國立台灣師範大學
資訊工程研究所碩士論文

指導教授：　林順喜　　博士

# 吹牛骰子之人工智慧研究
# On the Study of Artificial Intelligence in Liar Dice

研究生：　黃信翰　　撰

中華民國　　九十八　年　六　　月

# 摘要

## 吹牛骰子之人工智慧研究

黃信翰

吹牛骰子是一種較為特殊的骰子遊戲，屬於不完全資訊賽局的一種。在此遊戲中，骰子扮演著類似撲克牌的角色，玩家必須根據手中的骰子牌型採取喊牌方式，直到有一方決定要做勝負的判定。吹牛骰子最初起源於南美，經過長時間的演化，發展出多人共用一副骰子的 individual hand 類型與玩家各自使用一副骰子的 common hand 兩大類型。

本研究以 common hand 類型為主，因為骰子數變多，且遊戲中僅能獲得少量不可靠的資訊，要找出好的玩法策略顯得更加困難。我們希望能捨棄傳統上常用的賽局樹搜尋與亂數模擬法等耗用大量計算資源的方法，利用賽局理論，以一種簡單明快的作法來達到此遊戲的最佳(或較佳)玩法。並採用貝氏信賴網路，在連續的對局中對網路進行訓練，達成對手行為模擬的效果，藉由發掘對手的弱點來提高勝率。

實驗結果顯示，我們所找到的以隨機猜測為基礎的演算法，對於其他以各種啟發式規則所實作的 32 種測試程式，都有約 6~7 成的勝率，並且與具有一定水準的人類玩家對戰，也有與之抗衡的能力。在加入貝氏網路的學習之後，戰績也有明顯的提升。在不完全資訊遊戲的分析上，先以隨機玩法使自己不易吃虧，再搭配上對手行為模型的輔助，的確為一種可行的作法。

# Abstract


## On the Study of Artificial Intelligence in Liar Dice

By

Hsin-Han Huang

Liar dice is a special dice game. It’s a kind of imperfect information game. In this game, the dice play the roles like the cards in a poker game. Players have to make their calls according to the type of the dice they own until one of them decides to do the judgment. Liar dice was originated from South America and has been evolving for a long time. Eventually, this game evolved two different versions. In "individual hand", there is only a set of dice which is passed from player to player. In "common hand", each player has his own set of dice.

This thesis focuses on the study of the common hand liar dice. Because of the increasing number of dice, we receive too little reliable information when we play. This makes the common hand version much more difficult than the individual hand version. We hope that we can abandon the traditional algorithms such as game tree search and random simulation, which will consume lots of time and space. By applying game theory, we find a simple, fast way to compute the optimal (or suboptimal) solution. Furthermore, we use a Bayesian belief network, and train it by successive playing to build a model of an opponent. The model can help us to find the weakness of the opponent and to win more games.

The experiment results show that the proposed scheme based on random guessing, can achieve about 60 to 70 percent of winning rate against all 32 heuristic-based test programs we designed. And it is competitive when playing with a human player. After we add the Bayesian belief network, the performance of our program also has significant improvement. This shows that when we want to solve an imperfect information game, trying to explore the advantages of random guessing and use an opponent modeling technique is indeed a considerable approach.

目錄

第一章 簡介.....1
第一節 研究背景與動機.....1
第二節 問題敘述.....3
第三節 研究方向及目的.....4
第四節 論文架構.....5
第二章 吹牛骰子.....6
第一節 吹牛骰子簡介.....6
第二節 遊戲進行方式.....7
第三節 複雜度分析.....12
第三章 相關研究探討.....14
第一節 經驗法則.....14
第二節 對手行為模型.....16
第三節 動態規劃.....19
第四章 雙人吹牛骰子探討.....22
第一節 特性分析.....22
第二節 隨機玩法.....27
第三節 貝氏信賴網路.....34
第五章 實驗及結果.....47
第一節 系統研發.....47
第二節 測試結果.....51
第六章 結論與未來發展.....55
附錄 詳細測試結果.....58
參考文獻.....68

附表目錄

表 1.1 四種不同的賽局與相對應的均衡觀念.....2
表 1.2 不同賽局彼此間的差異.....2
表 1.3 各種遊戲的賽局分類.....3
表 2.1 各種遊戲的複雜度比較.....13
表 3.1 Freeman 所歸納出的經驗法則.....16
表 3.2 說謊者側寫.....18
表 4.1 列出所有策略的策略矩陣概念圖.....23
表 4.2 優勢策略舉例.....24
表 4.3 簡化的吹牛骰子策略矩陣.....24
表 4.4 猜拳的策略矩陣.....25
表 4.5 擁有 k 個特定點數骰子的機率.....28
表 4.6 各種可能牌型的種類數.....29
表 4.7 每種個數的組合數與累積數.....32
表 4.8「下雨」節點的機率表.....38
表 4.9「地面溼」節點的機率表.....38
表 5.1 程式對四位人類玩家所取得的勝率.....54

附圖目錄

圖 2.1 遊戲初始情形.....7
圖 2.2 叫牌.....7
圖 2.3「更大叫牌」的意義.....8
圖 2.4 勝負判定.....8
圖 2.5「吹牛」的情況.....9
圖 2.6「1 點」當作王牌.....10
圖 2.7 重新搖骰.....11
圖 3.1 手牌擁有「一對」時各種重搖方式的比較.....16
圖 3.2 Korb 等人對開牌梭哈建立的貝氏信賴網路模型.....19
圖 3.3 費氏數列的展開.....20
圖 3.4 求 fib(n)時所需要呼叫的 fib(1)~fib(5)次數.....21
圖 4.1 隨機玩法示意.....27
圖 4.2 我方考慮是否抓牌的例子.....28
圖 4.3 判斷「是否抓牌」的方法.....30
圖 4.4 判斷「如何喊牌」的方法.....32
圖 4.5 先考慮抓牌再考慮喊牌的例子.....33
圖 4.6 先考慮喊牌再考慮抓牌的例子.....33
圖 4.7 描述心臟疾病的貝氏網路.....35
圖 4.8 描述天氣與地面乾溼關係的貝氏網路.....37
圖 4.9 描述吹牛骰子遊戲的貝氏信賴網路.....39
圖 4.10 描述開叫情形的貝氏網路.....42
圖 4.11 假想牌組與限制陣列的初始情形.....44
圖 4.12 採信對手開叫時的情形.....44
圖 4.13 認為對手支持時的情形.....44
圖 4.14 採信對手叫牌時的情形.....45
圖 4.15 不採信對手開叫時的情形.....45
圖 4.16 不認為對手支持時的情形.....46
圖 4.17 不採信對手叫牌時的情形.....46
圖 5.1 測試程式的決策流程例子.....48
圖 5.2 文字介面測試程式運作畫面.....49
圖 5.3 勝率走向圖說明.....49
圖 5.4 吹牛骰子系統人機介面.....50
圖 5.5 玩家資訊畫面.....51
圖 5.6 對 type1 測試程式之戰績.....52
圖 5.7 對 type5 測試程式之戰績.....52
圖 5.8 對 type23 測試程式之戰績.....52

圖 5.9 套用貝氏網路前後對 type1 測試程式之比較.....53
圖 5.10 套用貝氏網路前後對 type8 測試程式之比較.....53
圖 5.11 套用貝氏網路前後對 type11 測試程式之比較.....54
圖 6.1 以實數進行猜牌.....57

# 第一章 簡介

## 第一節 研究背景與動機

賽局最通用的定義為：兩個或兩個以上的玩家，在理性的前提下，為追求己身目標而造成行為相互衝突而處於的一種對抗狀態。為了設法在這樣的狀態下獲得最後勝利，有了所謂的賽局理論。賽局理論在小至遊戲、大至人類的經濟、社會性活動的研究中都有廣泛的應用。

到 20 世紀中葉，有學者逐步蒐集過去經典的互動模式，成立了以分析人們互動為主旨的新學科：賽局理論(Game Theory)。在當時主要的學者是匈牙利裔的數學天才馮紐曼(John von Neumann)和美國經濟學家摩根斯坦(Oskar Morgenstern)，他們於 1944 年出版的「賽局理論與經濟行為(Theory of Games and Economic Behavior)」則被視為賽局理論的奠基之作[15]。馮紐曼以數學方式證明了在雙人賽局中。只要彼此利益完全對立，就永遠存在一個理性的行動方針。這一証明被稱為「大中取小定理」。它適用於所有一輸一贏的兩人遊戲，馮紐曼證明在這樣的賽局中，總有一種「正確的」玩法，或者更確切的說「最佳的」玩法[12]。

之後賽局理論在各個領域的應用迅速普及開來，並經由一些偉大的數學家一再突破馮紐曼的格局。

1950 年，納許(John Nash)提出「納許均衡(Nash equilibrium)」的概念，他擴展了馮紐曼的理論，證明了非零和的兩人賽局也存在著均衡解，只要對手的策略確定，競爭者就可以有最適反應，而當一組策略互為最適反應時，就稱為「納許均衡」。

賽局架構可依「靜態/動態」和「完全資訊/不完全資訊」兩種分類關係，分為四種不同的賽局，如表 1.1，也因此而有相應的均衡觀念，完全資訊中的靜態賽局均衡觀念是由納許提出，以此為基礎，塞爾登（Reinhard Selten）提出子賽局完美均衡(subgame perfect Nash equilibrium，簡寫為 SPNE)，另外還有不完全資訊的靜態賽局可用貝氏納許均衡(Bayesian Nash equilibrium，簡寫為 BNE)分析，由哈珊伊(John Harsanyi)提出，他們三人在 1994 年共同得到諾貝爾獎[14]。

| | 完全資訊 | 不完全資訊 |
|---|---|---|
| 靜態 | 納許均衡(NE) | 貝式納許均衡(BNE) |
| 動態 | 子賽局完美納許均衡(SPNE) | 完美貝式納許均衡(PBNE)或序列均衡(SE) |

表 1.1 四種不同的賽局與相對應的均衡觀念

表 1.2 歸納了賽局最常見的幾種分類與它們的差異。

| 賽局分類 | 差異 |
|---|---|
| 合作或不合作 | 合作：玩家協議出遵守的規則<br>不合作：玩家各自出招，諜對諜 |
| 靜態與動態 | 靜態：玩家同時出招<br>動態：玩家出招互有先後 |
| 兩人與多人 | 兩人：玩家只有兩位<br>多人：玩家有多位 |
| 零和與非零和 | 零和：玩家的報酬總和為 0<br>非零和：玩家的報酬總和不一定為 0 |
| 完全資訊與不完全資訊 | 完全資訊：賽局三大要素都清楚<br>不完全資訊：三大要素至少有一個不清楚 |

表 1.2 不同賽局彼此間的差異

其中賽局三大要素分別為：參賽者、行動、報酬。

表 1.3 列出了常見的幾種遊戲在賽局中的分類。而我們所研究的對象吹牛骰子，是屬於需考慮機率的不完全資訊賽局。

| | 完全資訊賽局 | 不完全資訊賽局 |
|---|---|---|
| 沒有機率問題 | 西洋棋(Chess)<br>圍棋(Go)<br>象棋(Chinese Chess) | 走私者(Inspection game)<br>戰艦遊戲(Battleship) |
| 需考慮機率 | 西洋雙陸棋(Backgammon)<br>大富翁(Monopoly) | OPEC 遊戲(OPEC game)<br>撲克(Poker)<br>吹牛骰子(Liar dice) |

表 1.3 各種遊戲的賽局分類

骰子是歷史悠久且使用相當廣泛的博奕道具，使用骰子來進行的遊戲大部分為不完全資訊遊戲，如 Pig、Yahtzee、Craps，而吹牛骰子算是其中規則較為複雜的一種。吹牛骰子起源自中南美，經由海路傳至世界各地，各自演變出許多類型，也由於規則上的歧異，造成現在僅有零星的研究成果。

## 第二節 問題敘述

吹牛骰子屬於不完全資訊賽局，與一般的骰子遊戲不同，規則較複雜且多變。主要分為 individual hand(多人共用一副骰子)與 common hand(玩家各自擁有一副骰子)兩種類型。Individual hand 類型在過去已有一些研究成果，使用的方法有近似模擬法、經驗法則、對手行為模型、動態規劃等。

而 common hand 類型的吹牛骰子目前尚無研究成果。由於骰子數目變多，遊戲複雜度提升，但又因為骰子彼此間的獨立性，以及由其他玩家行動所得到的資

訊並不可靠，造成在遊戲過程中所獲得的資訊極端的少，於是在不完全資訊賽局研究中常使用的方法，如賽局樹搜尋或近似模擬法，若套用在吹牛骰子上，不僅效果不佳，更會耗費大量時間。

於是在吹牛骰子上，如何找到一個能夠在短時間內明快的做出決策，且具有一定實力的演算法，就成了一項值得研究的課題。

## 第三節 研究方向及目的

本研究將以較簡易的兩人 common hand 吹牛骰子為主體，分析其遊戲結構。由於骰子擲出的點數是隨機且獨立的，無法由自己手中的骰子去推斷別人的手中的骰子，並且遊戲過程中所能獲得的唯一資訊來自對手的喊牌，由於考慮到有欺騙的可能性存在，也不該將其視為可靠資訊，於是此遊戲自始至終中僅有的可靠資訊都來自自己的手牌。

考慮到此遊戲的不完全資訊性，並不存在使用單一策略的優勢策略均衡，然而在無從得知對手行為背後所代表的意義的情形下，我們也不能確定該如何正確分配使用混合策略的比例。但可以試著找到「不吃虧」的玩法。

我們由一小部份的策略矩陣觀察遊戲的特性，並試著推廣到整個策略矩陣，發現或許存在一個以隨機選取為基礎的簡單玩法，能夠逼近屬於這個遊戲的真正隨機玩法，如此便能確保無論對手採取何種策略，皆保證至少能有大致一半的勝率。

又因為此遊戲的一大困難點在於對手的性格不明朗，且其行動背後的意義不明導致無法找出正確混合策略，除了消極的找出絕對不吃虧的方法外，也可以考慮對手行為模型，我們根據玩家在遊戲中可能遭遇的各種影響決策的參數，建立

起貝氏信賴網路，在每次對局中收集對手的資訊對網路進行訓練，使我們的程式能夠具有「洞察力」，以提高勝率。

## 第四節 論文架構

本論文共分為六章，首先簡單介紹吹牛骰子遊戲，並對研究動機及方向做大致的概述。第二章將對吹牛骰子的規則進行詳細的描述，並提出此研究主要的問題點。第三章介紹過去對於吹牛骰子的相關研究成果，以及對於處理其他類似賽局時常用的手法。第四章則對於雙人吹牛骰子賽局提出一個明快、不需耗用大量計算資源的決策方式並詳細說明其背後的理由。第五章會對我們提出的方法分別對電腦與人類進行測試並分析其結果。最後在第六章對本研究做總結，並說明未來可能的研究方向。

# 第二章 吹牛骰子

## 第一節 吹牛骰子簡介

骰子或許是人類史上最早出現的博弈道具，考古學家發現，早在西元前兩千多年，古埃及人便會以獸骨雕刻而成的六邊形方塊作為打發時間的娛樂。

時至今日，骰子仍廣泛的在各種博弈遊戲中被使用。有被作為主要道具使用的如 Craps、Pig、Yahtzee，也有被當作輔助道具的如麻將、牌九。

吹牛骰子(liar dice)，最初起源於南美，一開始只是一個在酒吧裡用來打發時間，或者用來賭酒的小遊戲。在當時海盜橫行，酒吧又是海盜們上陸時最常聚集的地方，也就把這個遊戲帶上船，並帶到世界各地，所以也有海盜骰子(pirate dice)的別稱。

後來在各地經過各自的演變，衍生出不同的規則及稱呼，如 Dudo、Cachito、Maxical。不同的稱呼在細部規則上各有不同，遊玩的場所也不再侷限於酒吧，於是衍生出比一開始更複雜的類型，成為一種正式的博弈遊戲。之中最大的改變在於，由最初的所有玩家共用一副骰子(individual hand )，後來演變成各個玩家各自使用一副骰子(common hand)。

此遊戲幾年前因為博奕節目「小氣大財神」而在台灣大為風行。

吹牛骰子在骰子遊戲中屬於規則較複雜、變化性較高的一種，並不是如一般的在擲出骰子後，便馬上以擲出的點數作為判斷勝負的依據，而是每個玩家各自擁有一副骰子，扮演著類似在梭哈遊戲中撲克牌的角色，玩家必須根據自己的手牌，做出一連串「叫牌」及「抓牌」的動作。

## 第二節 遊戲進行方式

此遊戲參與的人數無特別限制，通常為 2~5 人，以下以 2 人遊戲為例。

雙方各搖 5 顆骰子並以骰盅蓋住不讓別人看見，只有自己可以隨時查看，如圖 2.1。除第一回合的先手玩家必須「叫牌」之外，每一回合的玩家都可以選擇「叫牌」或「抓牌」。


圖 2.1 遊戲初始情形

先手進行叫牌「x 個 y」，代表宣告「兩位玩家的骰子總計，點數為 y 的至少有 x 個」，如圖 2.2。


圖 2.2 叫牌

因為叫牌的玩家只看得到自己的骰子，所宣告的叫牌未必成立，於是輪到的下家必須決定是否相信上家的叫牌。若相信，則要喊一個更大的叫牌，如圖 2.3 所示。「更大」的意義有如下兩種：

- 個數更大、點數任意，或
- 個數相同、點數更大

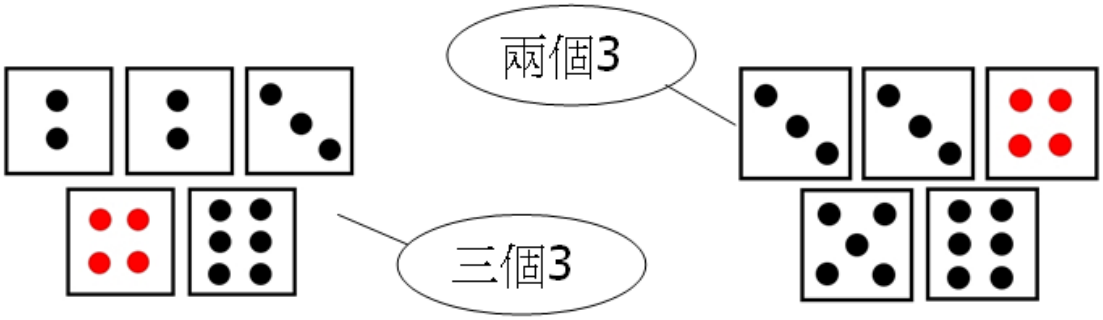



圖 2.3「更大叫牌」的意義

若輪到的玩家認為上家的喊牌不會成立，則「抓牌」，雙方揭開骰盅看叫牌是否成立，若不成立則抓牌的一方勝，若成立，則抓牌方輸。以圖 2.4 為例，雙方共有三個 3，但右邊的玩家叫牌「四個 3」，未能成立，故抓牌的左邊玩家獲勝。

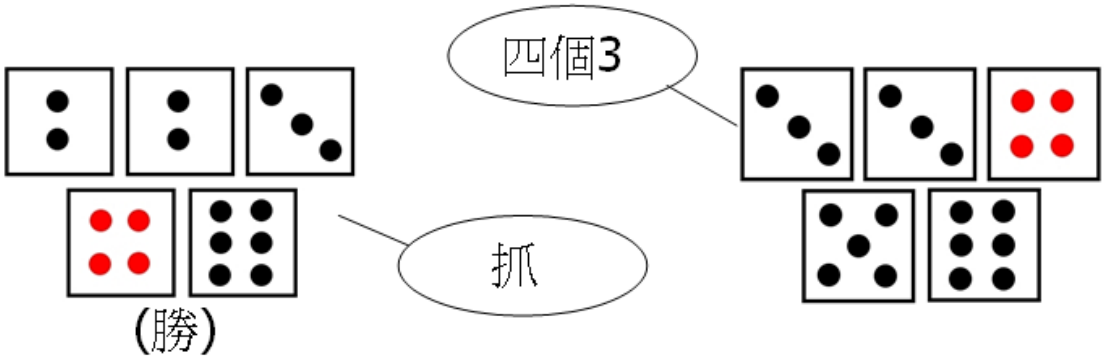



圖 2.4 勝負判定

就如名稱「吹牛」，因為在遊戲結束前無從得知對手的底牌，對手的叫牌過程是唯一的資訊，於是在叫牌中吹牛、提供錯誤的資訊來誤導對手也是合理的策略。

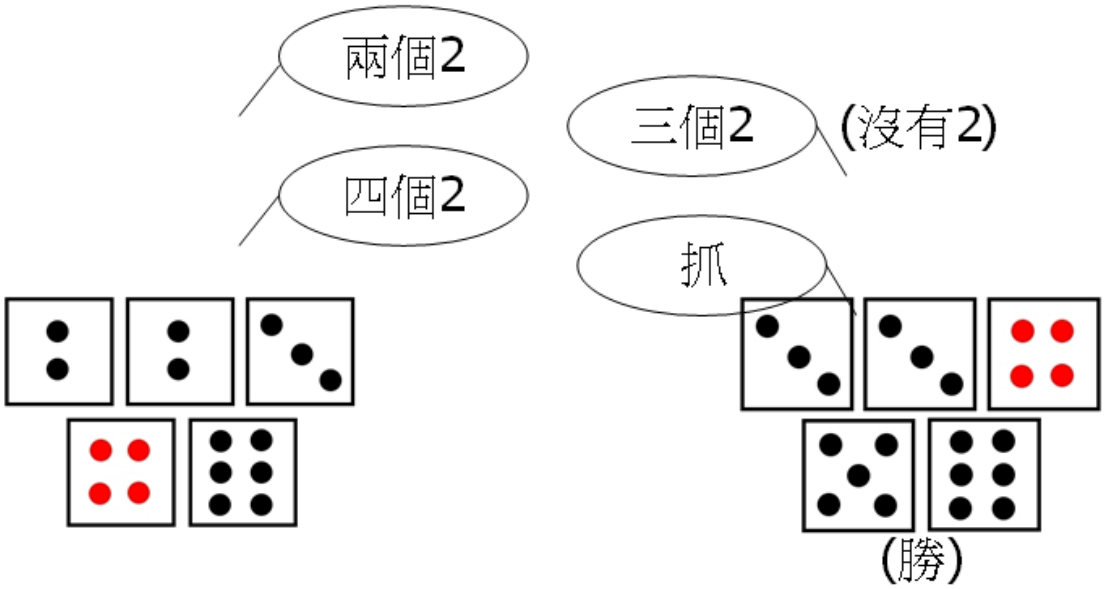



圖 2.5「吹牛」的情況

如圖 2.5，與之前的例子牌型相同而發展不同，右邊玩家在手牌沒有 2 的情況下叫出「三個 2」，即是一種欺騙的行為。若先手玩家採信此叫牌，誤以為對方在「2」上有所支援而喊出「四個 2」的話，便落入對方的陷阱。

即使是相同的牌型，也會因為玩家的心態、個性不同，影響到中途的發展，導致不同的結果。

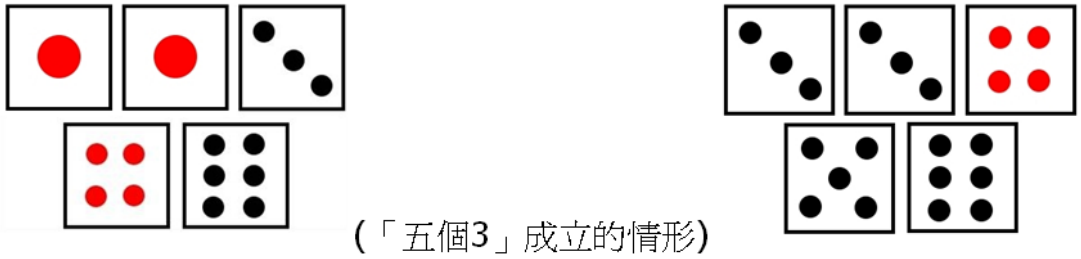
## 延伸規則

依照上述規則進行遊戲，通常一場遊戲會在 4~5 回合結束，為增加遊戲的趣味性，發展出了一些延伸的規則，簡述如下。

1.「1 點」可當作王牌：

雖然在兩名玩家的情況下，喊出的骰子個數上限是 10 個，但其實真的喊到這麼高的機會並不多，通常會結束在 5~6 顆骰子的時候，於是有了將 1 點當作王牌的規則。也就是在計算骰子個數時，1 點可以被當作任何點數計算，如圖 2.6，

如果遊戲結束時的叫牌是「五個 3」，單純計算 3 點的個數的話僅有三個，但左方玩家有兩個 1 點，也可以當作 3 點計算，於是在這個局面「五個 3」是成立的。



(「五個3」成立的情形)

圖 2.6「1 點」當作王牌

有了這項規則，玩家在考慮策略時，除了注意目前喊到的點數的個數，也必須注意到 1 點的個數，1 點的骰子有很大的運用空間，可能成為決定勝負的關鍵。

要注意的是這項規則僅在「遊戲過程中沒有玩家喊過 1 點」時才有效，若過程中有玩家喊了「1 點」，則遊戲結束後便只能當作 1 點計算。

2. 允許重新搖骰：

遊戲中常會碰到叫牌難以叫上去的情況，採用此規則，玩家可以有重新搖骰的機會，首先由骰盅取出任意個數的骰子，通常是對玩家有利而想要保留的，這些取出的骰子就成為所有玩家的公共資訊。之後玩家便可以搖骰盅內的剩餘骰子，再決定叫牌。

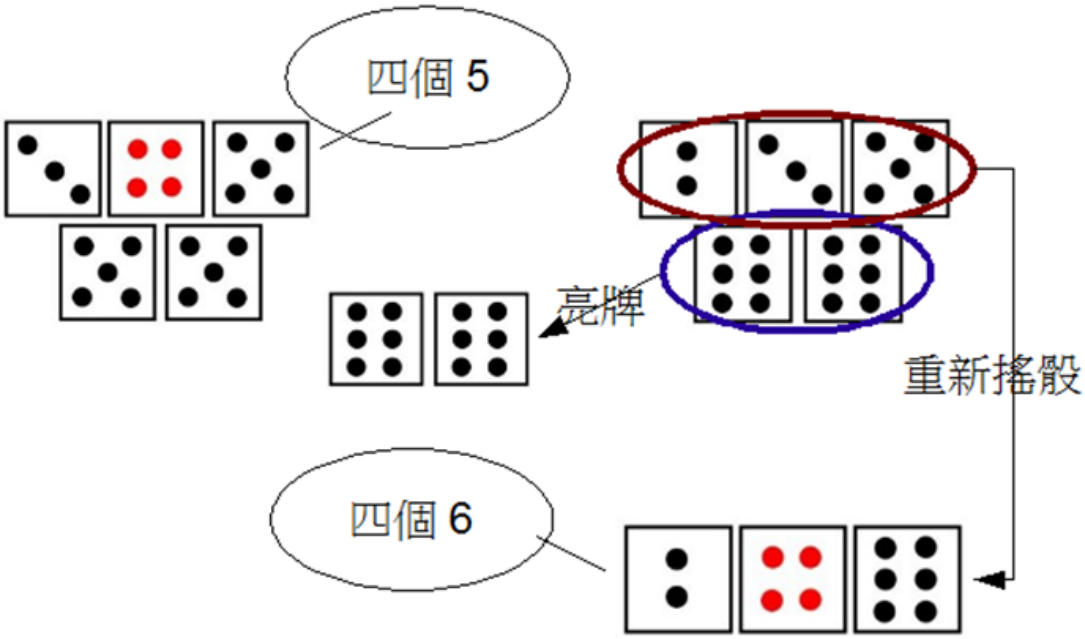



圖 2.7 重新搖骰

如圖 2.7，先手叫牌「四個 5」，後手玩家可能會考慮抓牌，或者喊手上有最多的「6」，但個數只有兩個。這時可以考慮重新搖骰，亮出手中的兩個「6」，重搖剩下的三顆骰子，但不需亮出這三顆骰子，若能再多搖到一顆「6」，就能更有把握喊出「四個 6」。

然而這條規則也有所限制，若是重新搖骰後就必須喊牌不能反悔，並且骰子亮出後，就不能再拿回，將以原本亮出的骰子放在牌桌上直到最後。

3. 懲罰機制：

落敗的玩家將被取走一顆骰子，在骰子數目較少的不利條件之下進行下一場遊戲。直到有一方玩家失去全部的骰子，遊戲才真正告終。

而在本研究中，我們僅以基本規則的雙人 common hand 吹牛骰子為研究對象，上述三種延伸規則並不納入考慮。

## 第三節 複雜度分析

Common hand 類型吹牛骰子，每位玩家各擁有五顆骰子，其他玩家的骰子在遊戲中是始終隱藏的資訊，所能得知的只有自己手上的牌，不確定因素相當多，判定時的分歧度也隨之增加。

遊戲玩家通常為 2~5 人，以下我們以 5 人賽局分析其複雜度，由於每個玩家的骰子各自獨立互不相干，所以可能性皆為 $6^5$ 種，5 位玩家的牌型共計 $\left(6^5\right)^5 = 6^{25}$。玩家的行動上，在 5 人賽局中，從「兩個 1」起直到「二十五個 6」共 144 種都是合法叫牌，但由於叫牌有大小限制，加上有些牌型實在太難出現，會被玩家考慮到的叫牌方式每次約為 30 種。而五位玩家在每場遊戲大約有兩次叫牌的機會，在行動上的複雜度約為 $\left(30^5\right)^2 = 30^{10}$。

於是總複雜度約為 $6^{25} \times 30^{10} \cong 1.67 \times 10^{34}$。

吹牛骰子的遊戲歷程與規則相較於其他遊戲都簡單的多，但卻由於過多的不確定性，使得賽局樹的分支度相當大，使搜尋空間提高許多。表 2.1 比較了各種遊戲的複雜度。

| 遊戲 | 類型 | 搜尋空間 | 目前程式解決狀況 |
|---|---|---|---|
| 五子棋 | 完全資訊無機率 | ~$10^{28}$ | 完全解決 |
| 西洋雙陸棋 | 完全資訊有機率 | ~$10^{20}$ | 完全解決 |
| 象棋 | 完全資訊無機率 | ~$10^{48}$ | 職業八段 |
| 西洋棋 | 完全資訊無機率 | ~$10^{50}$ | 幾乎可打敗所有人類 |
| 橋牌 | 不完全資訊有機率 | ~$2.3x10^{24}$ | 業餘水準 |
| 撲克 | 不完全資訊有機率 | ~$5x10^{40}$ | 業餘水準 |
| 圍棋 | 完全資訊無機率 | ~$10^{170}$ | 業餘初段 |
| 吹牛骰子 | 不完全資訊有機率 | ~$1.67x10^{34}$ | Common hand 類型尚無研究成果 |

表 2.1 各種遊戲的複雜度比較

# 第三章 相關研究探討

目前所能找到的關於吹牛骰子的文獻，僅有 individual hand 類型有少數研究，而 common hand 類型目前尚無研究資料。兩種類型的主要差異在於 individual hand 僅使用一組骰子，在各個玩家之間傳遞。

Individual hand 類型之規則簡述如下：第一個玩家搖五顆骰子，以骰盅蓋起不讓其他人看見。首先叫牌，但所叫牌組不一定成立。之後將蓋著的整組骰子傳給第二位玩家。第二位玩家選擇相信或抓牌，若相信，則查看骰子之後，可以選擇取出數顆骰子，並重搖剩下的，接著他必須做出比上一位玩家更大的叫牌，再把整組骰子傳給第三個玩家，如此持續下去直到有玩家決定抓牌。[1]

反之，在 common hand 類型的吹牛骰子中，玩家能夠掌握的資訊十分有限，機率性太高。不僅對手所擁有的手牌是直到遊戲結束前都完全無法得知，再考慮對手的叫牌可能含有欺騙的成分，可以說在整場遊戲中，玩家自己的手牌是僅有的可靠資訊。

在過去對於骰子類遊戲的研究中，常被使用到的方法大致可分為經驗法則、對手行為模型、動態規劃，以下就針對各個方法進行探討。

## 第一節 經驗法則

當賽局樹過於龐大難以完全展開，或是遊戲本身含有不完全的資訊時，經驗法則便可以派上用場。經驗法則也常常是一個遊戲的初學者最先學習的方法。

經驗法則指的是在玩家經歷了足夠多的遊戲場數後，依照經驗所歸納出的一

些簡單、易使用的建議規則，其優點是能從當下的遊戲局面，快速有效率的得到一個不錯的解，但問題在於解的品質好壞相當依賴玩家的經驗判斷正確與否。經驗法則是帶有主觀成見、偏好的，越是複雜的賽局就越不能保證經驗法則的正確性。在賽局樹過於龐大、或機率性太高難以下判斷時，經驗法則成為實用的變通方式，常常與不同的方法結合做為輔助。

以吹牛骰子而言，考慮到的經驗法則可以有如下幾項：

a. 對手是不是在說謊。

b. 對手的底牌的判讀。

c. 要不要抓牌。

d. 考慮欺騙。

e. 要喊什麼牌。

1989 年，Freeman[1]提出了最早的對於 individual hand 類型的吹牛骰子的研究。吹牛骰子的決策，大致可以分為兩個部分：「是否要抓牌」、與「要喊什麼牌」。Freemen 認為抓牌與否靠的是玩家的經驗，將討論範圍只限定在後者，也就是已經從上一家接收並查看過骰子後需要決定接下來的喊牌時的情形。分別討論查看骰子得知上家是說實話或謊話、以及實際上骰盅裡的牌型的各種可能性來討論最佳解。以近似模擬的方式，針對每個可能的局面進行大量的實驗，來得到該局面的最好應對方式。圖 3.1 為實驗結果之一，代表當接收到的骰子手牌中有「一對」時，重搖不同個數後能得到的效益。「×」代表重搖一顆，「o」代表重搖兩顆，「＋」代表重搖三顆，「□」則代表重搖四顆，而在此圖中，不論手中擁有的一對牌為何，「＋」都位於最高處，代表這種情況下，重搖三顆是最好的作法。

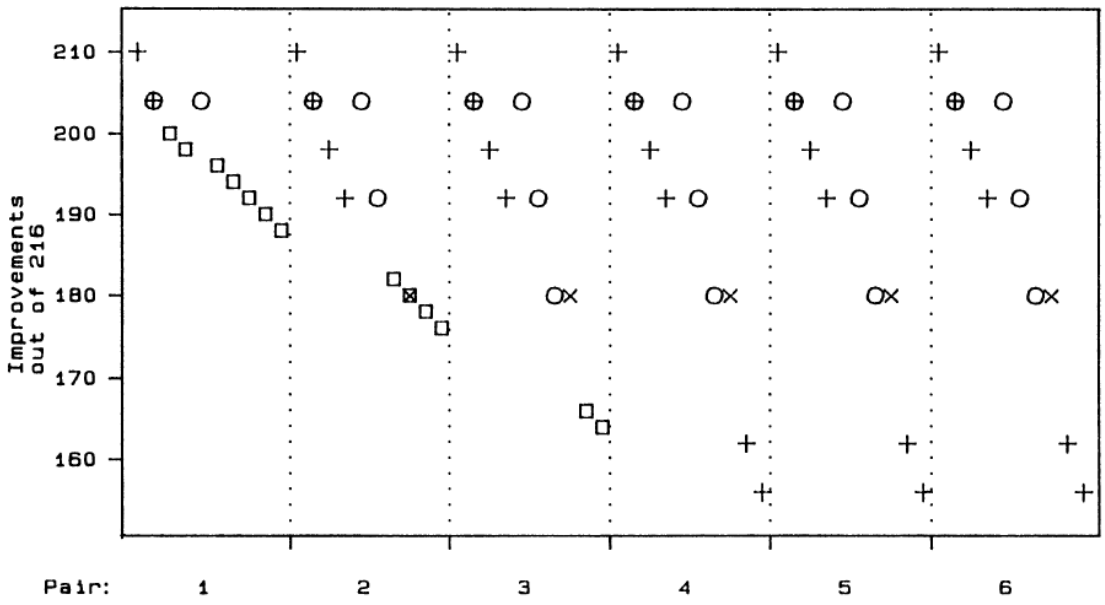



圖 3.1 手牌擁有「一對」時各種重搖方式的比較

Freeman 歸納出了一套經驗法則，當作其他玩家進行遊戲時的建議玩法。

大致內容如表 3.1：

*Some good choices when accepting the truth*

| *Given* | *Hold* | *Throw* | *Chance of improvement* |
|---|---|---|---|
| One pair | Two (pair) | Three | At least 13 to 5 on |
| Two pairs | Higher pair | Three | At least evens except with pairs of 5s and 6s |
| Three of a kind | Three (of a kind) | Two | Varies from 17 to 1 on to 5 to 4 against |
| Low straight | Four (2, 3, 4, 5) | One | 5 to 1 against |
| High straight | None or any one | Five or four | About 17 to 1 against |
| Full house | Three (of a kind) | Two | Varies from 5 to 4 against to worse than 2 to 1 against |
| Four of a kind | Four (of a kind) | One | Varies from 5 to 1 on to 5 to 1 against |
| Five of a kind | Lie! (this should never have been accepted) | | |

表 3.1 Freeman 所歸納出的經驗法則

## 第二節 對手行為模型

一般在遊戲中玩家最常使用來決定如何採取下一步行動的方法有二:使用搜尋技巧(如 minimax 或 alpha-beta pruning)、或者因為賽局樹太深沒辦法完全

搜尋而使用經驗法則作為評估。但真的能靠經驗法則而完全精確的評估局面而找出最好一步的遊戲並不多，尤其若自身的經驗法則評估錯誤，可能會導致失敗收場。

由於玩家進行決策時，往往會參入自身的情感、直覺、偏見，若我們能根據該對手之前遊戲的行動所展現出的個性，而非透過搜尋或自身經驗法則，來推敲對手行動的意義，或許能更準確的預測對手的行動，這就是對手行為模型的用意。

行為模型的主要好處有二[8]：

1. 對手行為的意義，若我們對於對手行為背後所代表的意義能確保一定正確率的掌握，便能對遊戲中的不完全資訊部分進行有根據的推測。當接下來需要進行各種狀況的考慮，或搜尋賽局樹時，對於可能性較高的分支情形可以給予較高的優先權，進行較深度的搜尋，或者是將可能性較低的分支排除，減少複雜度。
2. 對手行為的預測，我們對於可以採取的所有行動，能推測對手可能的回應，便能協助判斷每個走步的價值作出更正確的選擇。

透過建立對手行為模型，我們能使程式具有學習能力，能夠經由重複對局，發現當前對手的行為模式及弱點，進而提高勝率。

2003 年，Sum 和 Chan[2]，延續 Freeman 的成果，並補強其不足的地方，導入機率計算與簡單的建立對手模型(opponent modeling)技術，藉由重複對局來建立對手的說謊者側寫(liar profile)，如表 3.2，再結合經驗法則，發展了一套風險迴避(risk-averse)演算法，整合對於目前局面各項參數的推測，來進行決策。

| $P_R$ (Range) | Calls | Lies | $P_{LC}$ |
|---|---|---|---|
| 0.05 (0 - 0.1) | 18 | 3 | 0.17 |
| 0.15 (0.1 - 0.2) | 13 | 2 | 0.15 |
| 0.25 (0.2 - 0.3) | 27 | 12 | 0.44 |
| 0.35 (0.3 - 0.4) | 5 | 2 | 0.4 |
| 0.45 (0.4 - 0.5) | 4 | 2 | 0.5 |
| 0.55 (0.5 - 0.6) | 0 | 0 | - |
| 0.65 (0.6 - 0.7) | 0 | 0 | - |
| 0.75 (0.7 - 0.8) | 1 | 1 | 1 |
| 0.85 (0.8 - 0.9) | 0 | 0 | - |
| 0.95 (0.9 - 1) | 0 | 0 | - |

表 3.2 說謊者側寫

Korb 等人[5]，使用貝氏信賴網路(Bayesian belief network)進行對撲克牌梭哈的研究，他們對梭哈的規則進行簡化，設想了一種名為開牌梭哈(Heads-up 5-card stud poker)的版本，與一般梭哈的不同點在每位玩家手牌僅有一張是蓋住的，於是遊戲的目的就成了猜測對手的那張底牌為何。對這樣的簡易版本以貝氏網路建構起完整的賽局結構，如圖 3.2 所示，推定出對手模型，對照敵手目前的策略，協助己方判定下注策略。

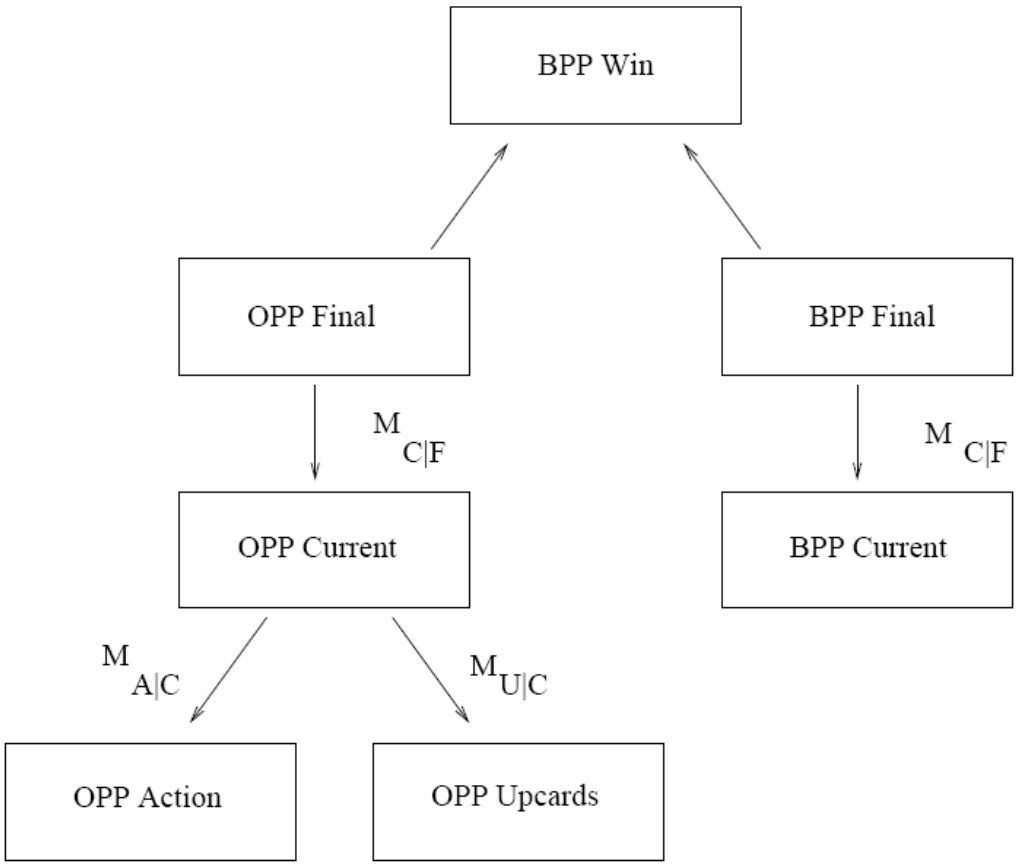



圖 3.2 Korb 等人對開牌梭哈建立的貝氏信賴網路模型

Egnor[4]所實作的猜拳程式「Iocaine Powder」，使用頻率分析與紀錄比對方式，結合連環猜測，推定出對手的下一步可能行動。此程式在 2000 年 ICGA 第一屆電腦猜拳大賽得到冠軍，之後並公布程式碼，成為之後大部份猜拳程式的基礎。

## 第三節 動態規劃

動態規劃(dynamic programming)與分而治之(divide-and-conquer)方法類似，藉由將子問題的答案合併來解決問題。分而治之演算法是將問題分割成獨立

的子問題，遞迴的解決這些子問題，再將它們的解答合併成為問題的答案。而動態規劃則使用在這些子問題並非彼此獨立的時候，也就是子問題之間共享子問題時，若使用分而治之法將會做上比實際所需的更多的工作，重複的解決共同子問題。動態規劃只解決每個子問題一次，並將答案儲存在一個表格內，來避免每次遇到相同子問題時都需要重複計算。

動態規劃通常應用在最佳化問題(optimization problem)上。在這類問題中有許多可能的解答。每個解答有一個數值，我們希望找出一個最佳(最小或最大)數值的解答[9]。

下面以費氏數列為例說明動態規劃與分而治之法的差別，費氏數列第一項 $fib_1$ 及第二項 $fib_2$ 為 1，其後每一項皆為前兩項的和。故數列如下：1、1、2、3、5、8、13、21、……。

$$\mathrm{fib}_n = \begin{cases} 1 & n = 1, 2 \\ \mathrm{fib}_{n-1} + \mathrm{fib}_{n-2} & n > 2 \end{cases}$$

以下以 fib(n)記為費氏數列的第 n 項，當使用分而治之法計算 fib(5)時，可以如圖 3.3 的遞迴呼叫計算。

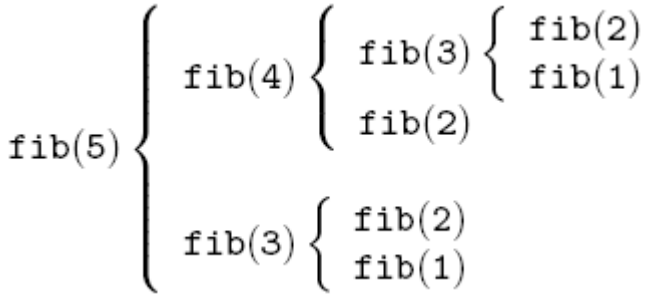



圖 3.3 費氏數列的展開

在此可以發現 fib(3)、fib(2)、fib(1)都被重複做了計算。當 n 變得越來越大，被重複的多餘計算也越來越多，圖 3.4 列出隨著 n 的增長，fib(1)~fib(5)需要的計算次數。使用分而治之法計算費氏數列時，效率十分差。

| | Recursive Calls for `fib(n)` | | | | |
|---|---|---|---|---|---|
| n | `fib(1)` | `fib(2)` | `fib(3)` | `fib(4)` | `fib(5)` |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 |
| 3 | 1 | 1 | 1 | 0 | 0 |
| 4 | 1 | 2 | 1 | 1 | 0 |
| 5 | 2 | 3 | 2 | 1 | 1 |
| 6 | 3 | 5 | 3 | 2 | 1 |
| 7 | 5 | 8 | 5 | 3 | 2 |
| 8 | 8 | 13 | 8 | 5 | 3 |
| 9 | 13 | 21 | 13 | 8 | 5 |
| 10 | 21 | 34 | 21 | 13 | 8 |
| ... | | | | | |
| 20 | 2584 | 4181 | 2584 | 1597 | 987 |
| ... | | | | | |
| 30 | 317811 | 514229 | 317811 | 196418 | 121393 |
| ... | | | | | |
| 40 | 39088169 | 63245986 | 39088169 | 24157817 | 14930352 |

圖 3.4 求 fib(n)時所需要呼叫的 fib(1)~fib(5)次數

使用動態規劃方式時只要建立一個表格，將計算過的 fib(n)存入表格中，其後需要時即可直接取用，省去了重複計算的資源與時間。

2007 年，Johnson[3]結合動態規劃與賽局理論進行僅使用一顆骰子的簡化版吹牛骰子的研究，並對數種方法進行實作及評估。

除了吹牛骰子，動態規劃也常被運用在其他骰子遊戲上。2003 年，Woodward[6]使用動態規劃方式得到了骰子遊戲「Yahtzee」的最佳解。2005 年 Neller 等人[7]將動態規劃應用在另一種骰子遊戲「Pig」上，對規則進行些微修改後得到了其簡化版本的最佳解。這兩種遊戲有些許雷同，皆為由玩家擲數次骰子後根據骰子的組合累積分數，直到有一方達到目標分數即獲得勝利。

# 第四章 雙人吹牛骰子探討

經過長時間的演化，吹牛骰子發展出多人共用一副骰子的 individual hand 類型與玩家各自使用一副骰子的 common hand 兩大類型。本研究以 common hand 為主題，而此類型也有各種衍生的規則與雙人或多人的差別。為方便起見，以下以基本規則的雙人 common hand 開始著手。

吹牛骰子是具有機率性的博弈遊戲，因為玩家手上僅有少量的資訊，無法確定是否能獲得勝利。因此要玩得好並不容易，具有一定程度的挑戰性。

## 第一節 特性分析

從這個遊戲的形式跟規則看來，很容易聯想到撲克牌遊戲，例如梭哈，五張撲克牌可對應到五顆骰子，而每回合的叫牌可對應到梭哈中的下注行為。

但吹牛骰子遊戲有以下幾種特性與梭哈有所差異：

- 遊戲中每位玩家各持有一副骰子，且骰子相互獨立，因此沒辦法由自己的牌型去推斷其他玩家的牌型。撲克牌中，每張牌皆為獨一無二的，且在每個回合皆會有幾張牌被攤開，也就得到額外資訊有助於判斷情勢。
- 就算能夠得知其他玩家的牌，也沒辦法就此判定勝負，無論是什麼樣的牌型，每個玩家都有贏的可能，事實上，在遊戲過程中，每回合的勝利條件都不同。在撲克牌中，每種牌組彼此之間都可判定強弱，且這種強弱關係是始終固定的。
- 任何玩家都有立刻使遊戲結束的權利，而這麼做的玩家也有機會獲勝。一般而言，在撲克遊戲中，雖然玩家有「蓋牌」此一結束遊戲的手段，

但這麼做的玩家將被判定為輸。

首先我們先以賽局理論的角度觀察這個遊戲的性質。

我們將吹牛骰子的所有策略及策略間的勝負關係製成一賽局損益表，大致會是如表 4.1 的形式。

| | 乙 | | | | | | |
|---|---|---|---|---|---|---|---|
| 甲 | | 策略 1 | 策略 2 | 策略 3 | 策略 4 | …… | 策略 k |
| | 策略 1 | 平 | 乙 | 甲 | 甲 | | 乙 |
| | 策略 2 | 甲 | 平 | 乙 | 甲 | | 乙 |
| | 策略 3 | 乙 | 甲 | 平 | 乙 | | 甲 |
| | 策略 4 | 乙 | 乙 | 甲 | 平 | | 乙 |
| | …… | | | | | | |
| | 策略 k | 甲 | 甲 | 乙 | 甲 | | 平 |

表 4.1 列出所有策略的策略矩陣概念圖

其實不只是吹牛骰子，所有的賽局都可以如此表示。

要注意的是，因為此遊戲中存在「對方的骰子手牌」此一始終被隱藏的資訊，任何策略都不能保證能夠每場都獲勝，在表中的勝利方，指的是雙方長期玩下來，大致能有較高勝率的一方。

## 優勢策略

優勢策略的意義為，當我方選擇此策略時，無論對方採取何種行動，我方都能獲得比選擇其他策略時更大的利益。

考慮如表 4.2 的策略矩陣，括弧內數對的意義為當雙方採用此種策略組合

時，所各能獲得的利益。

| | 乙 | | |
|---|---|---|---|
| 甲 | | 策略 1 | 策略 2 |
| | 策略 1 | (2,2) | (4,1) |
| | 策略 2 | (1,4) | (1,1) |

表 4.2 優勢策略舉例

首先觀察甲的狀況，當甲方採取策略 1，而乙方分別採取策略 1 及策略 2 時，甲方的獲益分別為 2、4，而當甲方採取策略 2，乙方分別採取策略 1 及策略 2 時，甲方的獲益分別為 1、1，於是無論乙方採取哪一個策略，甲方採用策略 1 的獲益始終比策略 2 大(2>1、4>1)，所以策略 1 為甲方的優勢策略。

以相同的推導過程，也可以得到乙方的優勢策略為策略 1。當所有玩家都找得到優勢策略時，則這些策略的組合稱為「優勢策略均衡」。

接下來設法將此優勢策略均衡的概念套用在如表 4.1 的損益表上。

吹牛骰子的實際策略數目相當龐大，要歸納出所有策略並建構出損益表是相當耗費時間或幾乎不可能的，為方便說明我們取出策略矩陣的一部分，並假設雙方玩家都僅能採用這一部分的策略，以表 4.3 舉例。

| | 乙 | | |
|---|---|---|---|
| 甲 | | 總認為對方說實話 | 總認為對方說謊話 |
| | 總說實話 | 乙 | 甲 |
| | 總說謊話 | 甲 | 乙 |

表 4.3 簡化的吹牛骰子策略矩陣

若甲方假設乙方是「總認為對方說實話」，那麼他應該要讓自己「總說謊話」，但這麼一來乙方也可以「總認為對方說謊」，於是甲又要改為「總說實話」。

也就是遊戲雙方為了找到優勢策略，他們所選擇的策略的組合會不斷的變動，這是由於每種策略都與其它的策略互有輸贏，這樣的性質稱為「非遞移性」。

遞移性為一種代數性質：若 a>b 且 b>c，則必 a>c，即 a、b、c 中能找出一個最大，其大小關係是絕對的。

而非遞移性則相反，僅能比較彼此間的相對大小，若將所有成員的大小排序，則會得到一個或一個以上的迴圈：若 a>b 且 b>c，則可能 c>a。

吹牛骰子也是非遞移性的遊戲：「總是說謊的玩家」會輸給「總是認為對手說謊的玩家」。「總是認為對手說謊的玩家」會輸給「總是說實話的玩家」。....，依此類推。

猜拳是最簡單的非遞移性遊戲，剪刀贏布、布贏石頭、石頭贏剪刀，如表 4.4。

| | | 乙 | | |
|---|---|---|---|---|
| 甲 | | 剪刀 | 石頭 | 布 |
| | 剪刀 | (0, 0) | (-1, 1) | (1, -1) |
| | 石頭 | (1, -1) | (0, 0) | (-1, 1) |
| | 布 | (-1, 1) | (1, -1) | (0, 0) |

表 4.4 猜拳的策略矩陣

由於策略的非遞移性，並不存在使用單純策略的優勢策略。接下來考慮使用混合策略的情形。

單純策略代表從頭到尾都採取單一策略，而混合策略的意義為按照某種比例的分配去採取各種策略，以猜拳賽局為例，「1/2 的機會出剪刀、1/4 的機會出石頭，1/4 的機會出布」即為一種混合策略。

而假設若我們能夠知道對手喜歡出石頭多一點，那麼我們便可以調整機率，降低出剪刀的機率及提高出布的機率。

對吹牛骰子賽局而言也是如此，以上述的策略矩陣而言，當我們在考慮不同策略的混合比例時，若能事先知道對手的行為偏好，例如喜歡說謊，那麼我們便能採取相對應的措施，調高認為對手說謊的機率來提高獲勝的機會。然而達到這個目的的困難點在於沒有辦法事先知道對手的偏好。雖然常有以經驗法則來假設對手行動意義的作法，但許多時候各種選擇的孰優孰劣難以釐清，若經驗法則的假設錯誤，可能造成反效果。

找不到單純優勢策略，而使用不同比例的混合策略仍有失敗的風險，也就是說並沒有任何有絕對把握能贏的策略。但我們能設法絕對不吃虧。

以同樣的策略矩陣為例，由於不知道對手是以何種比例決定策略，於是我方也沒有任何決定比例的根據，但當我們單純以各 50%的均等機率採取兩種策略，則無論對手採取任何比例的分配，我方都能有至少 50%的勝利機率。

將上述的概念套用在整個策略矩陣中的話，同樣的也由於非遞移性沒辦法找到優勢策略，以及沒有對手偏好的資訊而無法知道混合策略比例。那麼會得到與之前考慮簡化策略矩陣時相同的結果，也就是採用隨機策略，可以確保不吃虧。如圖 4.1 所示。

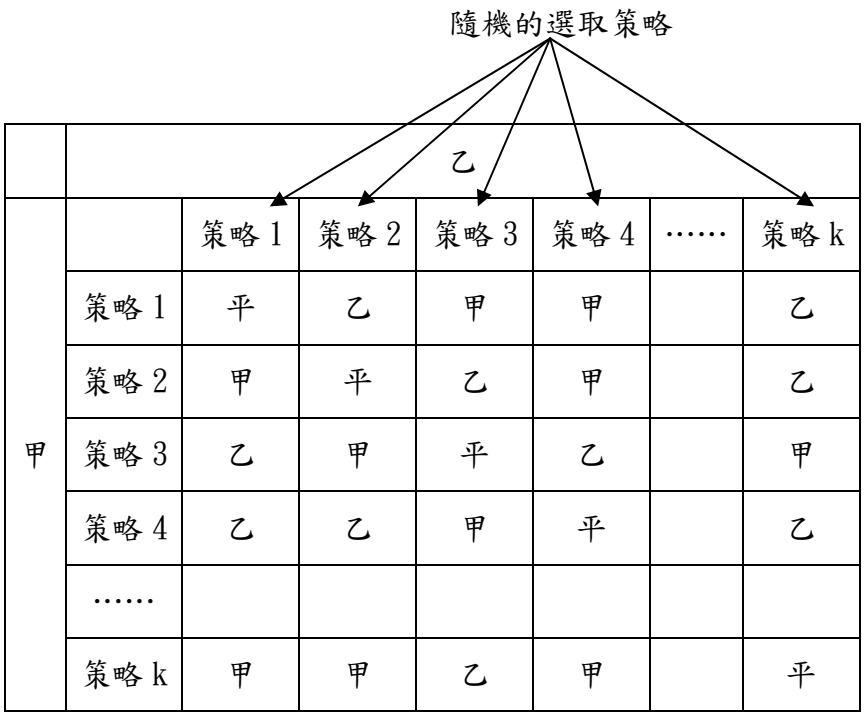



圖 4.1 隨機玩法示意

若我們能將完整的策略矩陣全部畫出來，就可以達到這個目標，但跟簡化版的矩陣不同的是，完整的策略矩陣非常的大，我們不可能將它全部畫完，於是只好退而求其次，希望能夠採用某種方法，其所展現出來的效果，能夠近似於在所有的策略中隨機挑選。

為達到這個目的，以下分成三個部份做討論：是否抓牌、如何叫牌、以及流程安排。

## 第二節 隨機玩法

### 一、是否抓牌

「對手的叫牌成不成立？」面對這樣的問題時，第一感通常是直接計算機率，若叫牌成立的機率大則判斷成立，反之則判斷不成立。以圖 4.2 為例，對手喊了「三個 5」，而我方手上有一個 5，那麼便去計算對手擁有為了使叫牌成立所

需要搖出的兩個(或以上)5 的機率。此機率大約是 0.2 左右。

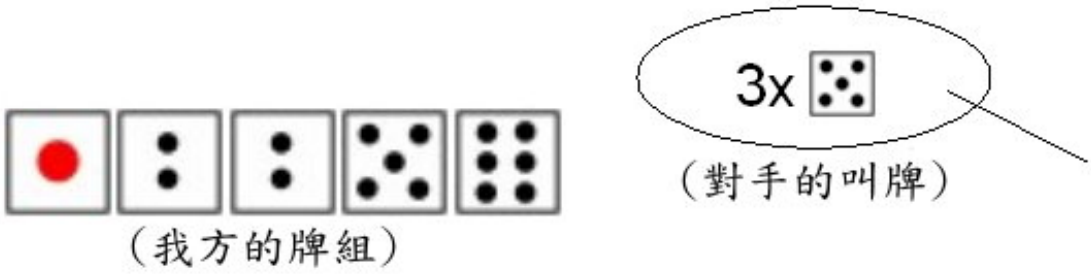



圖 4.2 我方考慮是否抓牌的例子

表 4.5 為對手擁有 k 個特定點數的骰子的機率。所需的骰子數為 k 時，機率的算法為：

$$C_k^5\left(\frac{1}{6}\right)^k\left(\frac{5}{6}\right)^{5-k}$$

要算 k 個以上的骰子的機率，便可從 k 個至 5 個的機率累加。

| 個數(k) | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 機率 | 1 | 0.5981 | 0.1962 | 0.0355 | 0.0033 | 0.0001 |

表 4.5 擁有 k 個特定點數骰子的機率

也就是說，在目前叫牌是需要對手提供兩個該點數骰子才能成立的情況下，若純按照機率而言，有 8 成的機會應該去抓牌。

然而，若實際去計算每種牌型的組合個數，如表 4.6[1]，可以發現到，大約有一半左右的機率，對手的手牌中，有某個點數是有兩顆以上的。

*Results at liar dice in increasing value*

| *Result* | *Number of ways* |
|---|---|
| Nothing (busted straight) | 480 |
| One pair—two dice the same, the other three different | 3600 |
| Two pairs of dice with the fifth die different | 1800 |
| Three of a kind—three dice the same, the other two different | 1200 |
| Low straight—1, 2, 3, 4, 5 | 120 |
| High straight—2, 3, 4, 5, 6 | 120 |
| Full house—three dice of one kind and two of another kind | 300 |
| Four of a kind with the fifth die different | 150 |
| Five of a kind | 6 |

表 4.6 各種可能牌型的種類數

使用機率的判定法的問題在於：對方是在看過了手上的牌、且擁有自由選擇權利的情況下，喊出了「三個 5」，而且對手的底牌有一半左右的機率是存在有點數是成對的，這就意味著或許不應該單純直接計算機率，而是要考慮到對手喊牌時的心態。

也就是在考慮抓牌問題時，真正的問題並不在於「對方可能有什麼樣的牌」，而是「對方為何喊這樣的牌」，對方所喊的牌的個數，跟他的手牌中所實際擁有的個數，之間的對應關係才是這個階段的真正問題。

我們希望得知這個對應關係，而在沒有任何可靠資訊的情況下，最佳做法就是「猜」。於是判斷是否要抓牌的做法如下：

對方喊牌「x 個 y」，我方就隨機從 0~5 中挑選一數當作對方所擁有的 y 的個數，將得到的數與我方擁有的 y 的個數相加，若大於等於 x 則不抓，反之則抓。

圖 4.3 為套用上述的方法，面對同樣的局面時可能會發生的兩種情形。

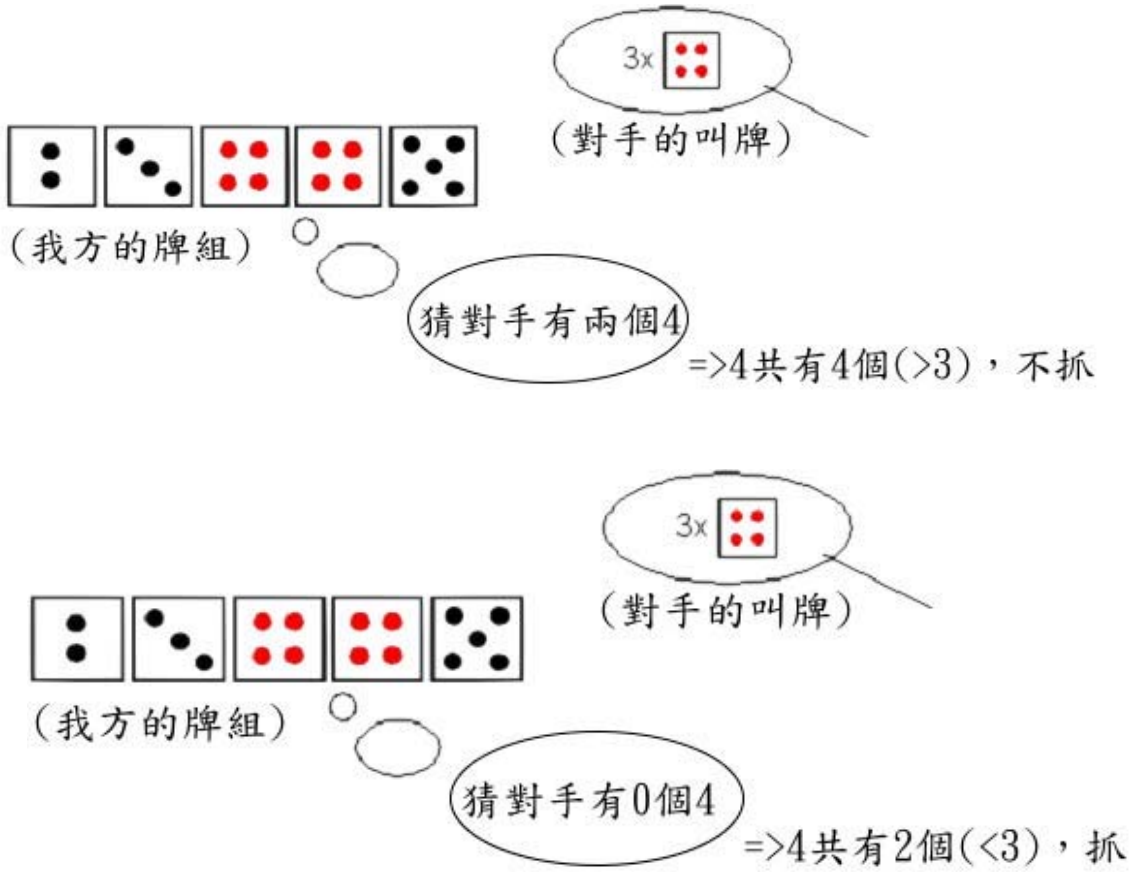



圖 4.3 判斷「是否抓牌」的方法

二、如何叫牌

與前一節相同，首先考慮的是我們的目的為何。在叫牌的階段，對我方最好的結果為「叫出成立的牌組，而對手抓牌，我方便能勝利」。這裡的目的有二:「所叫牌組成立」與「對手會抓」，首先考慮後者。

當我方叫牌後輪到對手時，對手遇到的也是跟我方同樣的問題—判斷我方喊牌的個數跟實際擁有個數的關係，並且以此為依據來決定要不要抓牌。那麼此時我方面臨的問題成了：對手會怎麼判斷我方個數。同樣的我們的做法是從 0~5 隨機選取一個來當作對手的猜測。

也就是將先前的判斷抓牌的方法逆轉，就成為判斷對手是否會抓牌的工具了。再來要找出候選的叫牌方式來讓這個工具評估。

這裡因為不能確定對手的牌，只能找出「期望上」成立的叫牌，也就是隨機產生對手的牌組，與我方搭配後找出所有成立的叫牌，由其中依照成立機率的比例隨機選取一個，再用上述的方法判斷對手會不會抓，若是符合「叫牌成立而對手會抓」的條件，則採用此種叫牌。

圖 4.4 為一叫牌的例子，對手叫牌「兩個 3」後，我方隨機產生一副對手的手牌，並找出所有大於「兩個 3」的牌組當做候選叫牌，接下來依照每個候選牌組成立機率的比例從中選取一個，選取的比例分配可參考表 4.7。例如「兩個 5」的叫牌需要對手提供一顆 5 才能成立，而對手擁有一顆以上的 5 的手牌種類數，根據表上所列為 4561 種，而「兩個 6」需要對手提供兩顆 6，可能種類數為 1526 種，依此類推。

假設我們選到「三個 4」，接下來則要再猜測對手的想法，從 0~5 隨機挑選一個數當作是對手所猜的「我方所擁有的 4 的個數」，若猜測為一個，再加上之前所產生的對手牌組中的一個，總共兩個，也就是我們認為，對對手而言「三個 4」並不成立而會選擇抓牌，這也就是我們所希望的，於是這個例子中，最後會喊出「三個 4」。

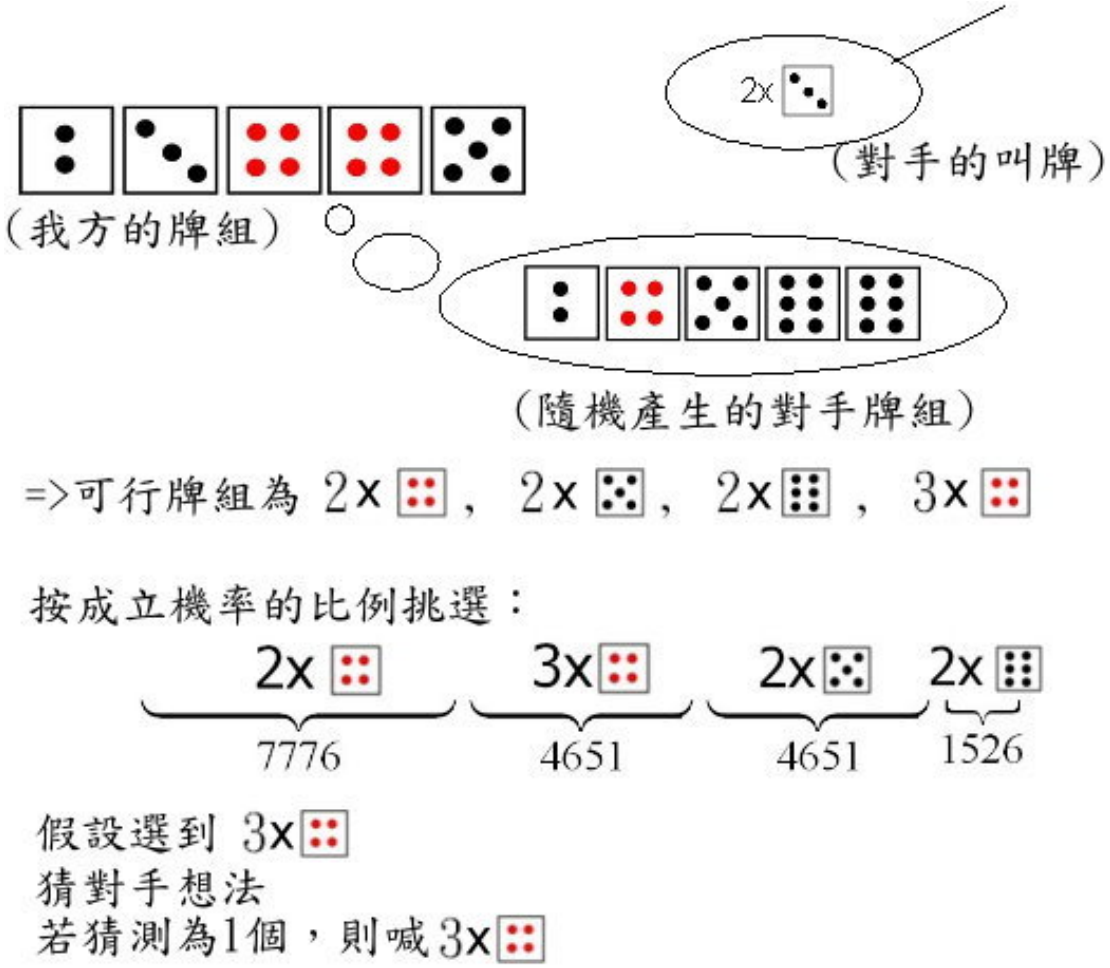



圖 4.4 判斷「如何喊牌」的方法

| 個數 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| 組合數 | 1 | 25 | 250 | 1250 | 3125 | 3125 |
| 累積數 | 1 | 26 | 276 | 1526 | 4651 | 7776 |

表 4.7 每種個數的組合數與累積數

三、流程安排

前面分別對於如何抓牌與如何叫牌的方法做了探討，但在每一個回合的行動中，抓牌與喊牌僅能選擇其一，那麼該如何選擇呢？

若只是單純按照固定的先後次序，則有如下的兩種選擇：

一、先決定要不要抓牌，若不抓的話再考慮要叫什麼牌。

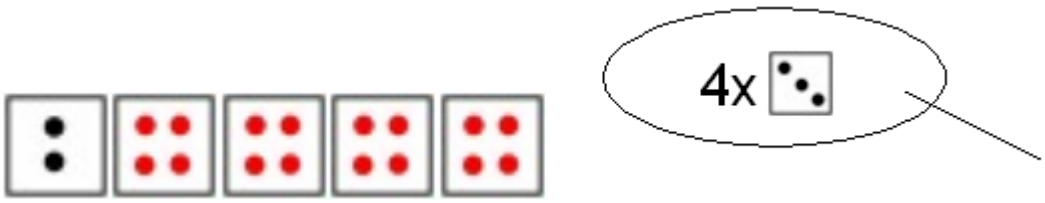



圖 4.5 先考慮抓牌再考慮喊牌的例子

如圖 4.5，對方喊「四個 3」，而我方手牌中沒有 3，依照抓牌的方法，大約有 2/3 的機率會判定為抓牌，但我方手牌中有 4 個 4，「四個 4」的叫牌為絕對安全。但若是在事先已判定為抓牌的情形下，絕對不會考慮喊出「四個 4」。

也就是說「優先考慮抓牌、再考慮喊牌」的作法，可能會有「明明有確定成立牌組卻絕對不去喊」的情況。

二、先看有沒有牌可叫，若無牌可叫再考慮是否抓牌

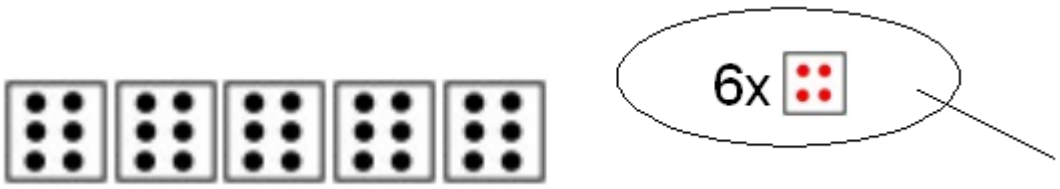



圖 4.6 先考慮喊牌再考慮抓牌的例子

考慮如圖 4.6 的極端狀況，對方喊了「六個 4」，而我方手牌為 5 個 6，則即使對方手牌為 5 個 4，「六個 4」也是絕對不可能成立的。但負責考慮喊牌的方法，會隨機假設對手的手牌，在我方手牌為 5 個 6 的情形下，一旦猜測的對方手牌中有 1 個 6，那麼「六個 6」便會成為候選叫牌。

也就是「優先考慮喊牌、再考慮抓牌」作法，可能會有「明明抓了就會贏，卻還是決定喊牌」的情況。

由上述兩點可以得知，以固定順序安排流程是有問題的。比較合理的想法是先各自決定好抓牌及叫牌的結果，再以某種原則選取兩者之一，而挑選的方法必須考慮到兩個的結果各自成立的機率，也就是希望擁有比較高正確率的一方也會有比較高的機率被選到。並且必須能夠一口氣解決上述提到的兩種固定流程時會遭遇的問題。

分別考慮過抓牌及叫牌的結果後，可能會有以下四種組合：

1. 決定不抓，且有牌可叫。
2. 決定抓牌，且無牌可叫。
3. 決定不抓，但也無牌可叫。
4. 決定抓牌，但也有牌可叫。

在情況 1 及情況 2，較不會有疑慮，因為僅有一種選擇，而在情況 3，則是無路可走的情況，不得已只好再重新考慮一次抓牌及叫牌。而情況 4 則是遇到兩種可選的路，雖然可以找到許多方法來決定如何挑選其中一種，但在這裡我們採取的方法是捨棄這一輪的處理並重新考慮一次抓牌及叫牌。依此流程直到有情況 1 或 2 出現時，就做出最後的決定。

## 第三節 貝氏信賴網路

因遊戲中無法獲得任何可靠的資訊，在這樣的條件下，玩家做決策時，往往會摻入自身的情感、直覺、偏見，於是對手行為模型(opponent modeling)也是一個可以考慮的方法。以建立對手行為模型的技術，使其有學習能力，能夠經由

重複對局，發現當前對手的行為模式及弱點，進而提高勝率。

在此考慮的方法是貝氏信賴網路，是一種資料探勘方法，常被使用在被觀察對象具有不確定性時。貝氏信賴網路(Bayesian Belief Network)，是以圖的方式來表達一群隨機變數之間的機率關係，主要由兩種元素組成：

1.一個有向、不循環的圖，代表變數彼此之間的相依關係。

2.每個節點都有一個機率表，關聯起節點與他們的父節點。

這個結構顯示出各個因素彼此間的互動關係，再藉由重複的觀測對網路進行訓練，便能夠推測出各個因素間相互影響的概略機率。

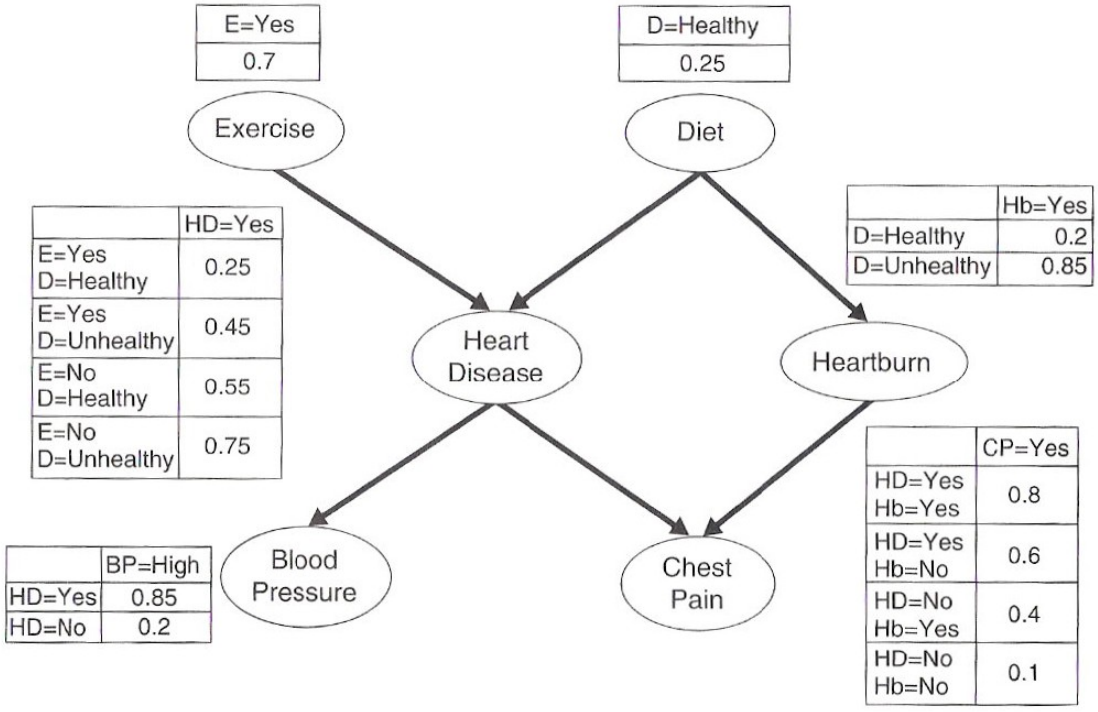



圖 4.7 描述心臟疾病的貝氏網路

圖 4.7 是一個描述心臟疾病病患身體狀況與生活習慣之間關連性的貝氏信賴網路模型，每個節點都各有兩種可能值。位於中層的兩個節點分別代表兩種疾病：心臟病(Heart Disease)及心臟灼熱(Heartburn)。它們的父節點代表生活習

慣，也就是不良的生活習慣可能導致這些疾病，例如飲食與運動等等。而兩種疾病的子節點，則是這些疾病所可能導致的外顯症狀。在圖中，心臟灼熱可能是由於不良的飲食習慣，並且會導致胸口疼痛[11]。

完成網路結構設計之後，接下來就是對網路進行訓練，每一個節點都存有一個紀錄與其相關節點的機率關係的表格，藉由重複對系統作觀測，統計出各項因素間是以什麼樣的機率互相影響著。一旦累積的觀測次數夠多，機率表格的內容便能夠逼近真實機率，也就能夠藉由對現有資訊的觀測，合理的以機率計算方法推測接下來的結果。

## 貝氏規則

貝氏規則是在使用貝氏網路計算機率的過程中，常會使用到的一個公式，將其說明如下。

若 A 和 B 是真實世界中的兩事件，假設 A 和 B 並不相互排斥，在一個事件已經發生的前提下另一個事件也可能在一定條件下發生。在事件 B 已經發生的前提下事件 A 發生的機率稱作條件機率。條件機率的數學運算式是 $p(A|B)$，完整的運算式可以解釋為「事件 B 已經發生的前提下事件 A 發生的機率」。其公式如下：

$$p(A|B)=\frac{A和B同時發生的次數}{B發生的次數}$$

A 和 B 同時發生的機率，稱為 A 和 B 的聯合機率。聯合機率數學運算式為 $p(A\cap B)$。若 B 發生的機率為 $p(B)$，則：

$$p(A|B)=\frac{p(A\cap B)}{p(B)} \qquad (1)$$

同樣，在事件 A 已經發生的前提下事件 B 發生的條件機率為

$$p(B|A)=\frac{p(B\cap A)}{p(A)}$$

因此

$$p(B\cap A)=p(B|A)\times p(A)$$

聯合機率具有可交換性，因此

$$p(A\cap B)=p(B\cap A)$$

所以

$$p(A\cap B)=p(B|A)\times p(A) \qquad (2)$$

將(2)帶入(1)，產生下式

$$p(A|B)=\frac{p(B|A)\times p(A)}{p(B)}$$

此公式即為貝氏規則(Bayesian Rule)，18 世紀英國的數學家 Thomas Bayes 首先提出這個規則，之後就以他的名字命名這規則[10]。

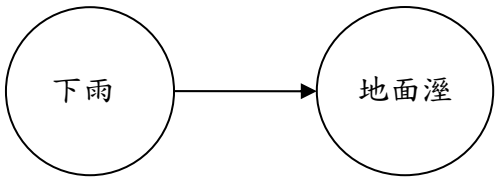



圖 4.8 描述天氣與地面乾溼關係的貝氏網路

圖 4.8 是一個觀察天氣的下雨與否跟地面的乾溼情形的簡單貝氏網路。

我們在其中套用一些實際數字，假設我們在過去的連續 100 天觀察了天氣與

地面的乾溼狀況，其中有 37 天有下雨，在這 37 天的雨天中，29 天的地面是溼的，而 63 天的晴天中，18 天地面是溼的。

經過這 100 天的觀察，在此貝氏網路中的兩個節點所各自存放的機率表格，內容將如下。

「下雨」節點的機率表，如表 4.8。

| 下雨 | |
|---|---|
| 是 | 否 |
| 37 | 63 |

表 4.8「下雨」節點的機率表

「地面溼」節點的機率表，為包含下雨與否條件的條件機率表，如表 4.9。

| | | 地面溼 | |
|---|---|---|---|
| | | 是 | 否 |
| 下雨 | 是 | 29 | 8 |
| | 否 | 18 | 45 |

表 4.9「地面溼」節點的機率表

到了第 101 天，這時我們希望可以靠著觀察地面有沒有溼，來猜測今天有沒有下過雨。如果觀察的結果，地面是乾的，則今天下過雨的機率為：

$$
\begin{aligned}
&p(\text{下雨}=\text{是}\,|\,\text{地面溼}=\text{否}) \\
&= \frac{p(\text{地面溼}=\text{否}\,|\,\text{下雨}=\text{是})\,p(\text{下雨}=\text{是})}{p(\text{地面溼}=\text{否})} \\
&= \frac{\left(\frac{8}{29+8}\right)\left(\frac{37}{37+63}\right)}{\left(\frac{18+45}{100}\right)} = 0.15
\end{aligned}
$$

這代表今天有比較高的可能性是晴天[13]。

將這項技術套用在吹牛骰子中，困難點在於能否找出所有會影響對手行為的因素，來架構出一個符合真實情形的信賴網路。這可能牽涉到很多層面，對每個玩家而言，影響其行動的因素也不盡相同。

## 網路結構與分析

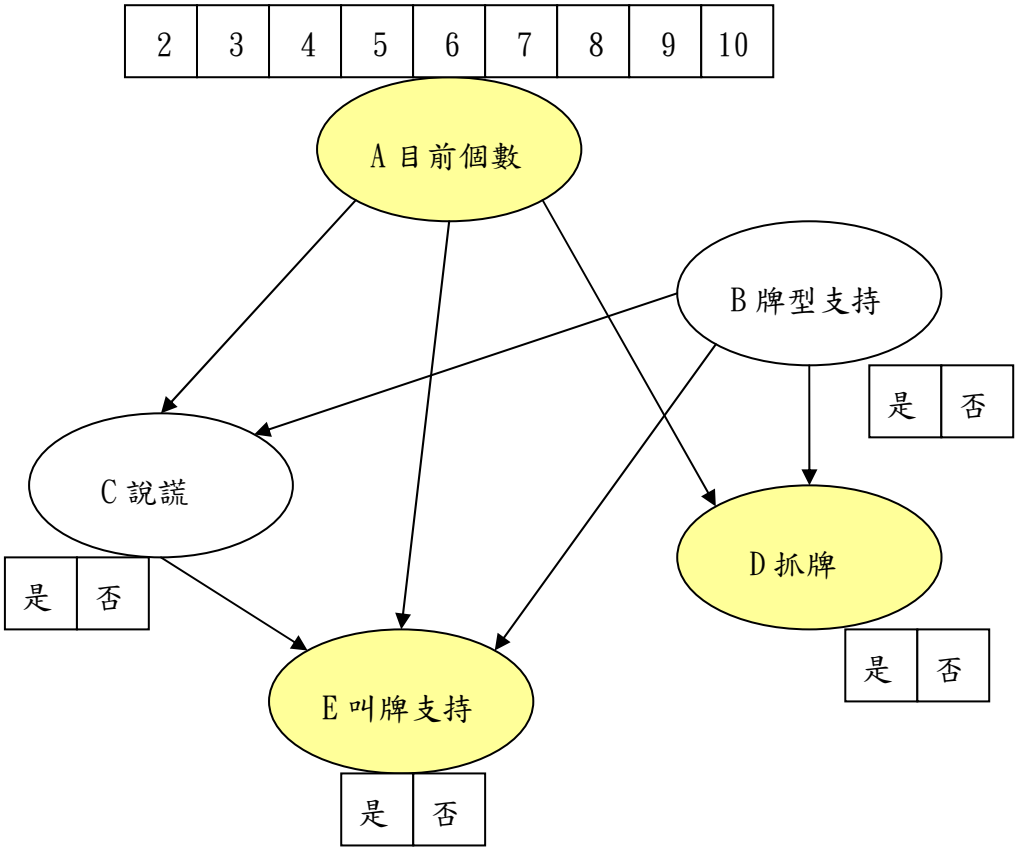



圖 4.9 描述吹牛骰子遊戲的貝氏信賴網路

圖 4.9 是針對吹牛骰子所建立的貝氏網路模型，我們設法找出在每一回合可能會影響玩家的參數及所需要做的決策，並根據其相依性做出連結。

A：前一名玩家所叫的牌的個數，值為 2~10。

B：玩家的手牌是否支持前一家玩家所喊的點數。在這裡我們把支持定義為：手

上擁有的數目跟對手所喊的數目差距小於 3，例如對手喊了「三個 5」而己方有 1 個，便算是支持。

C：玩家所決定的叫牌是否為說謊。說謊的定義為，所喊出的個數跟實際擁有的個數差距大於 1，例如喊出「三個 5」而手上有兩個便算是誠實，而若只有一個則算是說謊。由於六種點數分配在五顆骰子上的期望值為 5/6，大約一顆，也就期望對方提供一顆骰子來達成牌組是合理的，但期望兩顆以上便不合理，於是做如此定義。

D：玩家的叫牌是否支持前一家玩家所喊的點數。

E：玩家是否採取抓牌的行動。

節點 A 與 B 是輪到玩家時所面臨的局面，而節點 C、D、E 三項則是玩家所要做出的決定。

玩家結束一回合的行動之後，所透露出的資訊有 A、D、E 三項，於是我們接下來所要做的，就是藉由 A、D、E 三項資訊，與過去跟該名玩家對戰時所累積的機率值表，分別算出節點 B 與節點 C 的機率。計算的推導細節如下。

節點 B：牌型是否支持？

$$
\begin{aligned}
&p(B \mid ADE) \\
&= p(BC_0 \mid ADE) + p(BC_1 \mid ADE) \\
&= \frac{p(D \mid ABC_0)\,p(BC_0 \mid AE)}{p(D \mid A)} + \frac{p(D \mid ABC_1)\,p(BC_1 \mid AE)}{p(D \mid A)} \\
&= \frac{p(D \mid ABC_0)\,p(E \mid AB)\,p(BC_0 \mid A)}{p(D \mid A)\,p(E \mid A)} + \frac{p(D \mid ABC_1)\,p(E \mid AB)\,p(BC_1 \mid A)}{p(D \mid A)\,p(E \mid A)} \\
&= \frac{p(D \mid ABC_0)\,p(E \mid AB)\,p(C_0 \mid AB)\,p(B)}{p(D \mid A)\,p(E \mid A)} + \frac{p(D \mid ABC_1)\,p(E \mid AB)\,p(C_1 \mid AB)\,p(B)}{p(D \mid A)\,p(E \mid A)}
\end{aligned}
$$

在計算之前就知道的資訊為 A、D、E 三項，所以要計算的為 $p(B \mid ADE)$，而在計算節點 B 的機率時，節點 C(說謊)是不明朗的資訊，但它的值仍會對其他節

點產生影響，所以在計算的第二行，必須將節點 C 的每種情況都考慮進來，其中 $C_0$ 代表「說謊」、$C_1$ 代表「不說謊」。

節點 C：有無說謊?

$$
\begin{aligned}
&p(C \mid ADE) \\
&= p(B_0C \mid ADE) + p(B_1C \mid ADE) \\
&= \frac{p(D \mid AB_0C)\,p(B_0C \mid AE)}{p(D \mid A)} + \frac{p(D \mid AB_1C)\,p(B_1C \mid AE)}{p(D \mid A)} \\
&= \frac{p(D \mid AB_0C)\,p(E \mid AB_0)\,p(B_0C \mid A)}{p(D \mid A)\,p(E \mid A)} + \frac{p(D \mid AB_1C)\,p(E \mid AB_1)\,p(B_1C \mid A)}{p(D \mid A)\,p(E \mid A)} \\
&= \frac{p(D \mid AB_0C)\,p(E \mid AB_0)\,p(C \mid AB_0)\,p(B_0)}{p(D \mid A)\,p(E \mid A)} + \frac{p(D \mid AB_1C)\,p(E \mid AB_1)\,p(C \mid AB_1)\,p(B_1)}{p(D \mid A)\,p(E \mid A)}
\end{aligned}
$$

與計算節點 B 時的情形相同，在計算節點 C 的機率時，節點 B 成了不明朗的資訊，於是也必須將節點 B 的每種情況考慮進來，其中 $B_0$ 代表「牌型支持」、$B_1$ 代表「牌型不支持」。

然而上面的貝氏網路並不足以囊括所有的情形，例如在玩家首先開叫的情況下，之前並沒有其他玩家叫過牌，自然也不會有 A(目前個數)與 B(牌型支持)這兩項資訊，於是關於玩家首先開叫的歷史資訊，必須保存在另一個貝氏網路中，如圖 4.10(a)，此網路僅包含一個節點，用來記錄玩家的開叫是否說謊。

另一個情況是在我們所製作的程式首先開叫的時候，同樣因為並沒有之前玩家叫牌的資訊，無法套用上面的貝氏網路做計算，我們將程式首先開叫的資訊記錄在圖 4.10(b)的網路中，紀錄的是程式開叫的個數與隨後被下家玩家抓牌的機率關係。

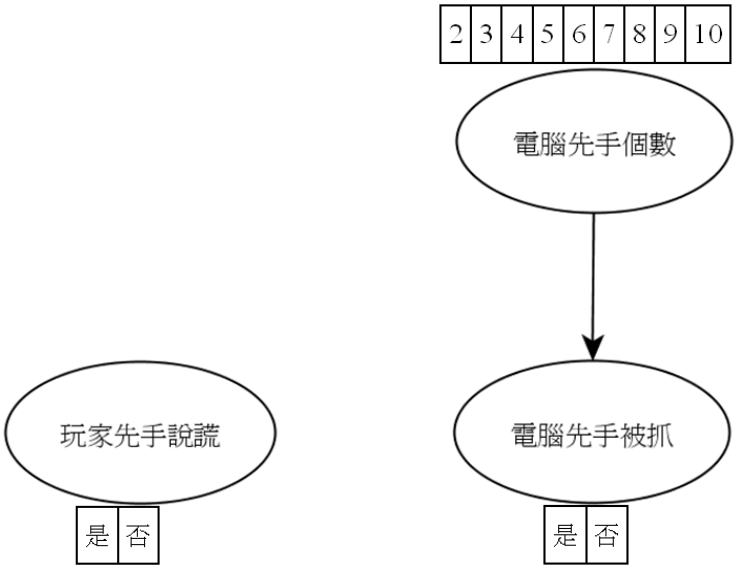



圖 4.10 描述開叫情形的貝氏網路

目前設計出來的三個貝氏信賴網路，經過計算之後所能夠得到的結果共有四項：

1. 玩家開叫說謊的機率
2. 玩家的牌型支持上家叫牌的機率
3. 玩家的回應為說謊的機率
4. 程式的叫牌會被抓的機率

接下來要思考的是如何運用這些機率來輔助我們做決策。

一開始要決定的是要不要採信該網路所得到的結果，貝氏網路是一種藉由長期訓練來提升效力的工具，所以在玩家遊戲初期所得到的機率分配或許並不十分正確，所以在初期可能較不採信網路的結果，而隨著遊戲場數的累積慢慢加重採信的比例。

我們採用紀錄開叫的貝氏網路的門檻為：

$$\frac{(\text{開叫次數})}{(\text{開叫次數})+10}$$

採用紀錄叫牌的貝氏網路的門檻為：

$$\frac{(\text{在目前個數爲}k\text{的情況下做決策的次數})}{(\text{在目前個數爲}k\text{的情況下做決策的次數})+10}$$

遊戲初期採用的機會為 0，而當同樣的情形出現第二次，就有 $\frac{1}{1+10}\approx 0.09$ 的機率採用，當同樣的情況出現 10 次，接下來採用的機率就有 $\frac{10}{10+10}=0.5$，採用的機率將隨著相同情況的累積漸漸提高。

若決定採信之後，再來要以得出的機率進行對手底牌的猜測與抓牌叫牌的輔助。

以下說明在採信網路提供的機率時，對於對手底牌的推論方法。

在每局遊戲一開始，我們先安排一個空的假想牌組，並藉著在遊戲中得到的關於對手的資訊，慢慢的將這個空牌組填滿。

另外有一個陣列，用來記錄我們所認為的對手底牌各點數的個數上限，以-1當作不限制。例如當陣列內容為{-1, -1, 2, 2, -1, -1}時，代表我們在接下來所猜的對手底牌，3 點與 4 點頂多有兩個，其餘點數不限制。

在每回合玩家做出行動後，程式經由貝氏網路的機率計算得到一組對手的假想牌組(通常個數不足五個)，接下來將用這組牌來輔助進行決策，本章前半部份所使用的尋找隨機解法的過程中，每當需要猜牌的場合，便以這組假想牌組為基礎。

以下舉一例說明在各種場合的猜牌方式。

在每局遊戲開始時設立一個空的假想牌組以及一個限制陣列，陣列中的六個 -1 代表目前對六種骰子點數都不做限制，如圖 4.11。

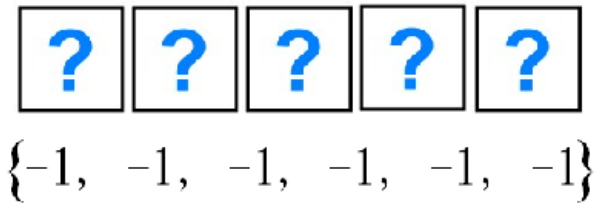



圖 4.11 假想牌組與限制陣列的初始情形

當採信對手的開叫時，將假想牌組加入一個，如圖 4.12，當貝氏網路認為對手開叫所喊的「兩個 3」是可信的，則在假想牌組中加入一個 3。

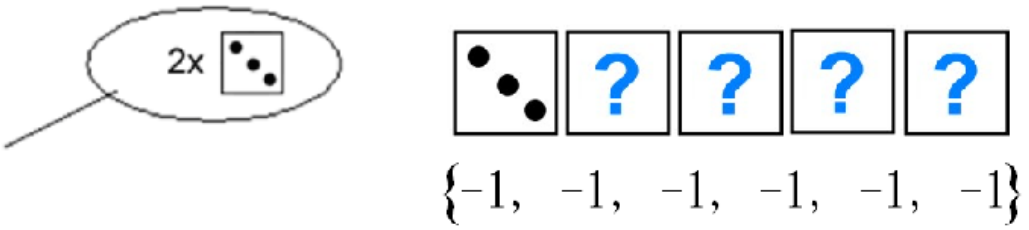



圖 4.12 採信對手開叫時的情形

當我方做出叫牌動作後，若根據對手回應而認定對手牌型支持時，則將假想牌組補足至「程式所喊的個數-2」，如圖 4.13，我方喊出「三個 5」後對手回應「三個 6」，若根據此回應認為對手支持我方所喊的「5」，則在假想牌組中加入一個 5。

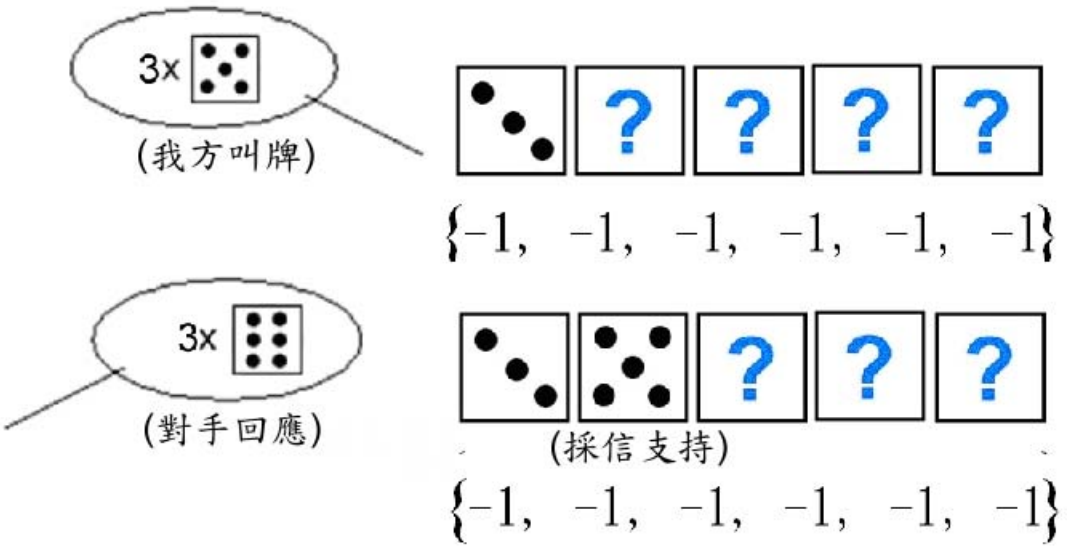



圖 4.13 認為對手支持時的情形

若接著認為對手叫牌可信，則將假想牌組的個數補足至「玩家所喊的個數-1」，如圖 4.14，對手叫牌「三個 6」，若認為是實話，則在假想牌組加入兩個 6。

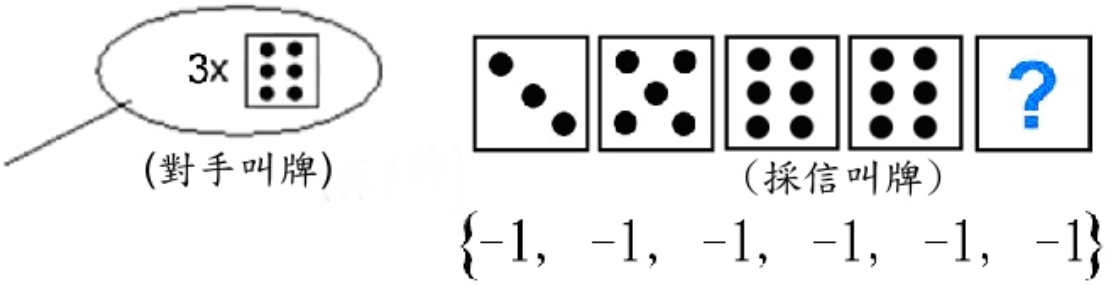



圖 4.14 採信對手叫牌時的情形

下例是各種不採信的處理方式介紹。

當不採信對手開叫，則將個數限制在「開叫個數-2」，如圖 4.15，我方不採信對手開叫的「兩個 3」，便將限制陣列的第三個數字設為 0。

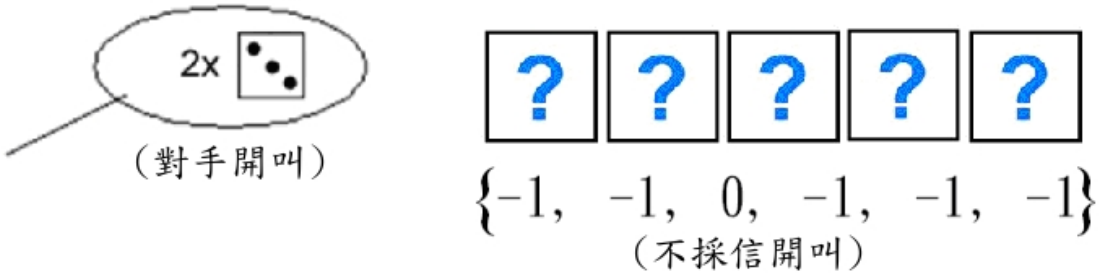



圖 4.15 不採信對手開叫時的情形

當我方做出叫牌動作後，若根據對手回應而認定對手牌型不支持，則將個數限制在「程式所喊的個數-2」，如圖 4.16，認為對手並不支持我方所喊的「三個 5」時，將 5 的個數限定在一個以下。

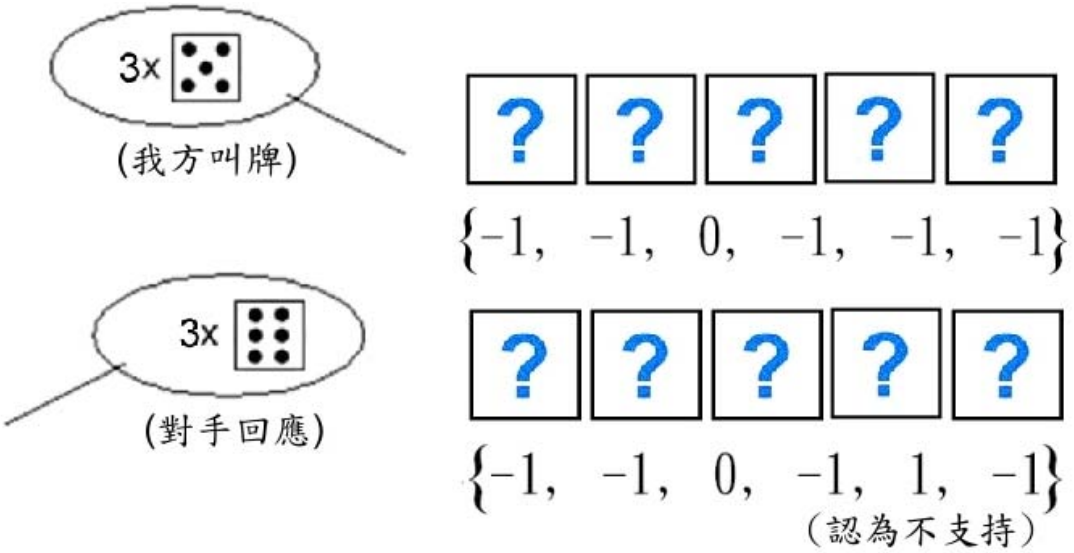



圖 4.16 不認為對手支持時的情形

若接著不採信對手叫牌，則將個數限制在「玩家所喊個數-2」，如圖 4.17，認為對手所喊「三個 6」不可信時，將 6 的個數也限定在一個以下。

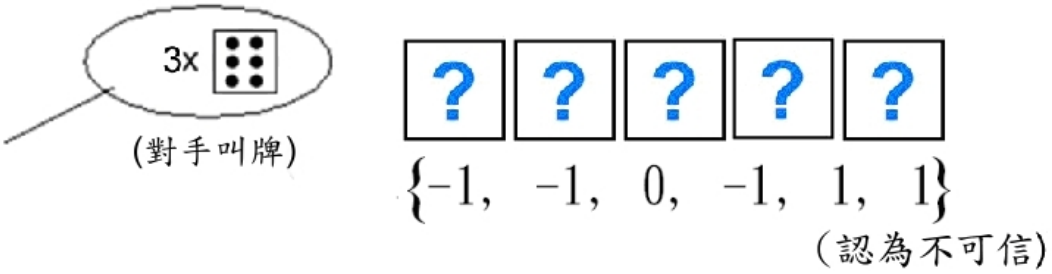



圖 4.17 不採信對手叫牌時的情形

在每回合的決策中皆以這種方式，一步步的猜測對手可能的牌型，如此便能減少不確定的資訊量，進而做出更有把握的決策。

# 第五章 實驗及結果

## 第一節 系統研發

目前並未找到他人有實作過吹牛骰子遊戲的系統，為方便系統測試、除錯與展示，我們自行實作了純文字介面的電腦自動對局程式與圖形介面的人機對局程式。

測試程式所使用的手法為模擬人類玩家在進行遊戲時可能採取的玩法，得出幾種簡易分類。對於吹牛骰子遊戲，以下列五點來當作參數：

1.「誠實 or 說謊？」：輪到自己叫牌時，有比較大的機率說實話或謊話? 若設定為實話，則每次的喊牌都有 70%的機率喊擁有個數最高的牌，30%的機率喊自己所缺門的牌。設定為謊話時則相反。

2.「信賴 or 猜忌？」：對於對手的叫牌，比較傾向相信與否? 若設定為信賴，則對於對手的喊牌，會隨機認為對手擁有的個數為「等於對手所喊的」或「對手所喊的減一個」。設定為猜忌時則隨機認為對手的個數為「對手所喊的減一個」或「對手所喊的減兩個」。

3.「上限高 or 低？」：對於一局遊戲，所能夠容忍的最大叫牌個數?也就是到達某個上限個數就必定會抓，設定為高時，上限會隨機設定為 5 或 6，設定為低時，則隨機設定為 4 或 5。

4.「攻擊強 or 弱？」：即會不會有「跳叫」的行為發生?例如對於「兩個 3」，接著喊「兩個 4」便可以壓過，若喊出「三個 4」便算是跳叫的喊法。設定為在有牌可喊的情況下，絕不跳叫、或有一半機率跳叫。

5.「傾向抓牌 or 叫牌？」: 此測試程式會先分別尋找抓牌及叫牌的可能性再做挑選，此項參數決定程式將優先選擇何者。

此五種參數各有 2 種選項，總共可以得出 32 種對局程式的組合，下舉一例說明測試程式的運作流程：

假設此測試程式的性格設定為：誠實、猜忌、上限低、攻擊弱、傾向抓牌。

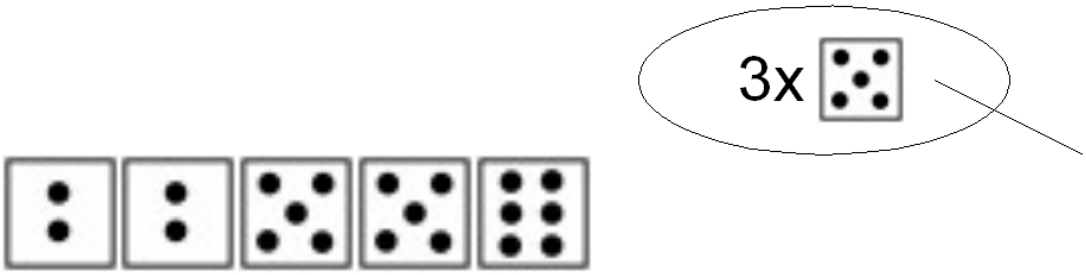



圖 5.1 測試程式的決策流程例子

則碰到如圖 5.1 的狀況時，此測試程式會有下列行為：

1. 個數上限為四，大於目前叫牌的個數三 ➔ 不抓。
2. 猜忌,認為對方只有一個 5。加手上的兩個 5 共三個➔抓牌設定為「否」。
3. 誠實 ➔ 喊目前個數最多的 5(在上一步認為總共有三個)。
4. 弱攻擊，喊四個 ➔ 叫牌設定為「四個 5」。
5. 傾向抓牌，但抓牌為「否」，轉而選擇叫牌 ➔ 最後喊出「四個 5」。

以這 32 種組合去實作一個文字介面的對戰程式,可以輸入對手的類型與對局場數後自動與根據第四章所研發的程式進行對局。運作畫面如圖 5.2 所示。

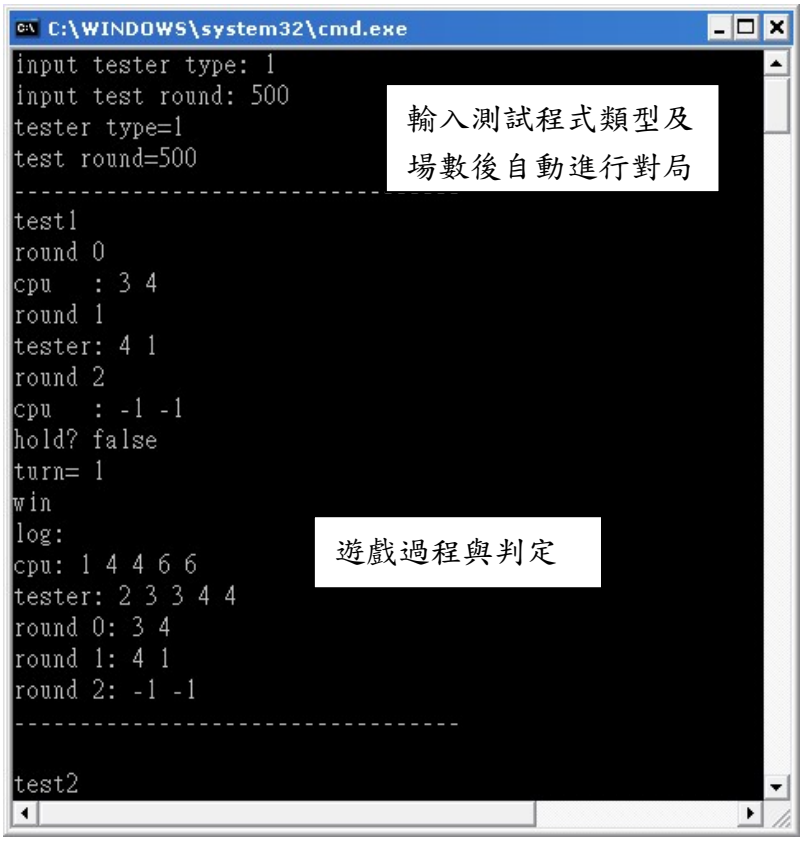



圖 5.2 文字介面測試程式運作畫面

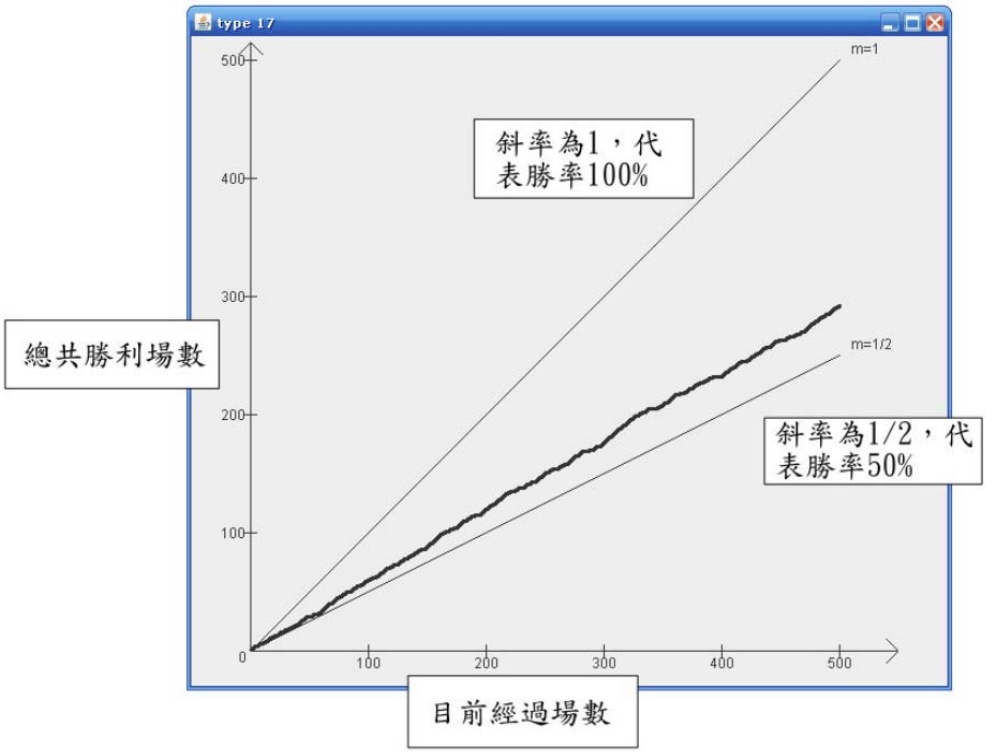



圖 5.3 勝率走向圖說明

圖 5.3 為勝率走向圖的說明，我們的目的是希望所研發的系統的勝率至少有 50%，也就是累積場數的走向能大致被夾在斜率為 1/2 及斜率為 1 的直線中間。

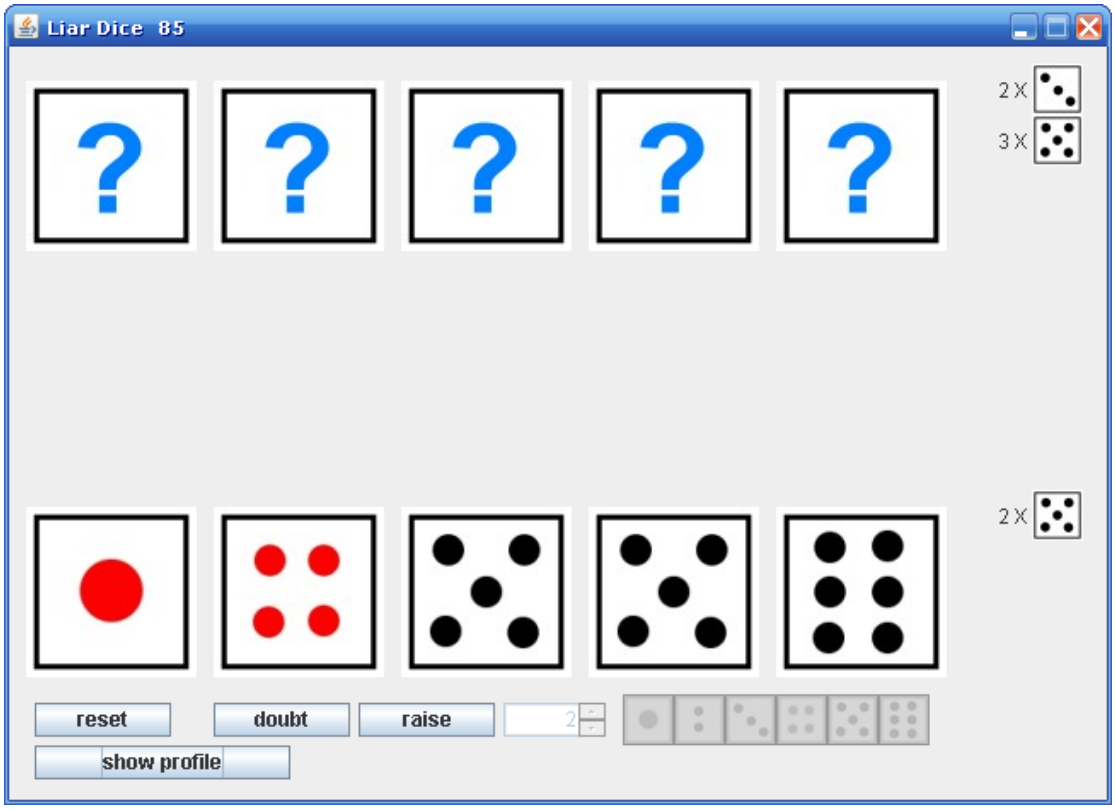



圖 5.4 吹牛骰子系統人機介面

圖 5.4 為可供人類玩家進行測試的吹牛骰子系統介面外觀。按鈕功能說明如下。

reset：重新開始一局新的遊戲

doubt：抓牌

raise：叫牌

show profile：顯示當前玩家的資訊(戰績、貝氏網路、各種行動的機率、猜測牌組)

數字旋鈕：決定叫牌的個數

骰子鈕：決定叫牌點數

按下「show profile」鈕後將顯示如下畫面：

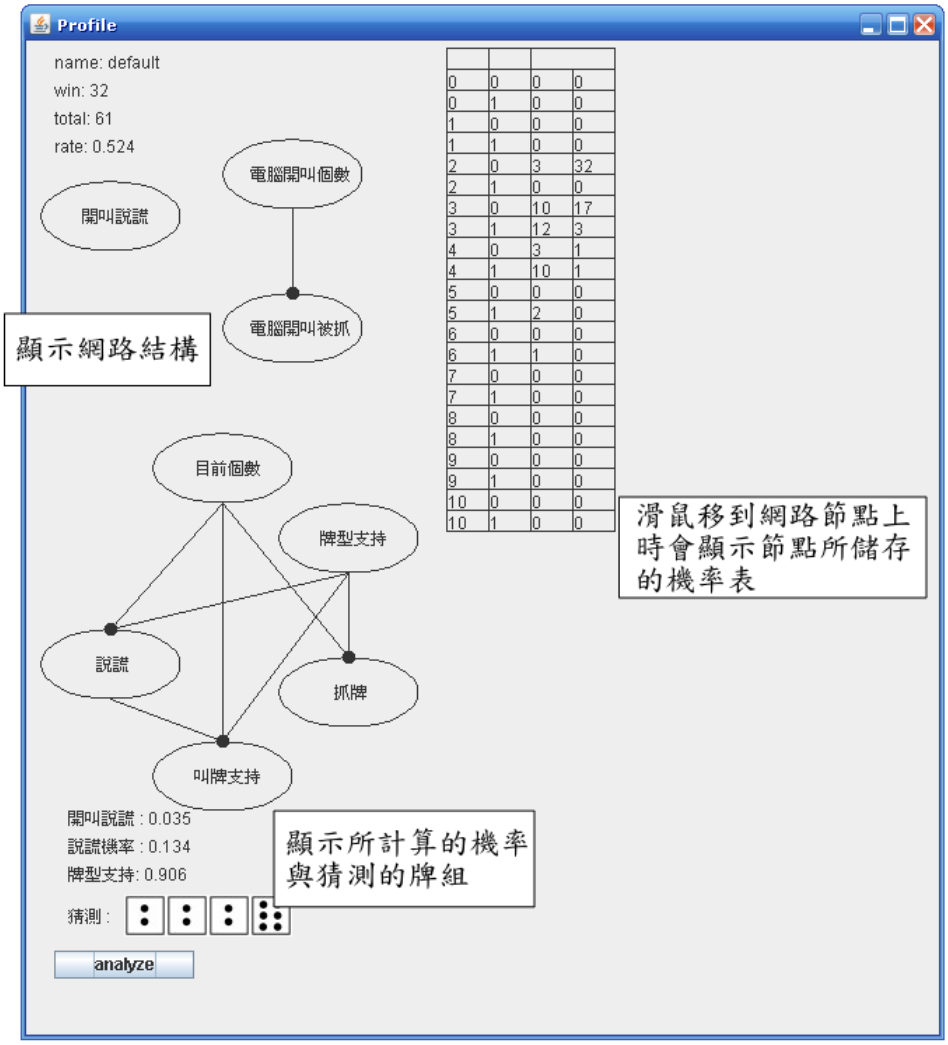



圖 5.5 玩家資訊畫面

## 第二節 測試結果

我們的測試分為兩部份，在第一部分僅使用隨機玩法的主程式分別與 32 種測試程式及人類玩家對戰，第二部份則加入貝氏網路的輔助。

以僅使用隨機玩法策略的主程式去與這 32 種測試程式進行對戰，都可以得到約 6~7 成的勝率。雖然這些對手僅是遵從簡單的規則，但我們盡量顧及到了各種玩家在遊戲中可能展現出的性格，顯示了對於這 32 種有鮮明差異的玩家性格，我們所使用的隨機玩法並不會輸給任何一種。

圖 5.6 至 5.8 列出與三種測試程式對局的結果，完整結果列於附錄中。

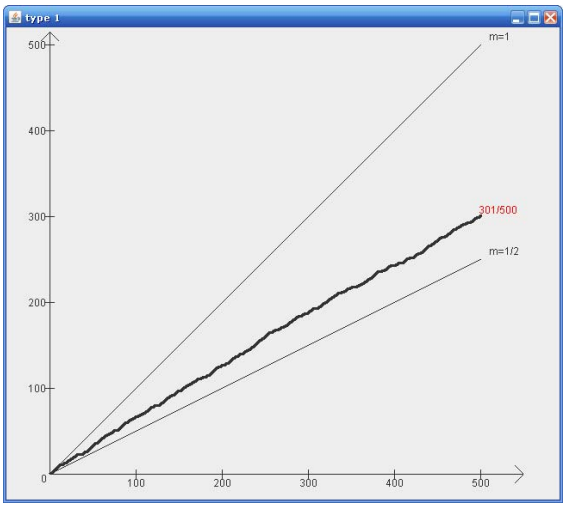



圖 5.6 對 type1 測試程式之戰績

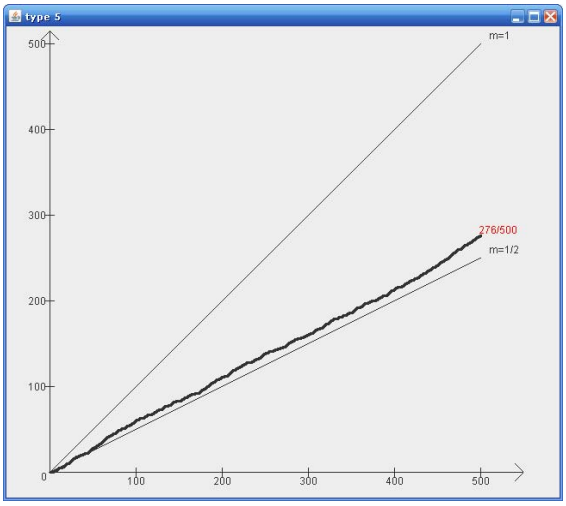



圖 5.7 對 type5 測試程式之戰績

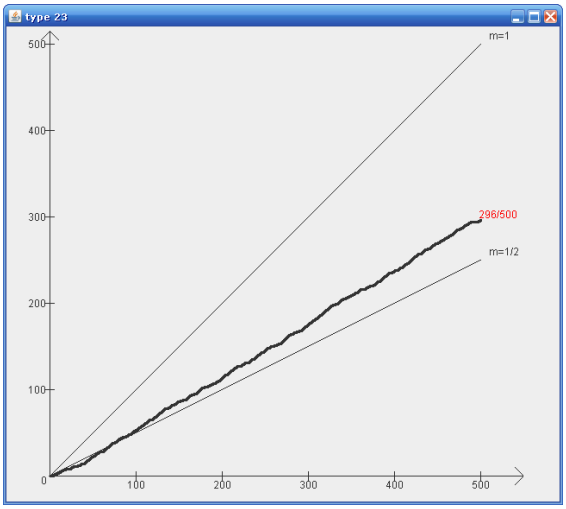



圖 5.8 對 type23 測試程式之戰績

而當隨機玩法的主程式再加入貝氏網路的輔助後，對於大部分的測試程式，勝率都提高了約 5%~10%，顯示了貝氏網路的確能夠從重複的對局中，發現對手的行為模式，對於對手的底牌能夠得到一定程度準確的猜測，並利用在決策的輔助上以提高勝率。但在少部分的測試類型中，貝氏網路並沒有明顯地提升勝率。

圖 5.10 至 5.12 列出三種測試程式在套用貝氏網路前後戰績的比較，完整結果同樣列於附錄中。

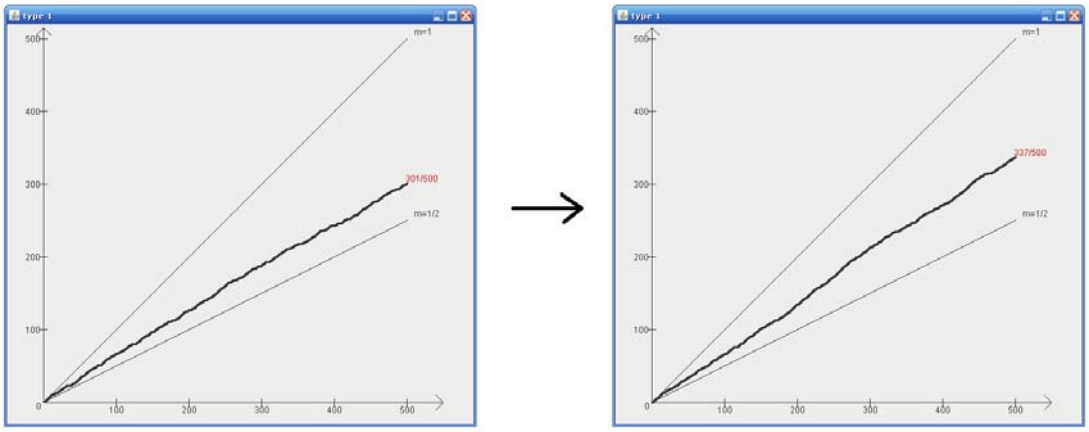



圖 5.9 套用貝氏網路前後對 type1 測試程式之比較

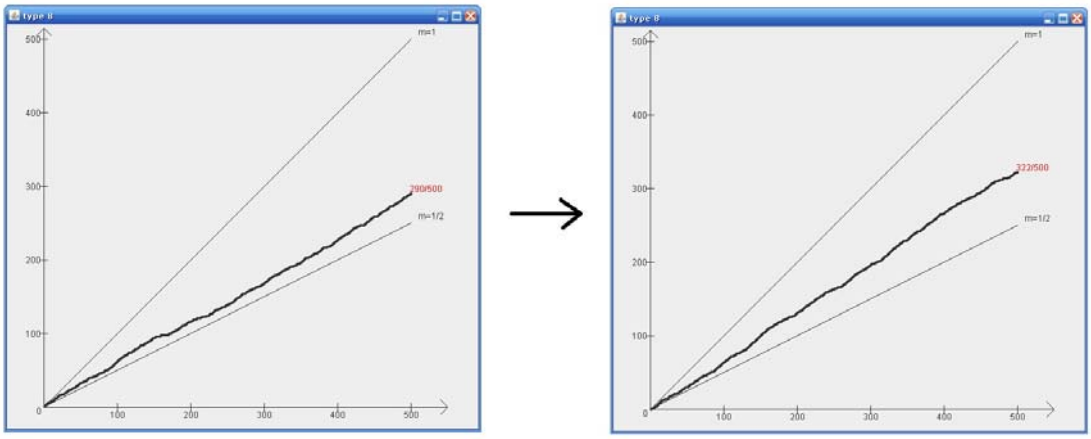



圖 5.10 套用貝氏網路前後對 type8 測試程式之比較

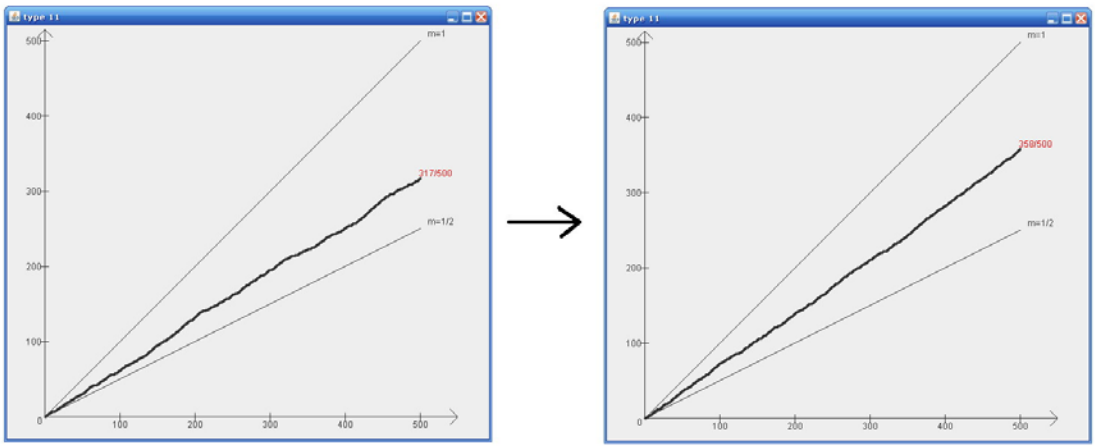



圖 5.11 套用貝氏網路前後對 type11 測試程式之比較

我們也找了四位人類玩家進行測試，測試結果如表 5.1。

| 玩家名稱 | 隨機玩法 | 搭配貝氏網路 | 差異 |
|---|---|---|---|
| Ad0219 | 49% | 52.5% | +3.5% |
| Gozha | 43.5% | 46% | +2.5% |
| Julian | 42.5% | 43% | +0.5% |
| Scott | 48.5% | 43.5% | -5% |

表 5.1 程式對四位人類玩家所取得的勝率

在僅使用隨機玩法的情形下，對於每位人類玩家的測試，都取得了 40%以上、最高有將近 50%的勝率，展現出約略足以與人類玩家相抗衡的實力。但在加入貝氏網路之後的程式，並非對每位玩家都有效果，甚至在對上其中一位玩家時勝率反而下降。

我們所建立的貝氏網路，無論是對電腦程式或是對人類玩家，皆有一部分的測試結果並不十分理想，實際去觀察一些對局的內容，也發現總是會有一些高估或低估的情形，沒有辦法很準確的去猜測對手的牌型。這可能與貝氏網路在設計上的瑕疵有關，這部份將在第六章進行討論。

# 第六章 結論與未來發展

在過去對於吹牛骰子遊戲，僅有在 individual hand 類型上有零星的研究成果。而本研究首度針對 common hand 類型，先以賽局理論的方式分析，認為隨機玩法能夠確保不吃虧，並希望找出一種玩法可以達到近似隨機的效果。在不吃虧的目標下，以貝氏信賴網路建立對手模型，藉由重複對局累積對手習性、偏好的理解。

以本論文中的方式所實作的吹牛骰子系統主程式，對三十二種測試程式進行實驗，實驗結果說明，我們所找出的以隨機猜測為基礎的玩法，能夠比其他使用固定規則的測試程式有更好的表現，對於人類玩家，也展現出了能與之抗衡的水準。

本研究進展到目前的地步，仍有許多尚可改進的空間。在規則上，本研究的對象是兩人 common hand 類型的最簡單型態，但通常此遊戲是多人進行的，且為了增加趣味性會加入延伸規則。所以接下來可考慮在人數或規則上進行擴充。「1 點當作王牌」及「懲罰機制」屬於玩家必須遵守，較無變化性的規則，只需要對原有規則稍作修改，應是較簡單的部份。而「重新搖骰」的規則就牽涉到較複雜的機率及策略，是屬於完全不同的課題。

在尋找隨機解的部分，當需要考慮是否抓牌時，我們所採取的方法是以均等的機率挑選 0~5 當作對手擁有的骰子數，而完全忽略了牌型成立機率造成的影響。然而實際上，通常在思考對策的時候是會連帶考慮機率的，例如「他敢喊到這麼高可能個數不少，可是應該很難搖到這麼多吧，所以……」諸如此類的過程。所以或許真正的隨機玩法，是能夠同時考慮對手心理的隨機性與骰子的機率性來達成的。其他在尋找叫牌、與流程安排的方法，也都可以很容易的找到不只一種

能夠大致上說是隨機的玩法，而本論文中所採用的，是經過實驗後認為能夠達到的最好組合，或許還能找到其他的更好玩法策略是我們所從沒考慮過的。

貝氏網路是一種牽涉到主觀意見，由不同的人的觀點會造成不同結構的工具，例如「說謊」及「牌型支持」的定義，且本論文因考慮到計算上的複雜度，在各方面做了簡化和妥協，舉例來說，我們設計的網路忽略了「對手是以何種心態面對我方的叫牌」，對手叫牌支持我方時，有時並不代表對手的牌型真的支持，而只不過是「對手的個性傾向相信我方」而已，像這樣在目前版本中所計算出來的某個節點的機率，其中可能實際上混雜了各種資訊，而使得計算的資訊意義不大。於是貝氏網路所發揮的效用是有一些，但並不十分顯著，尤其對於懂得視情況隨機應變的人類玩家，結構簡單的貝氏網路並不足以描述他們細膩的心理變化，可能要對於網路進行更細緻的設計，才有足夠的能力描述各種情況。

人類玩家在進行遊戲時所抱持的心態可能也有所影響，在事先已被告知所面對的程式是有學習能力的情況下，當玩家發現到程式正在根據自己的行為模式做出策略調整時，那麼就可以故意做出誘導的舉動使程式誤判，此時貝氏網路的存在反而成為弱點。而且即使是沒有事先告知，較敏銳的玩家仍有可能在遊戲中察覺程式行為。如何將這一點納入考量，也是一項值得探討的議題。

而理論上應該存在一個能夠完美描述吹牛骰子賽局的貝氏網路，遊戲中的所有要素及相依關係都被精確的找出來，建立起真正的對手模型。未來或許可再探討修正。

本論文所使用的方法，都是奠基在「隨機猜測」上，並且都是以整數當作猜測的單位，而在經過大量實驗後發現，以整數猜測有可能會造成太武斷的結果，由於玩家的手牌總共也才五顆骰子，某點數的骰子多一個少一個都會對決策造成很大的影響，因此或許可以考慮以實數來進行猜測。例如當需要猜測整副手牌

時，可以將長度為 5 的數線隨機分割為六段，以每段的長度當做猜出的個數，如圖 6.1 所示。

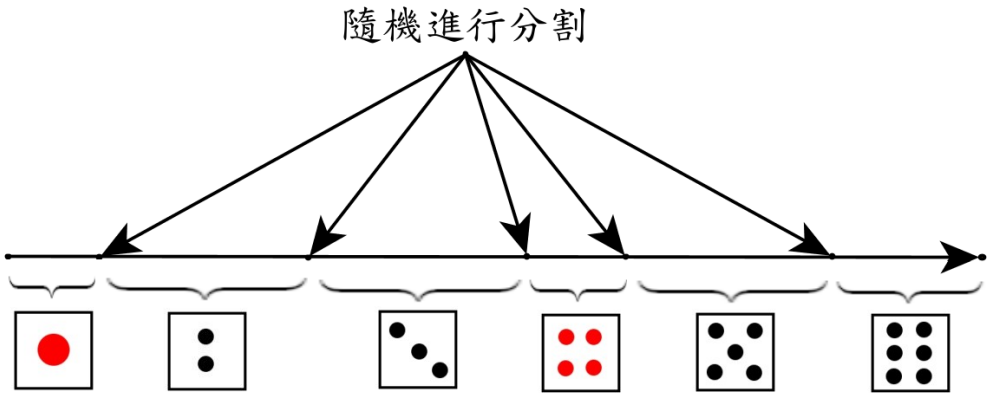



圖 6.1 以實數進行猜牌

同樣的方法也可使用在貝氏網路上，由於貝氏網路所計算出的對手行為有信賴度的高低之分，於是在假想牌組上或許也可以根據信賴度進行實數而非整數的加權。

雖然還有許多待改進的地方，但本論文所採用的以隨機策略，搭配對手行為模型的方式，的確發揮了一定程度的效用。或許未來也可將相同的流程，套用在其他的不完全資訊遊戲上。

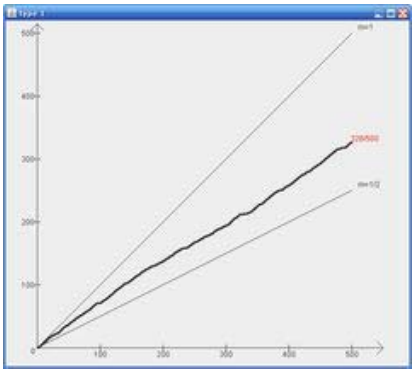
# 附錄　詳細測試結果

以下為 32 種不同性格(如下表)的測試程式，與我們所實作的吹牛骰子系統主程式各對戰 500 局後所得到的戰績走向圖。

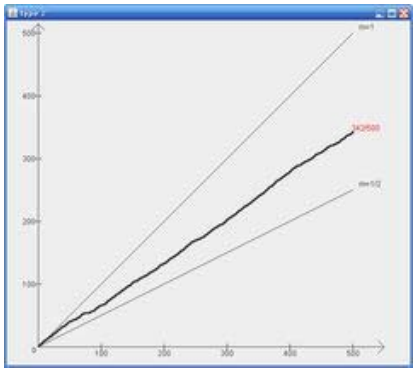
| | |
|---|---|
| type1 | 誠實、信賴、上限高、攻擊弱、傾向抓牌 |
| type2 | 誠實、信賴、上限高、攻擊強、傾向抓牌 |
| type3 | 誠實、信賴、上限低、攻擊弱、傾向抓牌 |
| type4 | 誠實、信賴、上限低、攻擊強、傾向抓牌 |
| type5 | 誠實、猜忌、上限高、攻擊弱、傾向抓牌 |
| type6 | 誠實、猜忌、上限高、攻擊強、傾向抓牌 |
| type7 | 誠實、猜忌、上限低、攻擊弱、傾向抓牌 |
| type8 | 誠實、猜忌、上限低、攻擊強、傾向抓牌 |
| type9 | 說謊、信賴、上限高、攻擊弱、傾向抓牌 |
| type10 | 說謊、信賴、上限高、攻擊強、傾向抓牌 |
| type11 | 說謊、信賴、上限低、攻擊弱、傾向抓牌 |
| type12 | 說謊、信賴、上限低、攻擊強、傾向抓牌 |
| type13 | 說謊、猜忌、上限高、攻擊弱、傾向抓牌 |
| type14 | 說謊、猜忌、上限高、攻擊強、傾向抓牌 |
| type15 | 說謊、猜忌、上限低、攻擊弱、傾向抓牌 |
| type16 | 說謊、猜忌、上限低、攻擊強、傾向抓牌 |
| type17 | 誠實、信賴、上限高、攻擊弱、傾向叫牌 |
| type18 | 誠實、信賴、上限高、攻擊強、傾向叫牌 |
| type19 | 誠實、信賴、上限低、攻擊弱、傾向叫牌 |
| type20 | 誠實、信賴、上限低、攻擊強、傾向叫牌 |
| type21 | 誠實、猜忌、上限高、攻擊弱、傾向叫牌 |
| type22 | 誠實、猜忌、上限高、攻擊強、傾向叫牌 |
| type23 | 誠實、猜忌、上限低、攻擊弱、傾向叫牌 |
| type24 | 誠實、猜忌、上限低、攻擊強、傾向叫牌 |
| type25 | 說謊、信賴、上限高、攻擊弱、傾向叫牌 |
| type26 | 說謊、信賴、上限高、攻擊強、傾向叫牌 |
| type27 | 說謊、信賴、上限低、攻擊弱、傾向叫牌 |
| type28 | 說謊、信賴、上限低、攻擊強、傾向叫牌 |
| type29 | 說謊、猜忌、上限高、攻擊弱、傾向叫牌 |
| type30 | 說謊、猜忌、上限高、攻擊強、傾向叫牌 |
| type31 | 說謊、猜忌、上限低、攻擊強、傾向叫牌 |
| type32 | 說謊、猜忌、上限低、攻擊強、傾向叫牌 |

本實驗中所使用的個人電腦處理器等級為 Intel Pentium Dual Core T3200 2.00GHz，記憶體 3GBytes，以 Java 開發程式，版本為 JDK6 update12。在主程式僅採用隨機玩法的情形下，與一種測試程式自動對局 500 局僅需約 10 秒。而主程式同時採用貝氏網路時，與一種測試程式自動對局 500 局約需 30 秒。

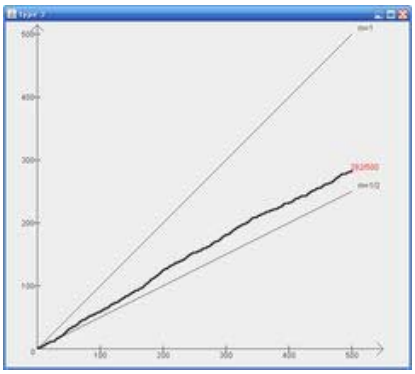
一、主程式僅採用隨機玩法



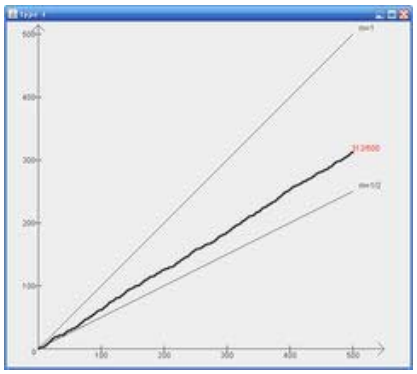
type1



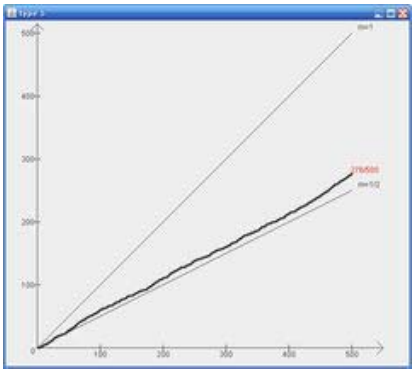
type2



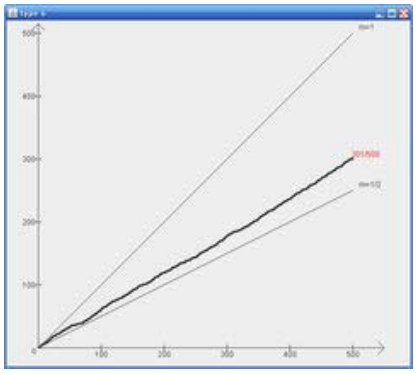
type3



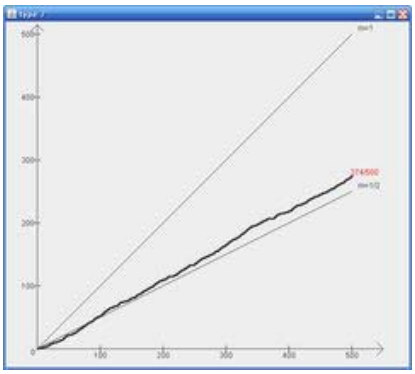
type4



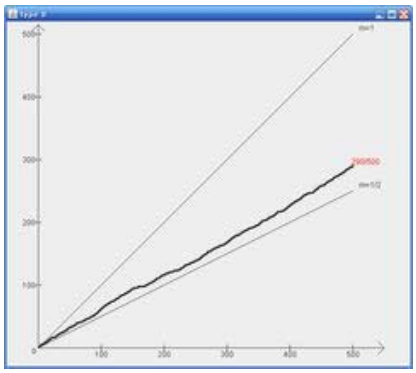
type5



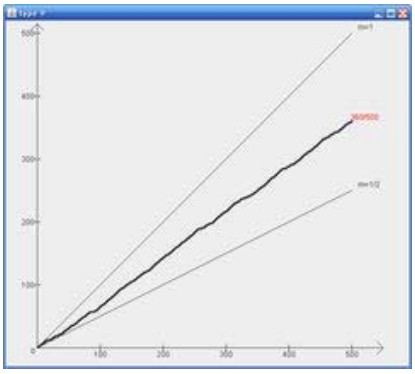
type6



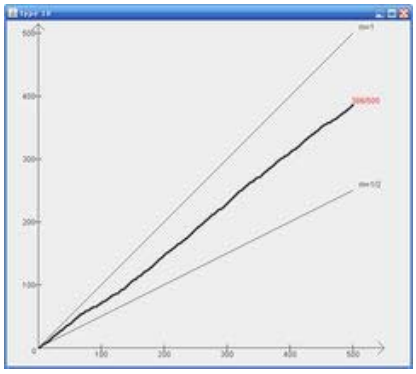
type7



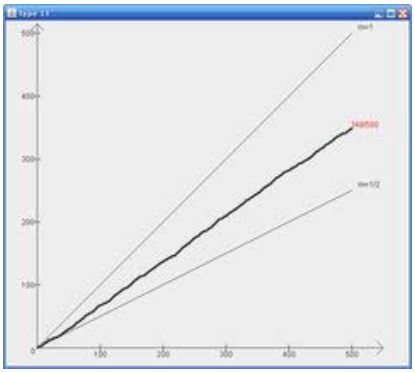
type8



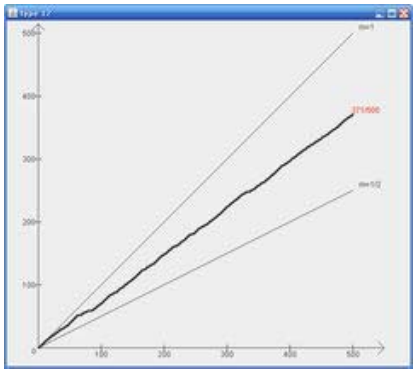
type9



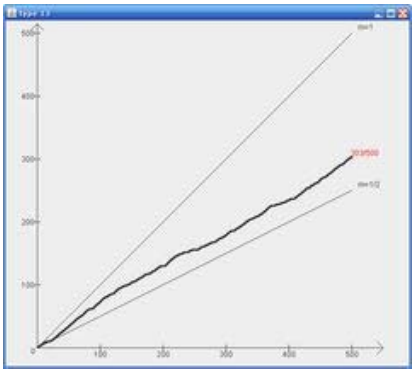
type10



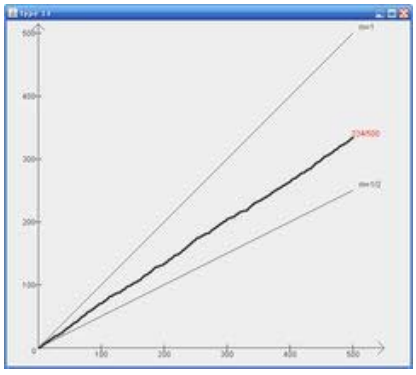
type11



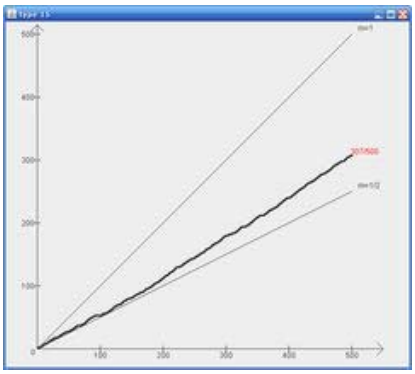
type12



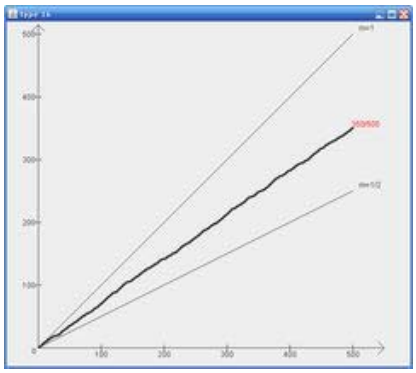
type13



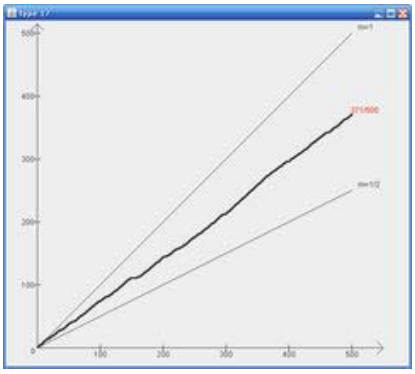
type14



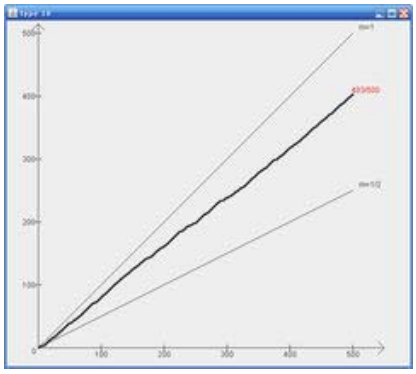
type15



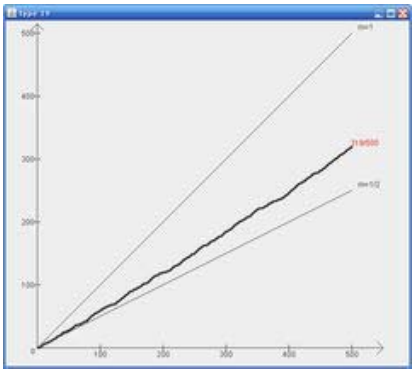
type16



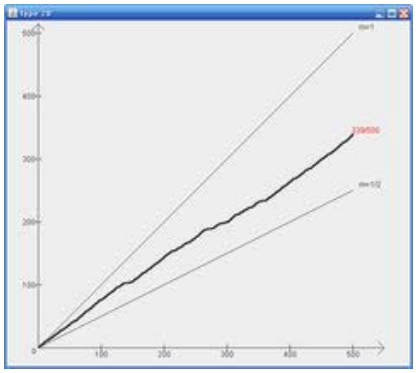
type17



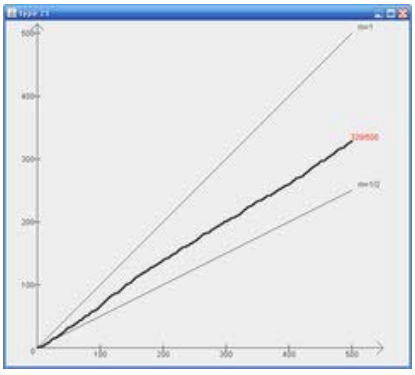
type18



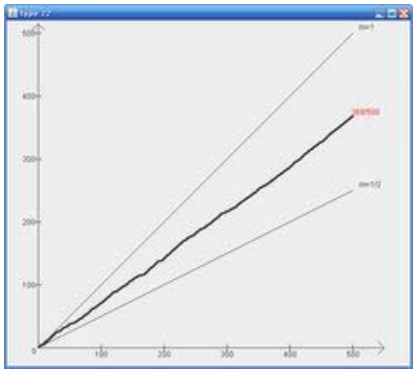
type19



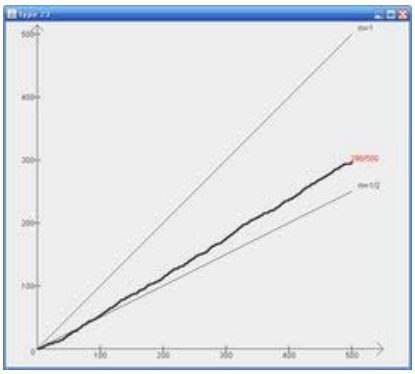
type20



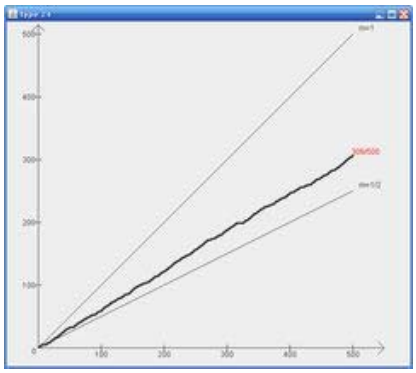
type21



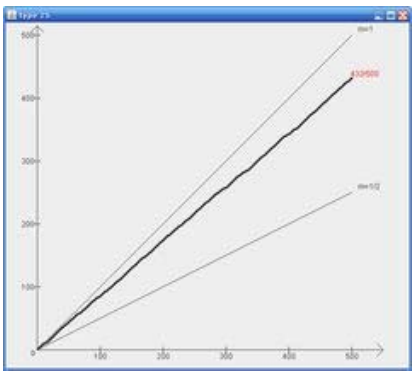
type22



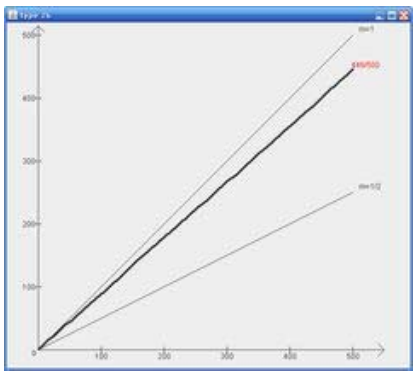
type23



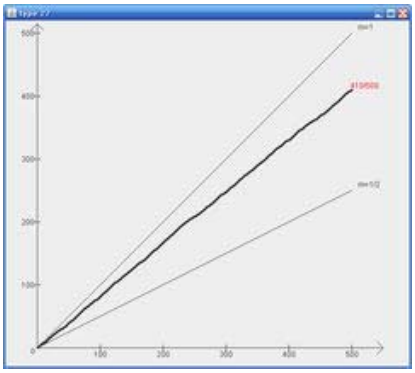
type24



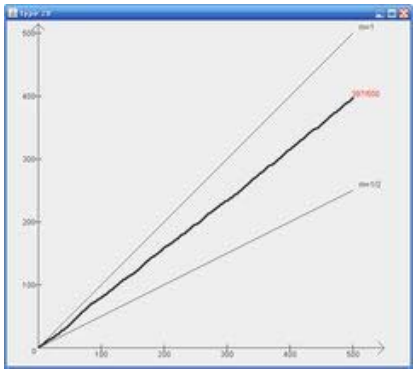
type25



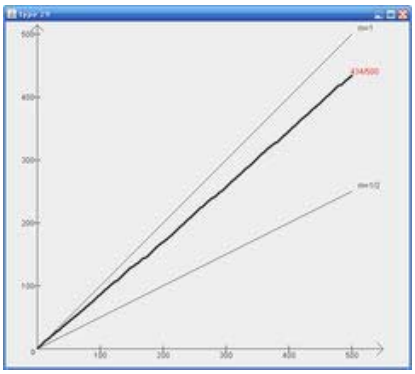
type26



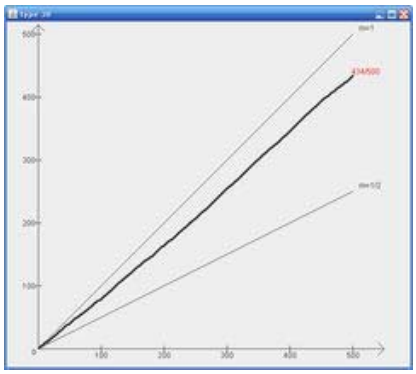
type27



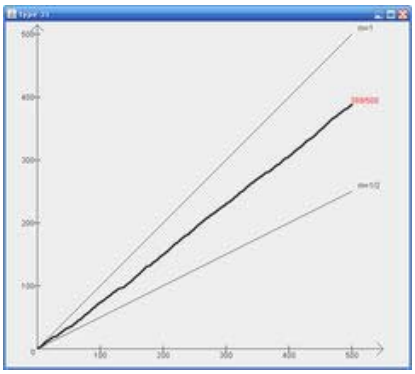
type28



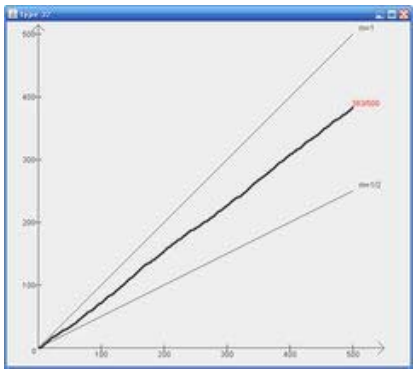
type29



type30

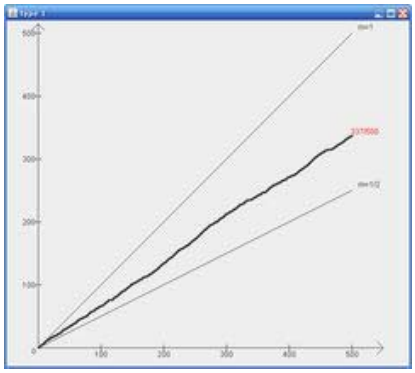


type31

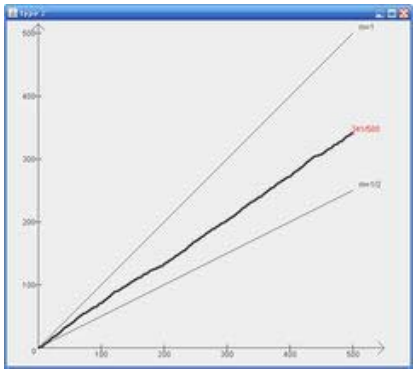


type32

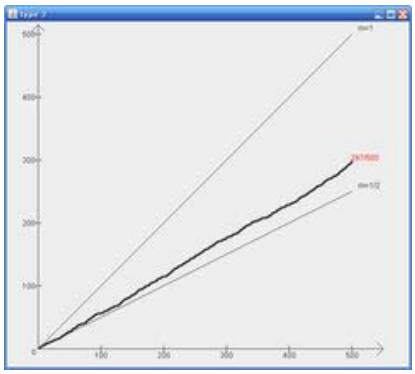
二、主程式同時採用貝氏網路



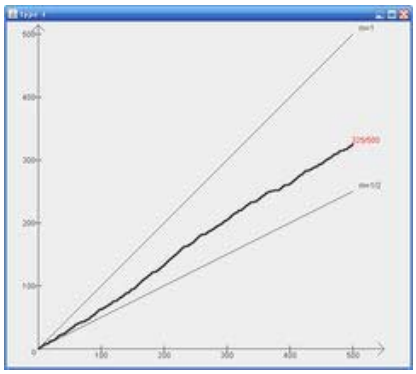
type1



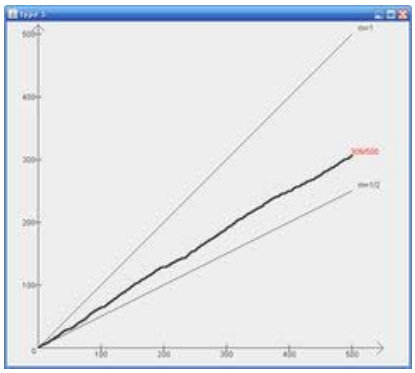
type2



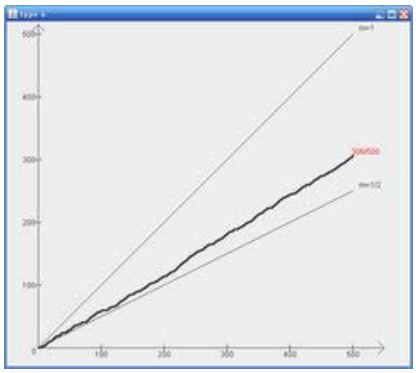
type3



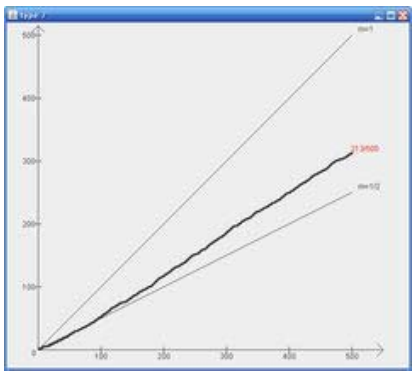
type4



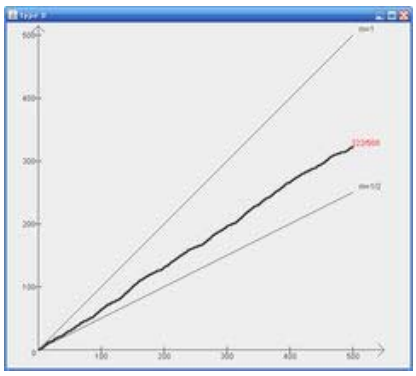
type5



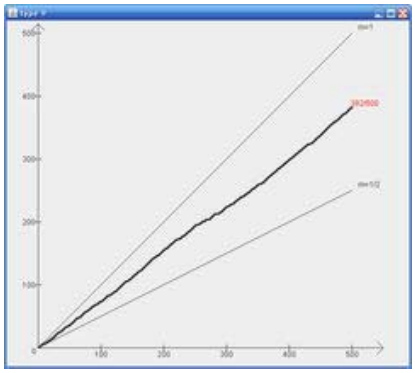
type6



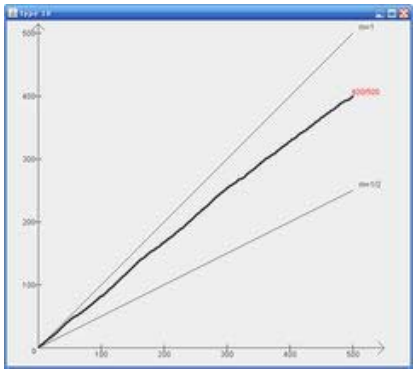
type7



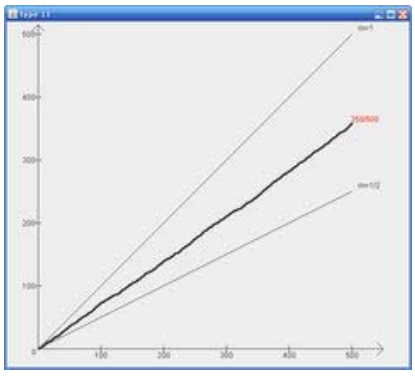
type8



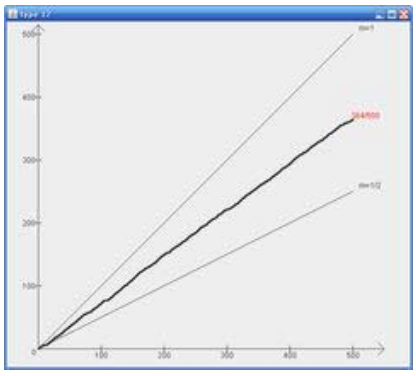
type9



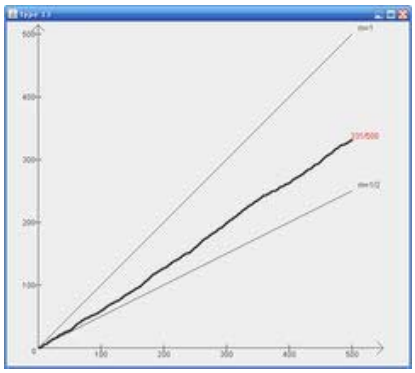
type10



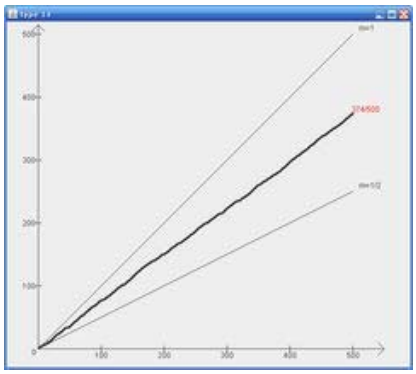
type11



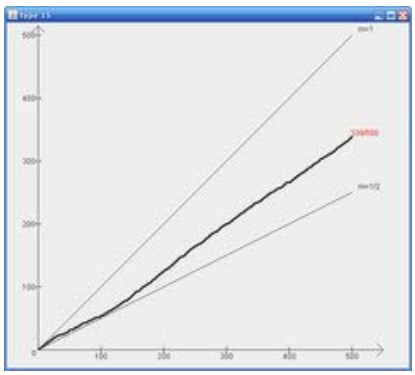
type12



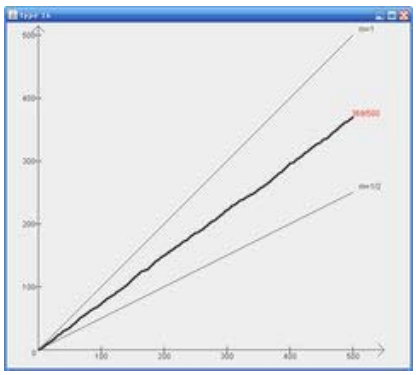
type13



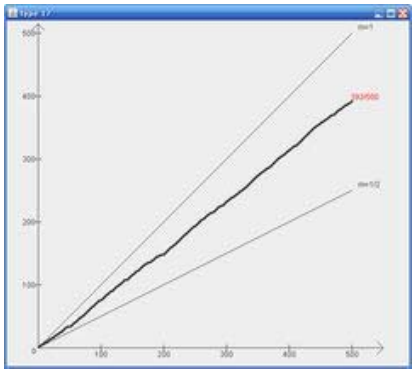
type14



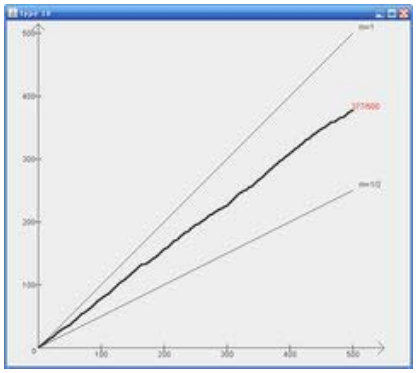
type15



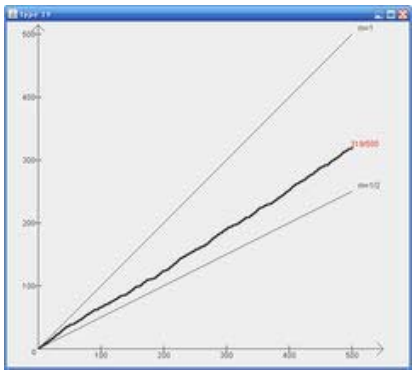
type16



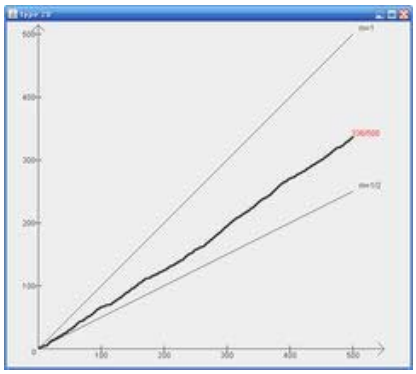
type17



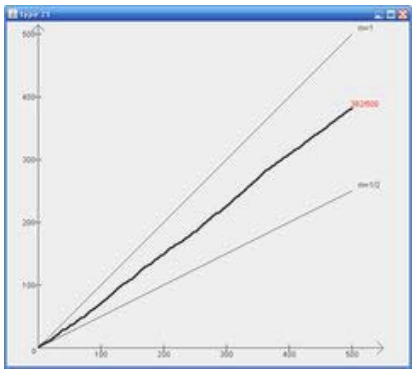
type18



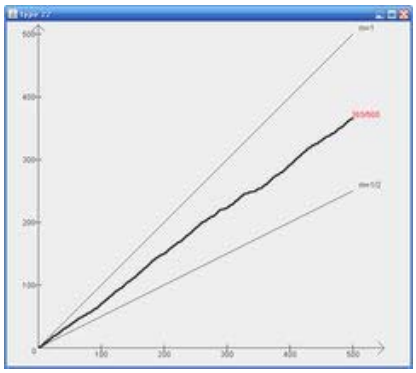
type19



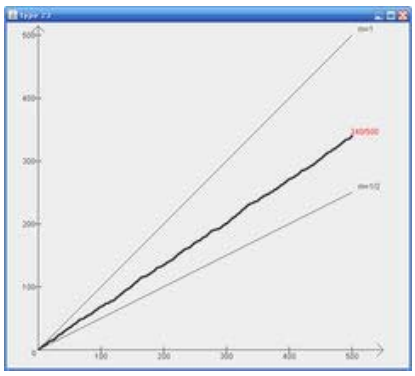
type20



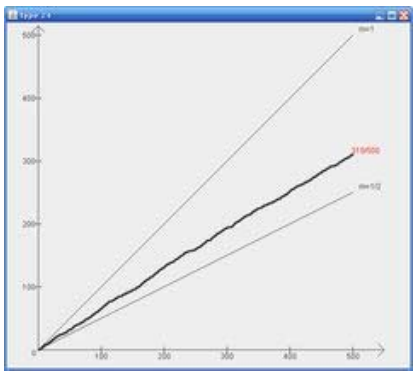
type21



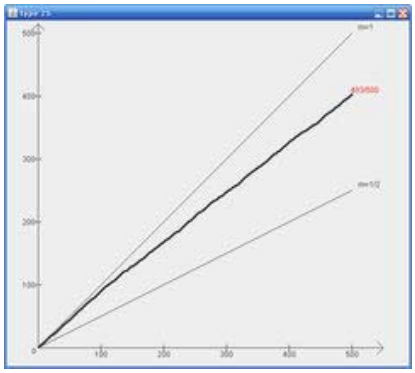
type22



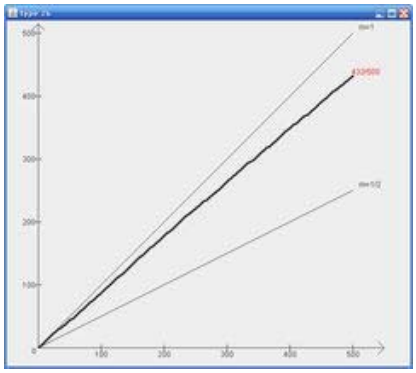
type23



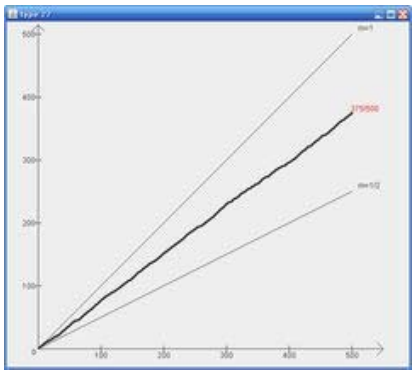
type24



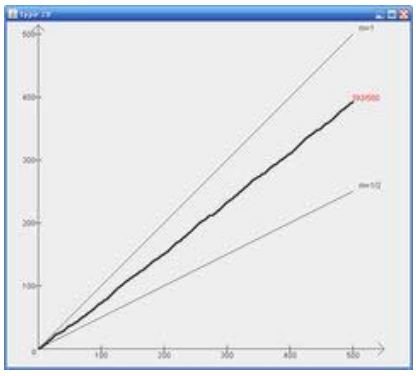
type25



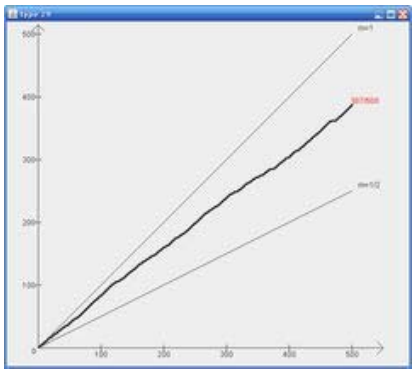
type26



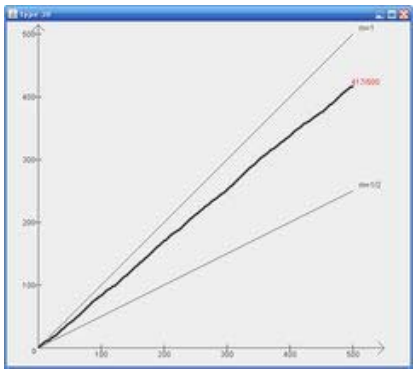
type27



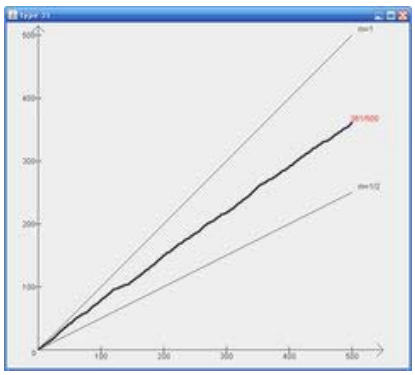
type28



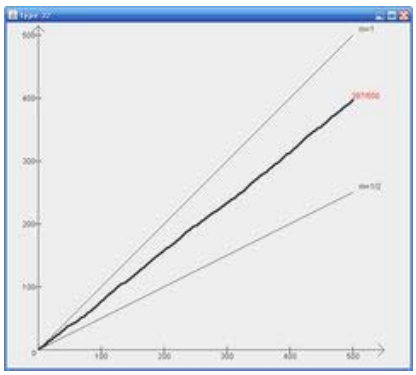
type29



type30



type31



type32

# 參考文獻

[1] G. H. Freeman, “The Tactics of Liar Dice,” Applied Statistics, Vol. 38, No. 3, pp. 507-516, 1989.

[2] J. Sum, J. Chan, “On a liar dice game - bluff,” International Conference on Machine Learning and Cybernetics Vol. 4, pp. 2179 - 2184, 2003.

[3] T. Johnson, “An evaluation of how Dynamic Programming and Game Theory are applied to Liar’s Dice,” Rhodes University, 2007.

[4] D. Egnor, “ Iocaine powder,” International Computer Games Association Journal, 23 (1), pp. 33–35, 2000.

[5] K. B. Korb, A. E. Nicholson, N. Jitnah, “Bayesian Poker,” Uncertainty in Artificial Intelligence, 1999.

[6] P. Woodward, “Yahtzee: The solution,” Chance, 16(1), pp. 10–22, 2003.

[7] T. W. Neller, I. Russell, Z. Markov, “Solving the Dice Game Pig: an introduction to dynamic programming and value iteration,” 2005.

[8] T. Hicks-Wright, E. Schkufza, “A Machine Learning Approach to Opponent Modeling in General Game Playing,” 2006.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, Introduction to Algorithms, MIT Press, Cambridge, 1990.

[10] M. Negnevitsky, Artificial Intelligence: A Guide to Intelligent Systems, Pearson, 2002.

[11] PN. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Addison-Wesley, 2005.

[12] W. Poundstone, The Prisoner's Dilemma, Doubleday, New York, 1992.

[13] D.M. Bourg, G. Seemann, AI for Game Developers, Cambridge, O’Reilly Media, Inc., 2004.

[14] 巫和懋、夏珍，賽局高手-全方位策略與應用，台北：時報出版，2002。

[15] 張振華，人生無處不賽局，台北：書泉出版，2007。