not higher.

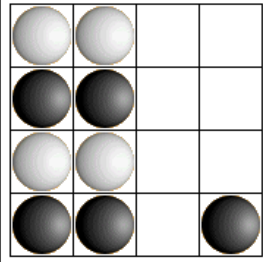
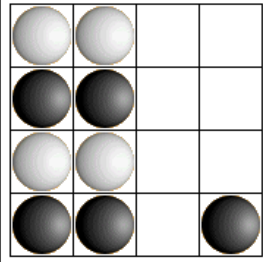
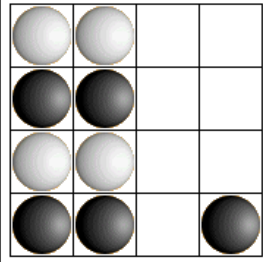
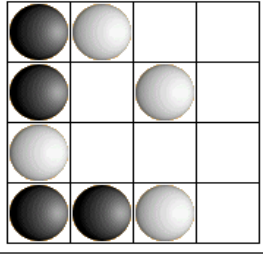
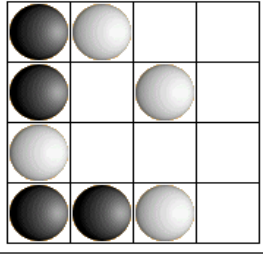
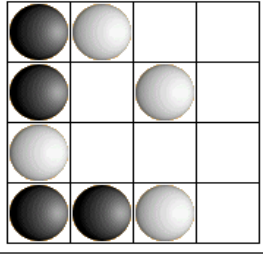
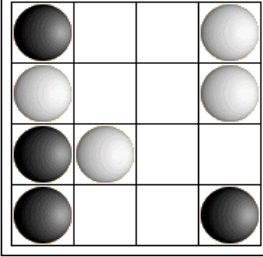
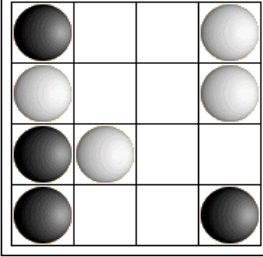
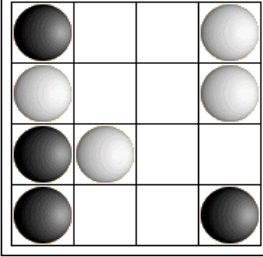
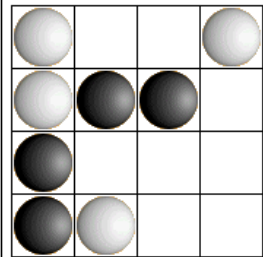
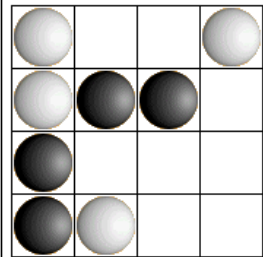
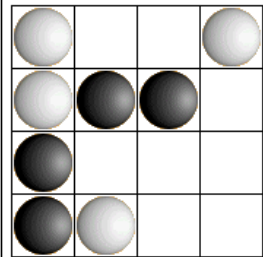
4.4.1 Game Values

The game values of 4×4 NoGo positions include numbers, infinitesimals, and hot games. In Table 8, the first 5 games, Game 1~5, are numbers with values, 2, 1, 0, -1 and -2, respectively; and the next 6 games, Game 6~11, are infinitesimals with values, *, \uparrow , \uparrow^* , $\uparrow\uparrow$, $\uparrow\uparrow^*$, and \uparrow^2 , respectively. Note that Left represents Black and Right represents White. The hot games are discussed further in the next two subsections.



ID	Position and Game Values									
1	"Position"				"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"
					2	[1]	[]	-1	2	"N/A"

2	<table border="1"> <thead> <tr> <th data-bbox="384 219 651 271">"Position"</th> <th data-bbox="655 219 740 271">"Value"</th> <th data-bbox="745 219 813 271">"Left"</th> <th data-bbox="818 219 887 271">"Right"</th> <th data-bbox="892 219 1050 271">"Temperature"</th> <th data-bbox="1054 219 1139 271">"Meam"</th> <th data-bbox="1144 219 1315 271">"Atomic Weight"</th> </tr> </thead> <tbody> <tr> <td data-bbox="384 277 651 539">  </td> <td data-bbox="655 277 740 539">1</td> <td data-bbox="745 277 813 539">[0]</td> <td data-bbox="818 277 887 539">[]</td> <td data-bbox="892 277 1050 539">-1</td> <td data-bbox="1054 277 1139 539">1</td> <td data-bbox="1144 277 1315 539">"N/A"</td> </tr> </tbody> </table>	"Position"	"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"		1	[0]	[]	-1	1	"N/A"
"Position"	"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"									
	1	[0]	[]	-1	1	"N/A"									
3	<table border="1"> <thead> <tr> <th data-bbox="384 602 651 654">"Position"</th> <th data-bbox="655 602 740 654">"Value"</th> <th data-bbox="745 602 813 654">"Left"</th> <th data-bbox="818 602 887 654">"Right"</th> <th data-bbox="892 602 1050 654">"Temperature"</th> <th data-bbox="1054 602 1139 654">"Meam"</th> <th data-bbox="1144 602 1315 654">"Atomic Weight"</th> </tr> </thead> <tbody> <tr> <td data-bbox="384 660 651 922">  </td> <td data-bbox="655 660 740 922">0</td> <td data-bbox="745 660 813 922">[]</td> <td data-bbox="818 660 887 922">[]</td> <td data-bbox="892 660 1050 922">-1</td> <td data-bbox="1054 660 1139 922">0</td> <td data-bbox="1144 660 1315 922">0</td> </tr> </tbody> </table>	"Position"	"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"		0	[]	[]	-1	0	0
"Position"	"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"									
	0	[]	[]	-1	0	0									
4	<table border="1"> <thead> <tr> <th data-bbox="384 985 651 1037">"Position"</th> <th data-bbox="655 985 740 1037">"Value"</th> <th data-bbox="745 985 813 1037">"Left"</th> <th data-bbox="818 985 887 1037">"Right"</th> <th data-bbox="892 985 1050 1037">"Temperature"</th> <th data-bbox="1054 985 1139 1037">"Meam"</th> <th data-bbox="1144 985 1315 1037">"Atomic Weight"</th> </tr> </thead> <tbody> <tr> <td data-bbox="384 1043 651 1305">  </td> <td data-bbox="655 1043 740 1305">-1</td> <td data-bbox="745 1043 813 1305">[]</td> <td data-bbox="818 1043 887 1305">[0]</td> <td data-bbox="892 1043 1050 1305">-1</td> <td data-bbox="1054 1043 1139 1305">-1</td> <td data-bbox="1144 1043 1315 1305">"N/A"</td> </tr> </tbody> </table>	"Position"	"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"		-1	[]	[0]	-1	-1	"N/A"
"Position"	"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"									
	-1	[]	[0]	-1	-1	"N/A"									
5	<table border="1"> <thead> <tr> <th data-bbox="384 1368 651 1420">"Position"</th> <th data-bbox="655 1368 740 1420">"Value"</th> <th data-bbox="745 1368 813 1420">"Left"</th> <th data-bbox="818 1368 887 1420">"Right"</th> <th data-bbox="892 1368 1050 1420">"Temperature"</th> <th data-bbox="1054 1368 1139 1420">"Meam"</th> <th data-bbox="1144 1368 1315 1420">"Atomic Weight"</th> </tr> </thead> <tbody> <tr> <td data-bbox="384 1426 651 1688">  </td> <td data-bbox="655 1426 740 1688">-2</td> <td data-bbox="745 1426 813 1688">[]</td> <td data-bbox="818 1426 887 1688">[-1]</td> <td data-bbox="892 1426 1050 1688">-1</td> <td data-bbox="1054 1426 1139 1688">-2</td> <td data-bbox="1144 1426 1315 1688">"N/A"</td> </tr> </tbody> </table>	"Position"	"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"		-2	[]	[-1]	-1	-2	"N/A"
"Position"	"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"									
	-2	[]	[-1]	-1	-2	"N/A"									

6	<table border="1"> <thead> <tr> <th>"Position"</th> <th>"Value"</th> <th>"Left"</th> <th>"Right"</th> <th>"Temperature"</th> <th>"Meam"</th> <th>"Atomic Weight"</th> </tr> </thead> <tbody> <tr> <td>  </td> <td>*</td> <td>[0]</td> <td>[0]</td> <td>0</td> <td>0</td> <td>0</td> </tr> </tbody> </table>	"Position"	"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"		*	[0]	[0]	0	0	0
"Position"	"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"									
	*	[0]	[0]	0	0	0									
7	<table border="1"> <thead> <tr> <th>"Position"</th> <th>"Value"</th> <th>"Left"</th> <th>"Right"</th> <th>"Temperature"</th> <th>"Meam"</th> <th>"Atomic Weight"</th> </tr> </thead> <tbody> <tr> <td>  </td> <td>↑</td> <td>[0]</td> <td>[*]</td> <td>0</td> <td>0</td> <td>1</td> </tr> </tbody> </table>	"Position"	"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"		↑	[0]	[*]	0	0	1
"Position"	"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"									
	↑	[0]	[*]	0	0	1									
8	<table border="1"> <thead> <tr> <th>"Position"</th> <th>"Value"</th> <th>"Left"</th> <th>"Right"</th> <th>"Temperature"</th> <th>"Meam"</th> <th>"Atomic Weight"</th> </tr> </thead> <tbody> <tr> <td>  </td> <td>↑*</td> <td>[0,*]</td> <td>[0]</td> <td>0</td> <td>0</td> <td>1</td> </tr> </tbody> </table>	"Position"	"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"		↑*	[0,*]	[0]	0	0	1
"Position"	"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"									
	↑*	[0,*]	[0]	0	0	1									
9	<table border="1"> <thead> <tr> <th>"Position"</th> <th>"Value"</th> <th>"Left"</th> <th>"Right"</th> <th>"Temperature"</th> <th>"Meam"</th> <th>"Atomic Weight"</th> </tr> </thead> <tbody> <tr> <td>  </td> <td>↑</td> <td>[0]</td> <td>[↑*]</td> <td>0</td> <td>0</td> <td>2</td> </tr> </tbody> </table>	"Position"	"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"		↑	[0]	[↑*]	0	0	2
"Position"	"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"									
	↑	[0]	[↑*]	0	0	2									

10	<table border="1"> <thead> <tr> <th colspan="4">"Position"</th> <th>"Value"</th> <th>"Left"</th> <th>"Right"</th> <th>"Temperature"</th> <th>"Meam"</th> <th>"Atomic Weight"</th> </tr> </thead> <tbody> <tr> <td></td> <td>\uparrow^*</td> <td>[0]</td> <td>[↑]</td> <td>0</td> <td>0</td> <td>2</td> </tr> </tbody> </table>	"Position"				"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"		\uparrow^*	[0]	[↑]	0	0	2					
	"Position"				"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"													
	\uparrow^*	[0]	[↑]	0	0	2																	
11	<table border="1"> <thead> <tr> <th colspan="4">"Position"</th> <th>"Value"</th> <th>"Left"</th> <th>"Right"</th> <th>"Temperature"</th> <th>"Meam"</th> <th>"Atomic Weight"</th> </tr> </thead> <tbody> <tr> <td></td> <td>\uparrow^2</td> <td>[0]</td> <td>[↓*]</td> <td>0</td> <td>0</td> <td>0</td> </tr> </tbody> </table>	"Position"				"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"		\uparrow^2	[0]	[↓*]	0	0	0					
	"Position"				"Value"	"Left"	"Right"	"Temperature"	"Meam"	"Atomic Weight"													
	\uparrow^2	[0]	[↓*]	0	0	0																	

Table 8. The list of special game values of 4×4 NoGo.

4.4.2 Means and Temperatures

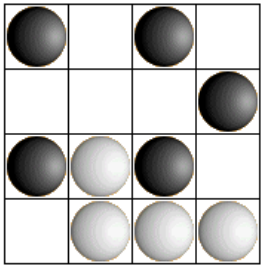
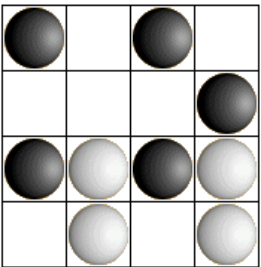
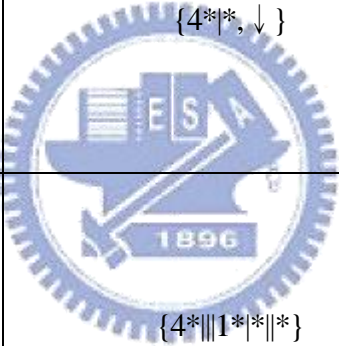
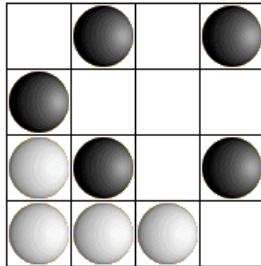
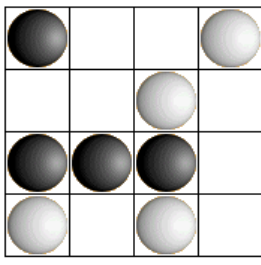
For analysis of hot games, we solved the game values of 89,964 and 21,454 4×4 NoGo positions with endgame search depth 6 and 8, respectively, in a brute force way. For positions with endgame search depth 6 of NoGo, the highest and lowest of means are $3\frac{1}{4}$, $-2\frac{1}{2}$, Game 12~13, respectively. For positions with endgame search depth 8 of NoGo, the highest and lowest of means are $2\frac{1}{4}$, $-1\frac{1}{2}$, Game 14~15, respectively. These NoGo positions are all shown as Table 9 (below).

ID	Position	Game Value	Temperature	Mean
12		$\{4^* 3^*2^*\}$	$\frac{3}{4}$	$3\frac{1}{4}$
13		$\{-2 -3\}$	$\frac{1}{2}$	$-2\frac{1}{2}$
14		$\{3^* 2 1\}$	$\frac{3}{4}$	$2\frac{1}{4}$
15		$\{-1\uparrow^*,\{^* -1,\{-1,-1^* -2\}\}-2\}$	$\frac{1}{2}$	$-1\frac{1}{2}$

Table 9. The highest and lowest of means with endgame search depth 6 and 8 of 4×4 NoGo

The maximum of temperature of all position is 2 with endgame search depth 6 of 4×4 NoGo positions as shown Table 10. There are totally 5 different positions whose temperatures are all the same and means may be different. The maximum of temperature of all position is $1\frac{3}{4}$ with endgame search depth 8 of 4×4 NoGo positions as shown Table 11. Note that the lowest temperatures are not listed since they are number games whose

temperatures are negative. There are also totally 6 different positions whose temperatures are all the same and means may be different.

ID	Position	Game Value	Temperature	Mean
16		$\{4^* 0\}$	2	2
17		 $\{4^* *, \downarrow\}$	2	2
18		$\{4^* 1^* ^*\}$	2	2
19		$\{2 1^* -5/2\}$	2	-1/2

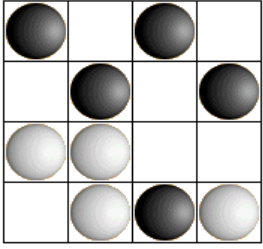
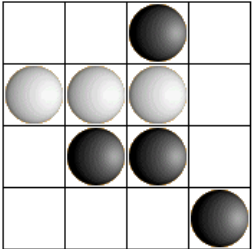
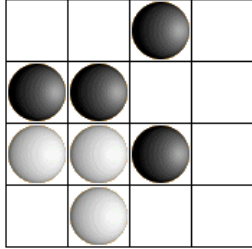

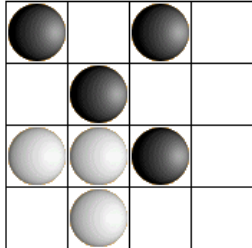
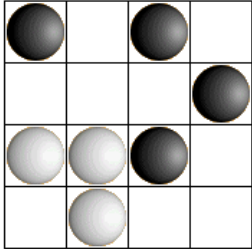
20		$\{3^* -1^*\}$	2	1
----	---	----------------	---	---

Table 10. The lists of maximum temperature with endgame search depth 6 in 4x4 NoGo.

ID	Position	Game Value	Temperature	Mean
21		$\{\{4 3\},\{4 3^*\} \{3 1\},\{3^* 1\} 2^* -1^*\}$	$1\frac{3}{4}$	$\frac{3}{4}$
22		 $\{3 2 -1\}$	$1\frac{3}{4}$	$\frac{3}{4}$
23		$\{3^* ^*,\{^*,\{2^* ^*\} \{^*,\downarrow -1\}\} -1^*,\{\{^* -1^*\},\{\downarrow -1^*\},\{^*,\downarrow -1\}\} -1 -2^*\},\{-1^* -2^*\}\}$	$1\frac{3}{4}$	$\frac{5}{4}$
24		$\{3^* \{^* -1^*\},\{^* \{-1/2^* -1\},\{-1/2^* -1^*\} -1,-1\downarrow^*\}$	$1\frac{3}{4}$	$\frac{5}{4}$

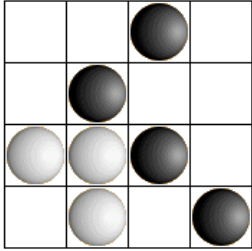
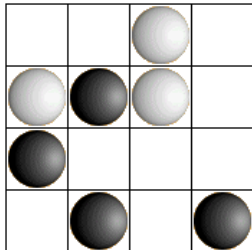
25		$\{3*\ \downarrow, \{*, \{1 0,*\} \} *-1*\}-1*\}$	$1\frac{3}{4}$	$\frac{5}{4}$
26		$\{3*\{*, \{\{1 0\}, \{1 1, \{1 0\}\} 0,*\}-1*\}, \{0 \{-1/2* -1\}, \{-1/2* -1*\}-1, -1\downarrow*\}\}$	$1\frac{3}{4}$	$\frac{5}{4}$

Table 11. The lists of maximum temperature with endgame search depth 8 in 4×4 NoGo.

In fact, the article in [15] also showed a specific 5×5 board NoGo position with temperature 2 as shown Figure 26. This position is favor for Black. If Black move a stone in the center grid, it will appear 4 B-Go in the board. The White cannot move any stone on the position. We calculate the game value of the specific position which the temperature is 2. If Black moves first, it has game value 4. Otherwise, if White first move, it has game value * (star). It is a hot game for both players. Both of players may try to compete the benefit in this position.

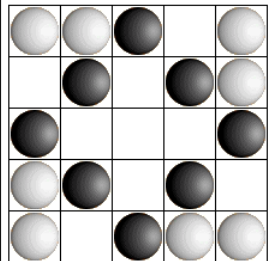
"Position"	"Value"	"Left"	"Right"	"Temperature"	"Meam"
	$\{4 *\}$	$[4]$	$[*]$	2	2

Figure 26. A specific 5×5 Nogo position.

In fact, there exists some NoGo position with the temperature greater than 2. Figure 27 shows such an example whose game value $G=\{7|6||2|1\}$ and temperature $2\frac{1}{2}$ is greater than 2. The upper bound of temperature is an open question.

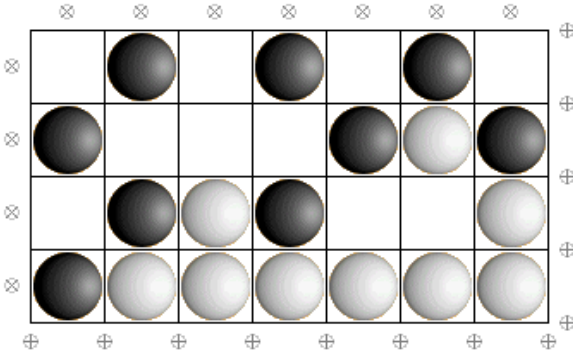


Figure 27. The temperature is greater than 2 in a 4x7 NoGo position.

4.4.3 More Analysis

We analyze, in total, 89,964 positions and 21,454 positions with endgame search depth 6 and 8 in 4x4 NoGo, respectively, in Table 12 and Table 13. In Table 12, the percentages of all positions that are numbers ($T<0$), infinitesimals ($T=0$) and hot games ($T>0$) are about 3.28%, 25.16% and 67.43%, respectively. In Table 13, the percentages of all positions that are numbers ($T<0$), infinitesimals ($T=0$), and hot games ($T>0$) are about 2.13%, 26.33% and 71.55%, respectively.

Infinitesimal (T=0)	Temperature						Total Positions
	T<0	Non- Infinitesimal (T=0)	0<T<1	1≤T<1.5	1.5≤T<2	T=2	
22639	2954	3719	43983	15930	734	5	89964
25.16%	3.28%	4.13%	48.89%	17.71%	0.82%	0.01%	100.00%

Table 12. The temperature analysis with endgame search depth 6 in 4×4 NoGo positions.

Infinitesimal (T=0)	Temperature						Total Positions
	T<0	Non- Infinitesimal (T=0)	0<T<1	1≤T<1.5	1.5≤T<2	T=2	
5648	456	519	11922	3353	75	0	21454
26.33%	2.13%	2.42%	55.57%	15.63%	0.35%	0.00%	100.00%

Table 13. The temperature analysis with endgame search depth 8 in 4×4 NoGo positions.

Moreover, we also investigate the game values of 3×3 and the 4×4 of empty boards are star (*) and zero (0), respectively. It represents that the first player wins in 3×3 NoGo and loses in 4×4 NoGo.

4.5. NoGo Propositions

In this section we show three important propositions using CGT analysis. It helps understand the characteristics of NoGo game. The propositions can efficiently reduce the depth and branches of search tree which improves the performance and helps determine who to win. Some of these propositions were used by a NoGo program, named HappyNoGo

[18][19], to improve the strength. HappyNoGo won the champion in the NoGo tournaments, such as 2013 Computer Olympiads in Japan, 2013 TCGA, 2012 TCGA and 2012 TAAI in Taiwan.

In order to investigate more propositions, we give more definitions as follows. Two grids are called *neighboring* if they are either vertical or horizontal neighboring. Among a set of grids, a grid is called to be *connected* to another, if there exists a sequence of neighboring grids between the two grids in the same set. A *connected component* is the maximum set of grids which are mutually connected. A *Black (White) string* is a connected component of Black (White) grids, where a Black (White) grid is a grid on which there exists a Black (White) stone. A *closed region* is a connected component of empty grids. A *Black (White) region* is a connected component of grids which are not Black (White) grids.

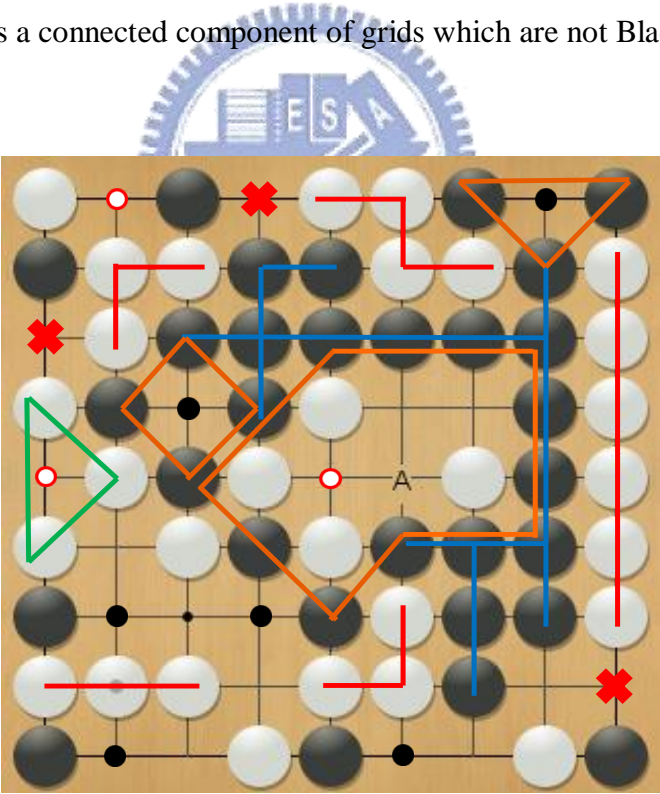


Figure 28. The example of NoGo definitions.

For example, in Figure 28 the stones of blue line are Black strings and the stones of red lines are White strings. The surrounded region of the stones of orange lines is Black closed

regions and the surrounded region of the stones of green lines are White closed region.

A *subgame* is defined to be a game where players can play on a designated region only, namely a set of designated grids. For a subgame played in a designated region, if its game value is independent of any moves on the grids not in this region, the subgame is called an *independent subgame*, and the region is called an *independent region*.

Proposition 1: (No-Go game value)

For a subgame on a No-Go, its game value is zero.

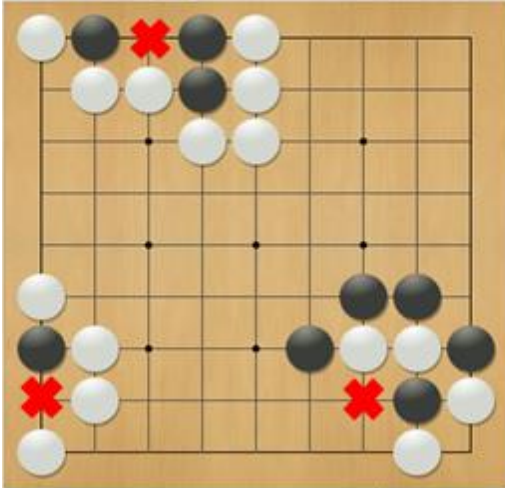


Figure 29. Each No-Go has game value zero.

In the No-Go of board position as shown Figure 29, it is trivial that the Black and White cannot place any stone on No-Go. Since No-Go cannot become 1-Go or 2-Go, the grid remains No-Go till the end of the game. Moreover, they are independent of any other No-Go which any available moves cannot change its state. So, the game value of No-Go is $\{\emptyset | \emptyset\} = 0$.

For CGT, the Number-Avoidance Theorem [2] states that one should only move on numbers as a last resort. When G is a game and x is a number, the sum can be simplified

as

$$G + x = \{G^L + x \mid G^R + x\} \quad (52)$$

$$x + G = \{x + G^L \mid x + G^R\} \quad (53)$$

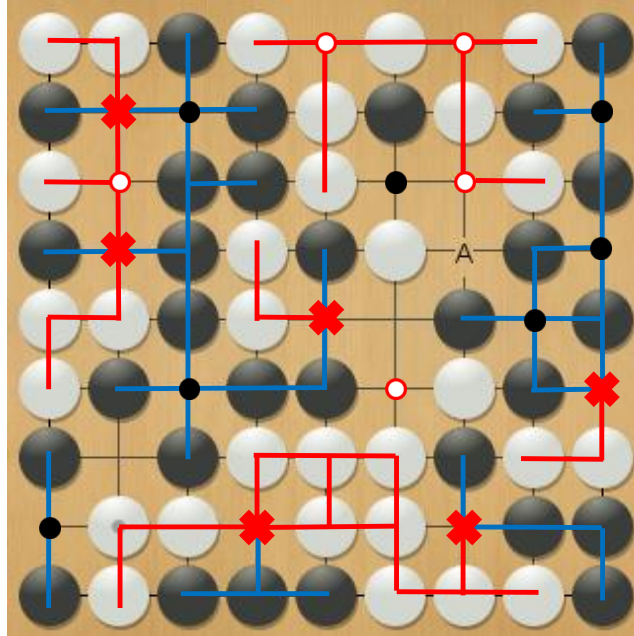


Figure 30. The example of B-Dragons and W-Dragons.

In order to investigate independent subgames, especially for those whose game values are numbers, we define more in the following:

- *B-Dragon*: It is a set of Black strings such that one string can be connected to another via No-Go or B-Go. Note that these No-Go and B-Go are not included in the B-Dragon. As illustrated in Figure 30, the Black strings, connected in blue lines, are B-Dragons.
- *W-Dragon*: It is a set of White strings such that one string can be connected to another via No-Go or W-Go. As illustrated in Figure 30, the White strings, connected in red lines, are W-Dragons. Note that B-Dragons and W-Dragons may intersect in No-Go.

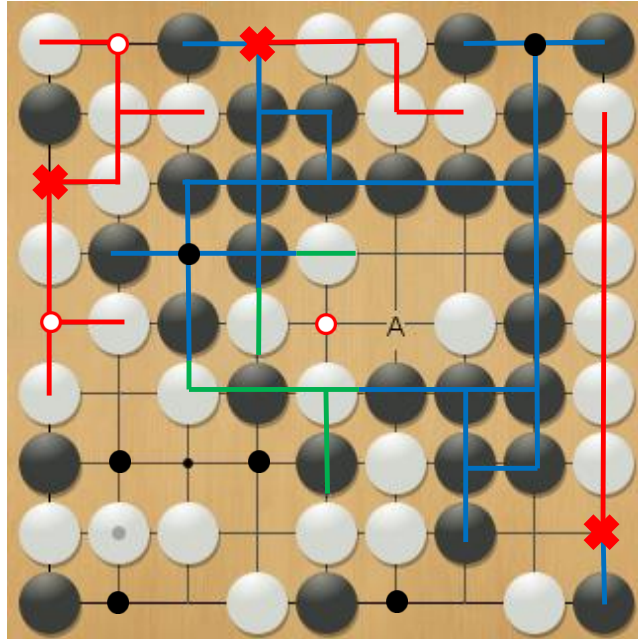


Figure 31. Three different walls in NoGo.

- *B-Wall*: A B-Dragon is also a B-Wall if the B-Dragon is adjacent to some No-Gos. As illustrated in Figure 31, the one connected in blue lines is also a B-Wall.
- *W-Wall*: A W-Dragon is also a W-Wall if the W-Dragon is adjacent to some No-Gos. As illustrated in Figure 31, the one connected in red lines is also a W-Wall.
- *Wall*: A Wall is one of No-Go, B-Wall and W-Wall.

Let us remove those grids on all the walls. Then, we can cut a position into several disjoint connected components of grids, named *group regions*. Proposition 2 (below) shows that each group region is also an independent region.

Proposition 2: (Independent region)

Each group region is an independent region. A subgame on each group region is an independent subgame. The game value is the sum of the values of all subgames.

Proposition 2 is satisfied for the following reason. For a string adjacent to a No-Go, assume that it neighbors to empty grids of more than two group regions, say R1 and R2. Since the string will not be captured due to a No-Go, all the moves on the empty grids of R1 will not affect the moves on those of R2, and vice versa. For a string adjacent to a 1-Go, say a Black string to a B-Go, assume similarly that it neighbors to empty grids of more than two group regions, say R1 and R2, and that the B-Go is in R2. All the moves on the empty grids of R1 will not affect the moves on those of R2 and vice versa. This is because the string will not be captured for the following reason. Even if the B-Go is occupied by Black stone, the string is still connected to a No-Go or another B-Go, and therefore can be proved recursively.

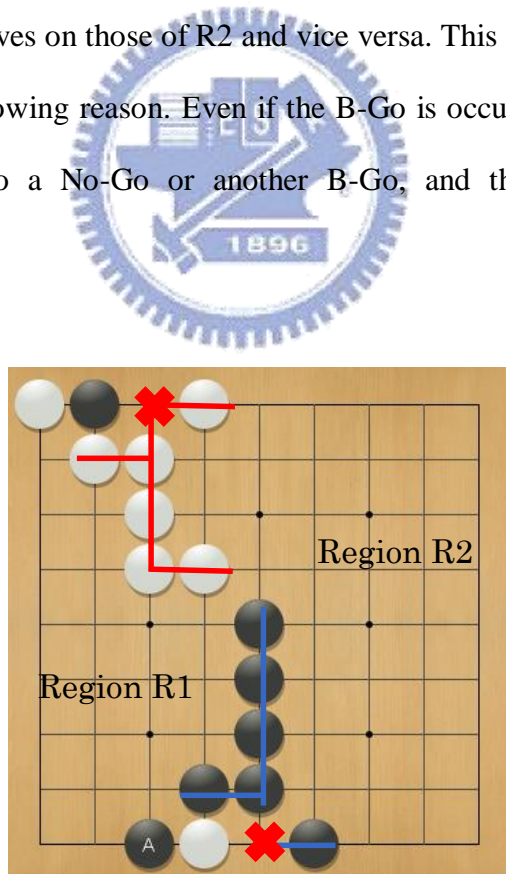


Figure 32. The board forms left and right independent regions.

Illustrated by Figure 32, we obtain that the group regions in the left and right regions

form two main independent regions.

Furthermore, we actually can cut it into more group regions, if we can extend the definition of Wall to including X-Wall. X-Wall is a Black or White string neighboring to one closed region only. The stones marked by green lines in Figure 31 are X-Walls. Proposition 2 is still true for the extension for the following reason. Since these strings are neighboring to one closed region only, all the moves in other group regions will not affect this group region.

From Proposition 2, the game value is the sum of the game values for all group regions. Here are some examples. If the game value of a subgame is zero, the group region can be ignored. If the game values on two regions are both star (*), the sum of the two regions is $*+*=0$. The Proposition 2 helps reduce the depth and branches in expanding search tree.

Unfortunately, the boards cannot partition the position and still leave a big region in most of game competitions. However, fortunately, we can easily identify those 1-Go with a game value 1 or -1. A B-Go is called B-eventually-Go or B-EGo, if Black will be able to place on the grid eventually (not become a No-Go), regardless of the subsequent moves. A W-EGo is defined similarly. Both are also called 1-EGo. Thus, we have Proposition 3 as below.

Proposition 3: (1-EGo)

A 1-Go is also a 1-EGo, if all of its neighbors are Walls. Each B-EGo (W-EGo) has game value 1 (-1).

Proposition 3 is correct for the following reason. Since all of its neighbors are Walls, the grid itself is a group region, an independent region. Since it is 1-Go, it will be played by one player only eventually. In addition, the game value is 1 (-1) if it is a B-EGo (W-EGo).

The example is illustrated as the black triangle in Figure 33. In this B-EGo, its game value is $\{ 0 | \emptyset \}=1$.

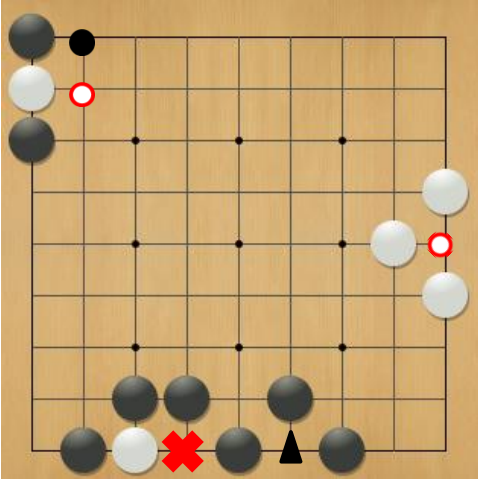


Figure 33. The symbol of black triangle is represented for B-EGo .

Similarly, each W-EGo has game value -1 for White. Below Figure 34 shows the example of B-EGo. Black has five triangles which gets the number of five and White has one triangle which gets the number of minus one. In Figure 34 the position is favor to Black.

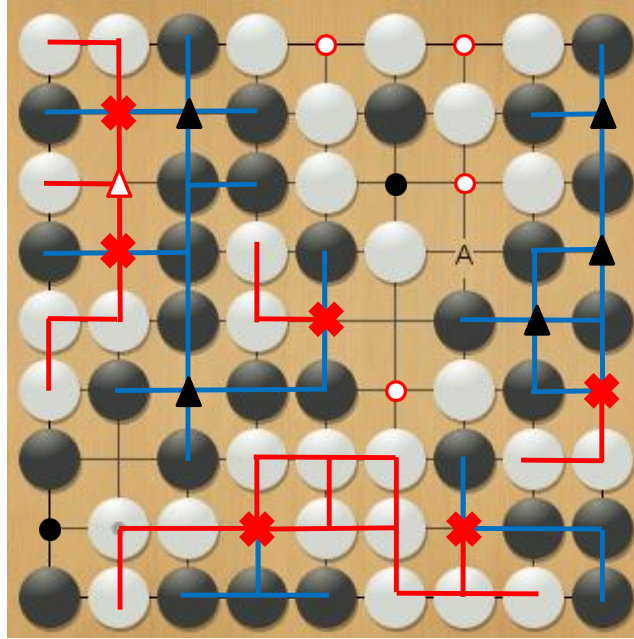


Figure 34. The example of B-Ego and W-Ego.

4.6. Conclusion

NoGo game analysis has the following three major contributions. First, we calculate the game values of many NoGo positions with endgame search depth 6 and 8. In 4×4 NoGo, the game values include integers such as 2, 1, 0, -1 and -2, and infinitesimals including star, ups and downs such as *, ↑, ↑*, ↑↑, ↑↑*, and ↑².

Second, we find the maximum of temperature is 2 among all the above 4×4 board positions and the temperature is low in most of these positions. There are five different positions whose temperatures are 2 and means may be different.

Third, we present three propositions to help understand the characteristics of NoGo game. Moreover, the propositions can efficiently reduce the depth and branches of search tree which improves the performance and helps determine who to win.

Chapter 5 Conclusions

In this thesis, we study to solve games and improve search performances with embedded combinatorial game knowledge. For this study, we investigate three combinatorial games, including Triangular Nim, XT Domineering and NoGo.

First, solving nine layer Triangular Nim has the following contributions. Using the retrograde methods, this thesis strongly solves 9 layer Triangular Nim. This thesis also proposes some methods to improve the performance, such as designing data structures in blocks, using the retrograde method, removing redundancy and selecting the block representation with the less number of inter-block updates. Especially, by removing redundancy, we reduce the memory by a factor of 5.72 and the computation time by a factor of 4.62.

Our experiment result also shows that the ratio of the number of P-positions to that of N-positions is low, 5.0% for 9 layer Triangular Nim. Due to the low ratio, the retrograde method does perform well when compared with the traditional forward checking.

Second, XT domineering has the following three major contributions. We present a new game, XT Domineering, which has higher game-tree complexity than Domineering. We also have presented a mathematical approach to solve sums of 3×3 XT Domineering. Again, this success demonstrates the potential of applying CGT to solving more of other intelligent games.

Third, NoGo endgame analysis has the following three major contributions. We calculate the game values of many NoGo positions with endgame search depth 6 and 8 in brute force search from empty 4×4 board. In NoGo 4×4 board, the game values include

integers such as 2, 1, 0, -1 and -2, and infinitesimals including star, ups and downs such as $*$, \uparrow , \uparrow^* , $\uparrow\uparrow$, $\uparrow\uparrow^*$, and \uparrow^2 . In experiments, we find the maximum of temperature is 2 among all the above 4×4 board positions and the temperature is low in most of these positions. Furthermore, we present three propositions to help understand the characteristics of NoGo game.

In this thesis, we use algebra characteristics of CGT to help solve and reduce the complexity of three combinatorial games. Based on the theory, playing or solving these combinatorial games may simply become mathematical calculations, such as summation, instead of a complex tree search. CGT also helps cut some unnecessary branches in search tree in some combinatorial games, e.g. NoGo.



References

- [1] M. H. Albert, J. P. Grossman, R. J. Nowakowski and D. Wolfe, “An introduction to Clobber,” *Integers*, Electr. J of Combinat. Number Theory vol. 5, no. 2, #A01, 2005.
- [2] M. H. Albert, R. J. Nowakowski, D. Wolfe, *Lessons in Play: An Introduction to the Combinatorial Game Theory*, A.K. Peters, Wellesley (2007)
- [3] S.-Q. Bai, S.-S. Lin, “On the Study of 8 Layer Triangular Nim,” (in Chinese) *National Computer Symposium (NCS 2009)*, Taipei, Taiwan, 2009.
- [4] R. Balla, A. Fern, UCT for tactical assault planning in real-time strategy games, in: *21st International Joint Conference on Artificial Intelligence*, pp. 40–45.
- [5] E. R. Berlekamp, “Blockbusting and Domineering,” *Journal of Combinatorial Theory Ser. A*, vol. 49, pp. 67-116, 1988.
- [6] E. R. Berlekamp, D. Wolfe, *Mathematical Go: Chilling Gets the Last Point*, A K Peters, Wellesley, MA, 1994.
- [7] E. R. Berlekamp, J. H. Conway, and R. K. Guy, *Winning Ways for your Mathematical Plays*, 1st edition, 2 vols, New York: Academic Press, 1982.
- [8] C. L. Bouton, “Nim, A Game with a Complete Mathematical Theory,” *the Annals of Mathematics*, 2nd Ser., Vol.3, 1/4. (1901-1902), pp.35-39.
- [9] D. M. Breuker, J. W. H. M. Uiterwijk and H. J. van den Herik, “Solving 8×8 Domineering,” *Theoretical Computer Science*, Vol. 230, pp. 195-206, 2000.
- [10] B. Brian, *A Mathematical Pandora’s Box*, Cambridge Univ Pr, 1993.
- [11] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, S. Colton, “A survey of Monte Carlo Tree Search,” *IEEE Transactions On Computational Intelligence and AI in Games*, Vol. 4, No. 1, March 2012.
- [12] N. Bullock, “Domineering: Solving Large Combinatorial Search Spaces,” *ICGA Journal*, vol. 25, no. 2, pp. 67-84, 2002.
- [13] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.

- [14] C.-Y. Chen. S.-S. Lin, *On the Study and Improvement of 8 Layer and 9 Layer Triangular Nim*, Master Thesis, National Taiwan Normal University, 2010.
- [15] C.-W. Chou, O. Teytaud, S.-J. Yen, "Revisiting Monte-Carlo tree search on a normal form game: NoGo," in: *The main European events on Evolutionary Computation (Evo/ 2011)*, Torino, Italy, April 27–29, 2011.
- [16] A. Cincotti, "Three-Player Domineering," *Proceedings of World Academy of Science, Engineering, and Technology* 36:92-95, December 2008.
- [17] A. Cincotti, "Further Results on Three-Player Domineering," *Proceedings of World Academy of Science, Engineering and Technology* 51:187-189, 2009.
- [18] Computer game tournaments in *the 2012 Conference on Technologies and Applications of Artificial Intelligence (TAAI 2012)*,
<http://idb.csie.ncku.edu.tw/taai2012conference/index.php/awards/tournaments>.
- [19] *Computer Olympiad 2013*, in Japan, Yokohama , http://icga.uvt.nl/?page_id=627.
- [20] J. H. Conway, *On Numbers and Games*, New York: Academic Press, 1976.
- [21] R. Coulom, "Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search", in: *5th International Conference on Computer and Games*, pp. 72–83.
- [22] E. D. Demaine, "Playing games with algorithms: algorithmic combinatorial game theory," in: *Mathematical Foundations of Computer Science (Mariánské Lázně)*, *Lecture Notes in Comput. Sci.* 2136, Springer, Berlin (2001), pp. 18–32.
- [23] N. D. Elkies. On numbers and endgames: combinatorial game theory in chess endgames. In R. J. Nowakowski, editor, *Games of No Chance, Proc. MSRI Workshop on Combinatorial Games*, July, 1994, Berkeley, CA, MSRI Publ., volume 29, pages 135-150. Cambridge University Press, Cambridge, 1996.
- [24] T. S. Ferguson, *Game Theory*, lecture notes, UCLA 2008.
- [25] H. Finnsson, Y. Björnsson, "Simulation-based approach to general game playing," in: *23rd Conference on Artificial Intelligence*, pp. 259–264.
- [26] R. Fleischer and S. U. Khan. Xinagqi and combinatorial game theory. *Technical Report HKUST-TCSC-2002-01*, Hong Kong University of Science and Technology, February 2002.
- [27] M. Gardner, "Mathematical games," *Scientific American*, Vol. 230, pp. 106-108, 1974.

- [28] S. Gelly, “A Contribution to Reinforcement Learning; Application to Computer Go”, Ph.D. Thesis, University of South Paris, 2007.
- [29] S. Gelly, D. Silver, “Achieving master level play in 9 x 9 computer Go,” in: *23rd Conference on Artificial Intelligence*, 2008, pp. 1537–1540.
- [30] S. Gelly, D. Silver, “Monte-Carlo Tree Search and Rapid Action Value Estimation in Computer Go,” *Artificial Intelligence*, Volume 175, Issue 11, pp. 1856-1875, 2011.
- [31] P. M. Grundy, “Mathematics and games,” *Eureka* 2, pp.6-8, 1939, Reprint, *Eureka* 27, pp. 9-11, 1964.
- [32] O. Hanner, (1959). “Mean Play for Sums of Positional Games,” *Pacific J. Math.* 9, pp. 81-99.
- [33] S.-C. Hsu, “Solving the Problem of 7 Layer of Triangular Nim,” (in Chinese) *National Computer Symposium (NCS 1985)*, pp.798-802, Dec. 1985.
- [34] T.-S. Hsu. and P.-Y. Liu, "Verification of Endgame Databases," *International Computer Game Association (ICGA) Journal*, volume 25, number 3, pp. 132-144, 2002.
- [35] 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.
- [36] K.-Y. Kao, “On Hot and Tepid Combinatorial Games,” Ph.D. Thesis, UNC Charlotte 1997.
- [37] K.-Y. Kao, (1998). “Mean and Temperature Search for Combinatorial Games,” *JCIS Proceedings*, Vol I, pp. 389-392.
- [38] K.-Y. Kao, “The game of un-impartial Kayles,” *Proceedings of the 25th Workshop on Combinatorial Mathematics and Computation Theory*, Chung Hua University, Hsinchu Hsien, Taiwan, April 25-26, 2008, pp. 151-153.
- [39] K. Y. Kao, “Sumbers – Sums of Ups and Downs,” *Integers*, E. Journal of Combinatorial Number Theory, Vol. 2005.
- [40] K.-Y. Kao, I.-C. Wu, Y.-C. Shan and H.-H. Lin, “Chilled Domineering,” *International Conference on Technologies and Applications of Artificial Intelligence (TAAI 2010)*, pp.427-432.
- [41] K.-Y. Kao, I.-C. Wu, S.-J. Yen and Y.-C. Shan, "Incentive Learning in Monte Carlo Tree Search," *IEEE Transactions on Computational Intelligence and AI in Games (IEEE*

TCIAIG), Feb 2013.

- [42] K.-Y. Kao, I.-C. Wu, Y.-C. Shan, and S.-J. Yen), "Selection Search for Mean and Temperature of Multi-Branch Combinatorial Games," *ICGA Journal*, Vol. 35, No. 3, pp. 157-176, Sep. 2012.
- [43] K.-Y. Kao, I.-C. Wu, and Y.-C. Shan, "XT Domineering: A New Combinatorial Game," *Knowledge-Based Systems*, Volume 34, pp. 55-63, October 2012.
- [44] M. Lachmann, C. Moore, I. Rapaport, "Who Wins Domineering on Rectangular Boards," in *More Games of No Chance*, R.J. Nowakowski, Ed. Cambridge University Press, 2002, pp. 307-315.
- [45] C.-S. Lee, M.-H. Wang, Y.-J. Chen, H. Hagrais, M.-J. Wua, O. Teytaud, "Genetic fuzzy markup language for game of NoGo," *Knowledge-Based Systems*, Volume 34, pp. 64-80, October 2012.
- [46] L. Lew, and Coulom, R. (2010). "Simulation-based Search of Combinatorial Games," in *ICML Workshop on Machine Learning and Games*, Israel.
- [47] H.-H. Lin, I.-C. Wu and Y.-C. Shan, "Solving Eight Layer Triangular Nim," (in Chinese) *National Computer Symposium (NCS 2009)*, Taipei, Taiwan, November 2009.
- [48] R. Lorentz, "Amazons discover Monte-Carlo," in: *6th International Conference on Computers and Games*, pp. 13-24.
- [49] F. Mäser, Global threats in combinatorial games: a computational model with applications to chess endgames, in: *More Games of No Chance, Proc. MSRI Workshop on Combinatorial Games*, July, 2000, Berkeley, CA, MSRI Publ. (R. J. Nowakowski, ed.), Vol. 42, Cambridge University Press, Cambridge, pp. 137-149.
- [50] S. K. McCurdy and R. Nowakowski, "Cutthroat, an all-small game on graphs," *Integers*, Electr. J of Combinat. Number Theory vol. 5, no. 2, #A13, 2005.
- [51] G. A. Mesdal, "Partzan Splittles," *Games of No Chance 3*, Cambridge University Press, 2009, pp 447-461.
- [52] J. Milnor, (1953). "Sums of Positional Games," in Kuhn and Tucker (eds.) "*Contributions to the Theory of Games*," Ann. Math. Studies #28, Princeton, pp. 291-301.
- [53] M. Müller, "Computer Go," *Artificial Intelligence*, Vol.134, No.1-2 (Special issue on Games, Computers and AI), January 2002, pp145-179.

- [54] M. Müller, Enzenberger, M., and Schaeffer, J., (2004). "Temperature discovery search," In *AAAI*, pp. 658–663, San Jose, CA.
- [55] M. Müller, NoGo History and Competitions,
<http://webdocs.cs.ualberta.ca/~mmueller/nogo/history.html>
- [56] J. Schäfer, "The UCT algorithm applied to games with imperfect information," Diploma Thesis. Otto-von-Guericke-Universität at Magdeburg, 2008.
- [57] Y.-C. Shan, I.-C. Wu, H.-H. Lin, and K.-Y. Kao, "Solving Nine Layer Triangular Nim," *Journal of Information Science and Engineering*, vol.28, No.1, pp.99-113, January, 2012.
- [58] C. E. Shannon. "Programming a Computer for Playing Chess," in *Philosophical Magazine*, 7th series, 41, NO. 314 (March 1950): 256-75.
- [59] A. N. Siegel, *CGSuite*, A java based toolkit for evaluating games, <http://www.cgsuite.org/>
- [60] R. P. Sprague, "Über mathematische Kampfspiele," *Tohoku Mathematical Journal* 41, pp. 438-444, 1936.
- [61] N. Sturtevant, "An analysis of UCT in multi-player games," in: *6th International Conference on Computers and Games*, pp. 37–49, 2008.
- [62] F. Teytaud, O. Teytaud, "Creating an Upper Confidence Tree program for Havannah," in: *12th Advances in Computer Games Conference*, pp. 65–74, 2010.
- [63] K. Thompson. "Retrograde analysis of certain endgames," *ICCA Journal*, Vol. 9, No. 3, pp.131-139, 1986.
- [64] K. Thompson. "6-Piece Endgames," *ICCA Journal*, Vol. 19, No. 4, pp. 215-226, 1996.
- [65] A. Tucker, *Applied Combinatorics*, 3rd edition, New York: Wiley, pp396-403, 1994.
- [66] M. Winands, Y. Björnsson, "Evaluation function based Monte-Carlo LOA," in: *12th Advances in Computer Games Conference*, pp. 33–44.
- [67] I.-C. Wu, C.-P. Chen, P.-H. Lin, G.-Z. Huang, L.-P. Chen, D.-J. Sun, Y.-C. Chan, and H.-Y. Tsou, "A Volunteer-Computing-Based Grid Environment for Connect6 Applications," *The 12th IEEE International Conference on Computational Science and Engineering (CSE-09)*, August 29-31, Vancouver, Canada, 2009.
- [68] I.-C. Wu, Y.-C. Shan, C.-H. Lin and S.-J. Yen, "LongCATMJ Wins Mahjong Tournament in TCGA 2011," *ICGA Journal*, vol. 34, no. 3, 2011, pp. 166–167.

- [69] I.-C. Wu, C.-H. Lin, Y.-C. Shan, "Tournament Framework for Computer Mahjong Competitions," *The International Workshop on Computer Games (IWCG 2011)*, Chunli, Taiwan, November 2011.
- [70] P.-H. Wu, P.-Y. Liu and T.-S. Hsu, "An External-Memory Retrograde Analysis Algorithm," *Proc. 4th International Conference on Computers and Games (CG)*, Springer-Verlag LNCS# 3846, pp. 145-160, 2004.



Appendix A The Derivations for XT Domineering

Games Values of Positions

The power of using combinatorial theory is to derive the game value (or result) without tree search as many board games do. This is well described in many articles such as [1][50]. In this appendix, a simple Domineering example with C and E in Figure 18 as well as a XT Domineering example is illustrated to demonstrate the power of using combinatorial theory.

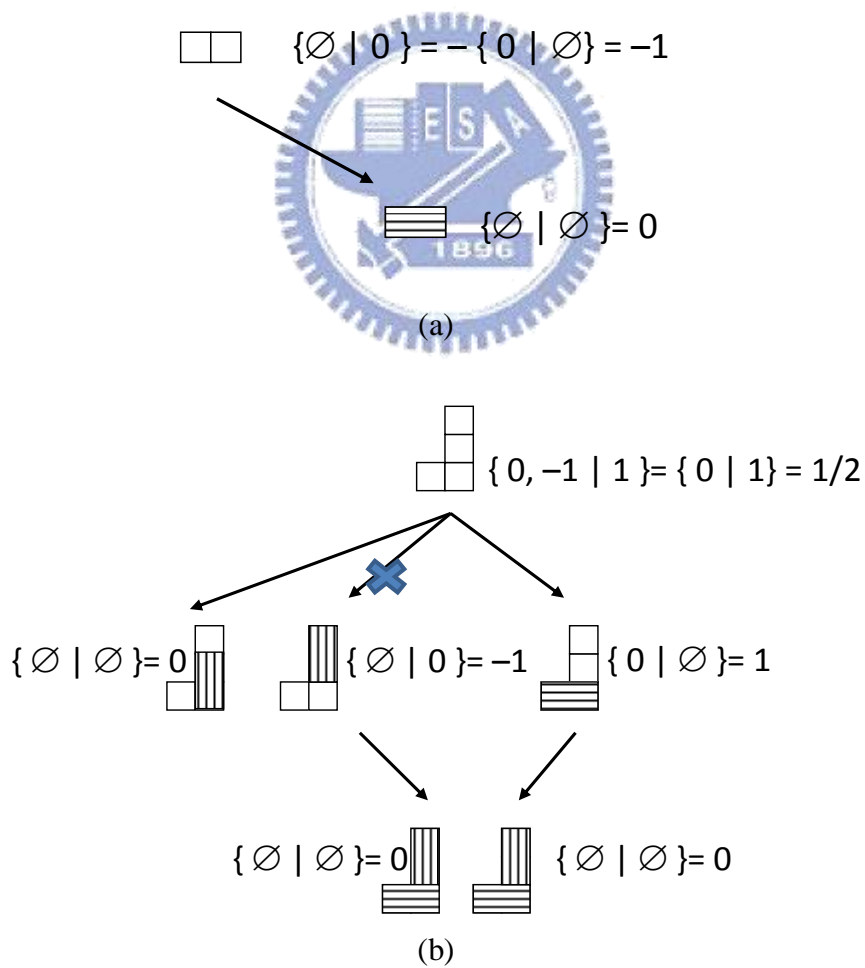


Figure 35. Deriving both game values of C and E of Figure 18 in (a) and (b) respectively.

First, let us investigate the game of Domineering. The game value of C , -1 , is derived in Figure 35 (a). The negative game value indicates that Right wins the game. The game value of E , $1/2$, is derived in Figure 35 (b). The positive value indicates that Left wins the game. In the derivation, a cross is used to indicate that Left does not choose -1 since choosing 0 is better to Left.

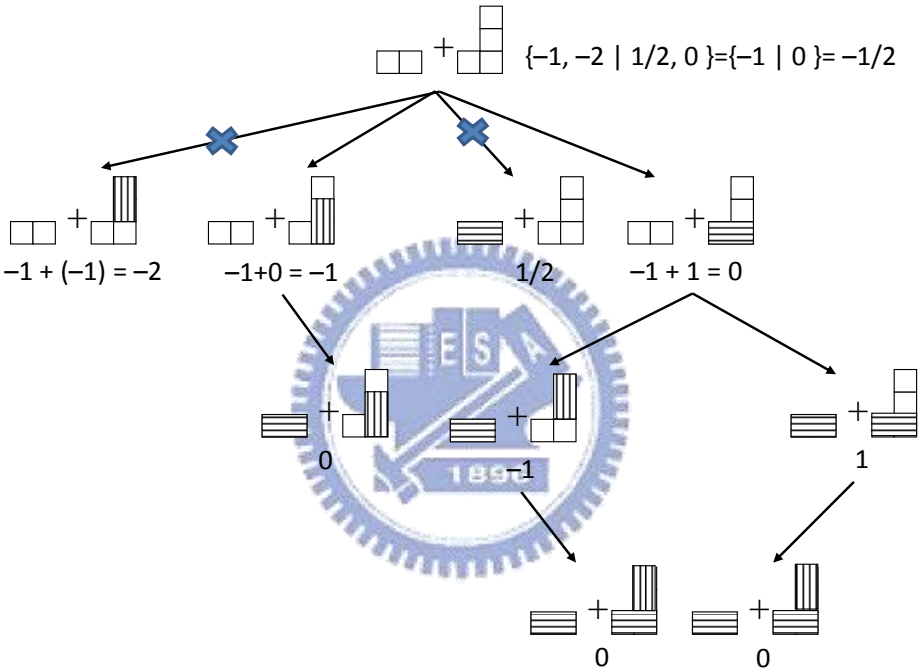
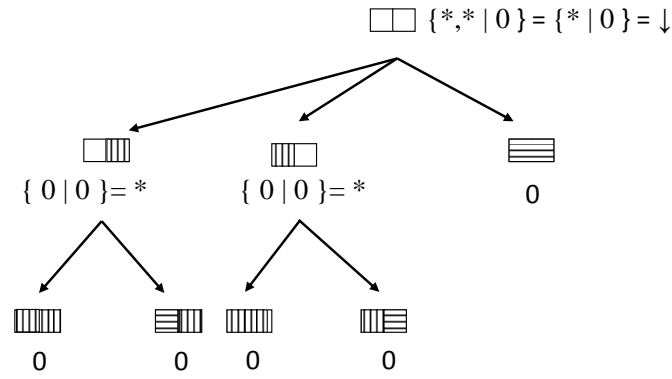
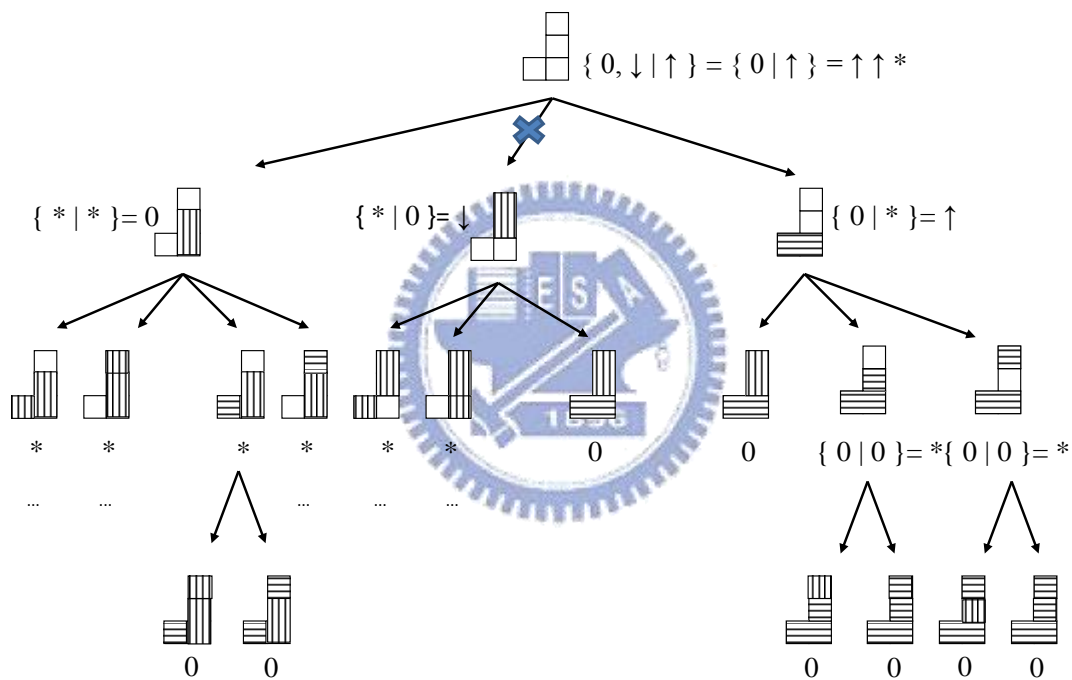


Figure 36. Deriving the game values of $C + E$.

If both C and E are left in a game, we can derive the game value, $-1 + 1/2 = -1/2$, by using the combinatorial theory, and easily conclude that Right wins the game due to the negative game value. However, in case of using tree search, we need to derive the same game value as shown in Figure 36, whose computational complexity grows exponentially as more are added.



(a)



(b)

Figure 37. Deriving both game values of C and E of Figure 18 in (a) and (b) respectively.

Now, let us investigate the game of XT Domineering. As described in Section 3.3, the game becomes more complex since 1×1 dominos are also allowed to be placed. For both games C and E , the derivations for both are shown in Figure 37 (a) and (b), respectively. The game value of C , \downarrow (a negative infinitesimal), indicates that Right still wins the game,

while the game value of E , $\uparrow\uparrow^*$, indicates that Left wins the game. The derivations for both are clearly much more complex, when compared with Figure 35.

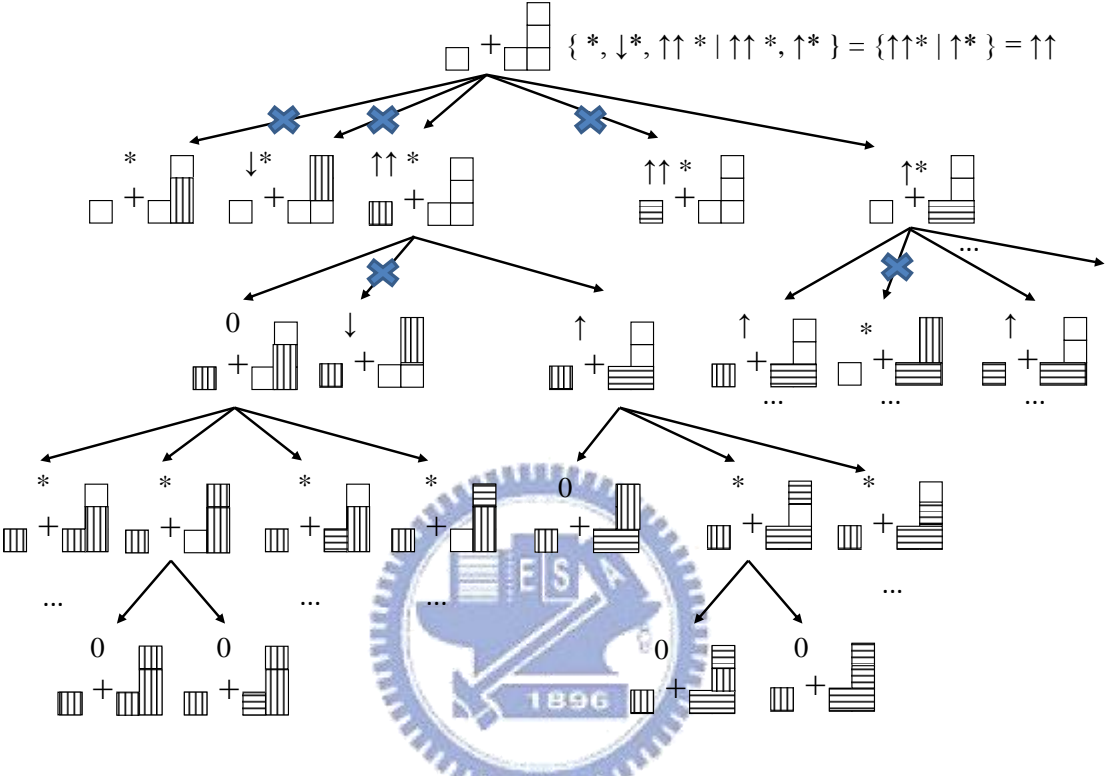


Figure 38. Deriving the game values of $C + E$.

For simplicity, we choose the game $* + E$, as shown in Figure 38. Its game value is $\uparrow\uparrow$ with atomic weight 2, which indicates that Left wins the game, as described in 3.2.4.

Appendix B The proof for XT Domineering inequalities

Proposition: The ups in the grids of Table 7 are the sufficient and necessary conditions for

$$\text{grid}(i, j) + \text{row}(i) + \text{col}(j) > 0.$$

Proof:

Let $(G_{i,j})$ denote the inequality

$$\text{grid}(i, j) + \text{row}(i) + \text{col}(j) > 0.$$

We first show the sufficiency of the conditions.

- Since $\uparrow > \uparrow/2 > \uparrow^2 > 0$, we have $(G_{3,1})$, $\uparrow^2 > 0$ and $\uparrow/2 - \uparrow^2 > 0$.
- Since $* / 2 + \triangle_* + \uparrow/2 > 0$, we have $(G_{7,1})$.
- Since $* / 2 + \uparrow - \uparrow^2 > 0$ and $* / 2 + \uparrow/2 + 2 \cdot \uparrow^2 > 0$, we have $(G_{8,1})$.
- Since $* / 2 + \uparrow/2 + \uparrow^2 > \star$, we have $(G_{8,2})$.
- Since $\triangle_* + \triangle_{*} > *$, we have $(G_{1,3})$, and

$$(G_{7,1}) \Rightarrow (G_{5,3}),$$

$$(G_{8,1}) \Rightarrow (G_{6,3}), \text{ and}$$

$$(G_{8,2}) \Rightarrow (G_{6,4}).$$

- Since $* / 2 + \uparrow/2 + \uparrow + \uparrow^2 > \star + *$, we have $(G_{8,4})$.

$$(G_{8,2}) \text{ and } (G_{8,4}) \Rightarrow (G_{9,2}) \text{ and } (G_{9,4}).$$

- Since $\uparrow^2 > \star$ and $\uparrow + \uparrow^2 > \star + *$, we have $(G_{3,2})$ and $(G_{3,4})$.

$$(G_{3,2}) \text{ and } (G_{3,4}) \Rightarrow (G_{4,2}) \text{ and } (G_{4,4}).$$

- Since $\uparrow^2 > \star$, we have $(G_{3,2})$, and

$$(G_{4,2}) \Rightarrow (G_{4,1}),$$

$$(G_{9,2}) \Rightarrow (G_{9,1}),$$

$$(G_{1,3}) \Rightarrow (G_{1,4}),$$

$$(G_{4,4}) \Rightarrow (G_{4,3}), \text{ and}$$

$$(G_{9,4}) \Rightarrow (G_{9,3}).$$

- Since $\triangle_{*} > 0$, we have $(G_{2,1})$, and

$$(G_{2,1}) \Rightarrow (G_{1,1})$$

$$(G_{7,1}) \Rightarrow (G_{6,1}) \Rightarrow (G_{5,1}),$$

$$(G_{3,2}) \Rightarrow (G_{2,2}) \Rightarrow (G_{1,2}),$$

$$(G_{8,2}) \Rightarrow (G_{7,2}) \Rightarrow (G_{6,2}) \Rightarrow (G_{5,2}),$$

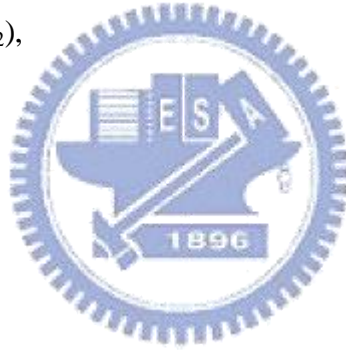
$$(G_{4,3}) \Rightarrow (G_{3,3}) \Rightarrow (G_{2,3}),$$

$$(G_{9,3}) \Rightarrow (G_{8,3}) \Rightarrow (G_{7,3}),$$

$$(G_{3,4}) \Rightarrow (G_{2,4}),$$

$$(G_{6,4}) \Rightarrow (G_{5,4}), \text{ and}$$

$$(G_{8,4}) \Rightarrow (G_{7,4}).$$



This completes proof for the sufficiency of the conditions.

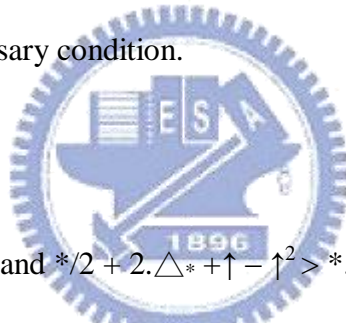
Next, we prove the necessary of the conditions. We need to show that any sums of ups less than or confused with the value in a corresponding grid will result in an insufficient condition. Note that the smallest increments of sums ups are \uparrow^2 and $\uparrow/2 - \uparrow^2$, and the only possible sums of ups confusing with 0 are $\uparrow/2 - (n + 1) \cdot \uparrow^2$, $n > 0$.

For $(G_{3,1})$, $(G_{6,3})$ and $(G_{8,1})$, we only need to show that if the value in the corresponding grid reduced by \uparrow^2 , then the inequality will not hold. For all the other grids, in order to prove the necessary conditions, we need to show that if the value in a grid reduced by \uparrow^2 or $\uparrow/2 - \uparrow^2$, or, if the value in a grid increased or reduced by $\uparrow/2 - (n + 1).\uparrow^2$, $n > 0$, then the corresponding inequality will not hold. Since $\uparrow/2 - 2.\uparrow^2 \geq \uparrow/2 - (n + 1).\uparrow^2 > -(\uparrow/2 - \uparrow^2) > -(\uparrow/2 - (n + 1).\uparrow^2)$, it is sufficient to show: if the value in a grid reduced by \uparrow^2 or increase by $\uparrow/2 - 2.\uparrow^2$ then the corresponding inequality will not hold.

- Consider $(G_{3,1})$, $\uparrow^2 > 0$ and $\uparrow/2 - \uparrow^2 > 0$.

But $0 \not\geq 0$ and $\uparrow/2 - 2.\uparrow^2 \not\geq 0$.

Thus \uparrow^2 or $\uparrow/2 - \uparrow^2$ is a necessary condition.



- Consider $(G_{6,3})$,

$*/2 + 2.\Delta_* + \uparrow/2 + 2.\uparrow^2 > *$ and $*/2 + 2.\Delta_* + \uparrow - \uparrow^2 > *$.

But $*/2 + 2.\Delta_* + \uparrow/2 + \uparrow^2 \not\geq *$ and $*/2 + 2.\Delta_* + \uparrow - 2.\uparrow^2 \not\geq *$.

Thus $\uparrow/2 + 2.\uparrow^2$ or $\uparrow - \uparrow^2$ is a necessary condition.

Note that, since $2.\Delta_* > *$, the necessary condition of $(G_{6,3})$ implies the necessary condition of $(G_{8,1})$.

- Consider $(G_{1,2})$, $(n + 1).\Delta_* + \uparrow^2 > \star$.

But $(n + 1).\Delta_* \not\geq \star$ and $(n + 1).\Delta_* + \uparrow/2 - \uparrow^2 \not\geq \star$

Thus \uparrow^2 is a necessary condition.

- Consider $(G_{1,4}), (n+1).\Delta_* + \uparrow^2 > \star + *$.

But $(n+1).\Delta_* \not\geq \star + *$ and $(n+1).\Delta_* + \uparrow/2 - \uparrow^2 \not\geq \star + *$

Thus \uparrow^2 is a necessary condition.

- Consider $(G_{2,3}), \Delta_* + \uparrow + 2.\uparrow^2 > *$.

But $\Delta_* + \uparrow + \uparrow^2 \not\geq *$ and $\Delta_* + \uparrow + \uparrow/2 \not\geq *$

Thus $\uparrow + 2.\uparrow^2$ is a necessary condition.

- Consider $(G_{4,1}), -n.\Delta_* + \uparrow + 2.\uparrow^2 > 0$.

But $-n.\Delta_* + \uparrow + \uparrow^2 \not\geq 0$ and $-n.\Delta_* + \uparrow + \uparrow/2 \not\geq 0$

Thus $\uparrow + 2.\uparrow^2$ is a necessary condition.

- Consider $(G_{5,2}), */2 + n.\Delta_* + \uparrow/2 + \uparrow^2 > \star$.

But $*/2 + n.\Delta_* + \uparrow/2 \not\geq \star$ and $*/2 + n.\Delta_* + \uparrow - \uparrow^2 \not\geq \star$

Thus $\uparrow/2 + \uparrow^2$ is a necessary condition.

- Consider $(G_{5,4}), */2 + (n+2).\Delta_* + \uparrow/2 + \uparrow^2 > \star + *$.

But $*/2 + (n+2).\Delta_* + \uparrow/2 \not\geq \star + *$ and

$*/2 + (n+2).\Delta_* + \uparrow - \uparrow^2 \not\geq \star + *$.

Thus $\uparrow/2 + \uparrow^2$ is a necessary condition.

- Consider $(G_{7,3}), */2 + \Delta_* + \uparrow/2 + \uparrow + 2.\uparrow^2 > *$.

But $*/2 + \Delta_* + \uparrow/2 + \uparrow + \uparrow^2 \not\geq *$ and $*/2 + \Delta_* + 2.\uparrow \not\geq *$.

Thus $\uparrow/2 + \uparrow + 2.\uparrow^2$ is a necessary condition.

- Consider $(G_{9,1})^*, \frac{1}{2} - n \cdot \Delta + \frac{1}{2} + \frac{1}{2} + 2 \cdot \uparrow^2 > 0$.

But $\frac{1}{2} - n \cdot \Delta + \frac{1}{2} + \frac{1}{2} + \uparrow^2 \not\geq 0$ and $\frac{1}{2} - n \cdot \Delta + 2 \cdot \uparrow \not\geq 0$.

Thus $\frac{1}{2} + \frac{1}{2} + 2 \cdot \uparrow^2$ is a necessary condition.

Let $(G_{i,j})^*$ denote the inequalities

$grid(i, j) + row(i) + col(j) - \uparrow^2 \not\geq 0$, and

$grid(i, j) + row(i) + col(j) + \frac{1}{2} - 2 \cdot \uparrow^2 \not\geq 0$

- Since $\uparrow^2 > \star$, we have

$$(G_{1,2})^* \Rightarrow (G_{1,1})^*,$$

$$(G_{1,4})^* \Rightarrow (G_{1,3})^*,$$

$$(G_{2,3})^* \Rightarrow (G_{2,4})^*,$$

$$(G_{4,1})^* \Rightarrow (G_{4,2})^*,$$

$$(G_{5,2})^* \Rightarrow (G_{5,1})^*,$$

$$(G_{5,4})^* \Rightarrow (G_{5,3})^*,$$

$$(G_{7,3})^* \Rightarrow (G_{7,4})^*, \text{ and}$$

$$(G_{9,1})^* \Rightarrow (G_{9,2})^*.$$



- Since $\Delta_* > 0$, we have

$$(G_{1,1})^* \Rightarrow (G_{2,1})^*,$$

$$(G_{5,1})^* \Rightarrow (G_{6,1})^* \Rightarrow (G_{7,1})^*,$$

$$(G_{1,2})^* \Rightarrow (G_{2,2})^* \Rightarrow (G_{3,2})^*,$$

$$(G_{5,2})^* \Rightarrow (G_{6,2})^* \Rightarrow (G_{7,2})^* \Rightarrow (G_{8,2})^*,$$

$$(G_{2,3})^* \Rightarrow (G_{3,3})^* \Rightarrow (G_{4,3})^*,$$

$$(G_{7,3})^* \Rightarrow (G_{8,3})^* \Rightarrow (G_{9,3})^* ,$$

$$(G_{2,4})^* \Rightarrow (G_{3,4})^* \Rightarrow (G_{4,4})^* ,$$

$$(G_{5,4})^* \Rightarrow (G_{6,4})^* , \text{ and}$$

$$(G_{7,4})^* \Rightarrow (G_{8,4})^* \Rightarrow (G_{9,4})^* .$$

This completes the proof for the necessary of the conditions.



Vita

Yi-Chang Shan was born in Taipei, Taiwan in 1969. He received the B.S. and M.S. degrees in Computer Science from National Taiwan Normal University, in 1991 and 2002, respectively, and Ph.D. degree in Institute of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, in 2013. His research interests include computer game, combinatorial game theory, and cloud computing.

