

國立交通大學

多媒體工程研究所

碩士論文

挑戰和技能的自動評估 - 以益智遊戲
"數獨" 為例



Automatic Appraisal on Challenges and Skill-Taking

Puzzle Game "Sudoku" As an Example

研究生：王郁雯

指導教授：孫春在 教授

中華民國 一 百 年 九 月

挑戰和技能的自動評估 - 以益智遊戲”數獨”為例
Automatic Appraisal of Challenges and Skill – Taking Puzzle Game
"Sudoku" As an Example

研究生：王郁雯

Student : Yu-Wen Wang

指導教授：孫春在

Advisor : Dr. Chuen-Tsai Sun

國立交通大學

多媒體工程研究所

碩士論文



Submitted to Institute of Multimedia Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

September 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年九月

挑戰和技能的自動評估 - 以益智遊戲”數獨”為例

學生：王郁雯

指導教授：孫春在 教授

國立交通大學資訊學院多媒體工程研究所

摘要

對於難度如何分類，以前很多學者提出了不少方法去做探討，大部分都是先針對如何把問題有效解決，再根據解決的方法對它的複雜度做分析，藉此對難度做評估。但是，如此一來很有可能會因為不同的問題，會有不一樣的解題技巧，進而影響解題的複雜度。再者，並不是所有電腦解題方式都是適合人類的，換句話說，也就是一般人不會用這些方式去解決問題，所以這些方法的複雜度自然就無法反映出人類所感知到的難度。

因此本研究嘗試以人的角度去構想一個人在解題過程當中會用什麼方式去解決問題。本研究以這樣的概念，對數獨解題程式做設計，量化解題過程的複雜度，並與實際人類解題速度做比較，找出評估數獨難度的方程式。利用此方程式，我們可以對每個數獨關卡計算其難度值，因此可以對多個數獨關卡作難度的排序，以及自動挑選適合難度的關卡給玩家。本研究並發現在解數獨的過程當中，從一開始至解完一題，搜尋樹中分支的不確定性、猜測次數和回溯的次數，對於人類所感知的難度有很大的影響。

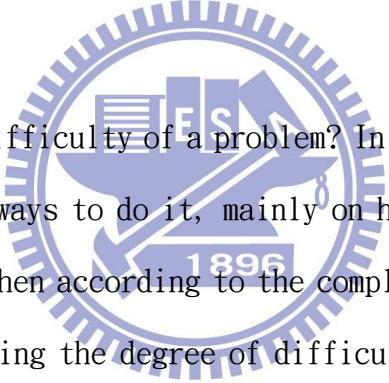
關鍵字：難度、數獨、人類解題

Automatic Appraisal of Challenges and Skill - Taking Puzzle Game
"Sudoku" As an Example

Student: Yu-Wen Wang Advisor: Dr. Chuen-Tsai Sun

Institutes of Multimedia Engineering, College of Computer Science,
National Chiao Tung University

Abstract

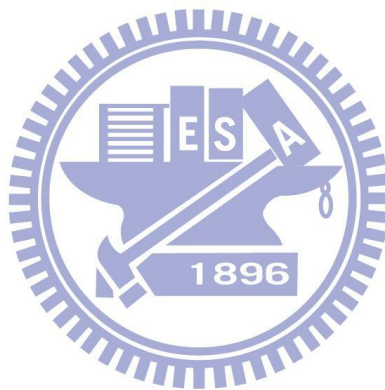


How to define the difficulty of a problem? In the past, many scholars have invented a lot of ways to do it, mainly on how to solve the problem more effectively, and then according to the complexity of the method, to provide a way of assessing the degree of difficulty. However, different problems require different problem-solving skills. Therefore, it will affect the complexity of solving problems a lot. Moreover, not every computer problem-solving approach is suitable for human, in other words, it is not necessary for the average person using these skills to solve the problem.

Therefore, this study attempts to think about a person in the problem solving process so as to define the difficulty of the problem from human perspective. Klein pointed out that ordinary people use common heuristic method to solve problems, such as means-end analysis and trial-and-error.

For such concept, we design a Sudoku AI program to quantify the complexity of AI program. Then, we compare the actual rate of human solving time to find the formula that represents the difficulty of Sudoku. In the process of solving Sudoku, we find that branch of uncertain nodes in search tree and the times of guessing and backtracking have an impact on the difficulty of Sudoku.

Key words: difficulty, Sudoku, human problem solving.



誌謝

經過這兩年多的時光，終於讓我的碩士生涯有一個完美的句點。在這過程當中，與別人有點不一樣的是我多了一點波折，但這也讓我學習到許多的事物，謝謝交大這個美麗的校園，讓我感受到另一番風味的人文氣息。

能夠完成在我人生當中的第一份著作，首先所要感謝的是孫老師，謝謝您收了我這隻迷途羔羊，並在研究過程中不斷給予教誨、叮嚀。其次，要謝謝豪哥，常常請教你有關做研究的方法，你也不厭其煩的給我們這群研究菜鳥一堆建議，相信你已經胸有成竹了，祝你能夠順利拿到博士學位！最後，要謝謝各位口試委員，由於你們的建議讓整個論文可以更加完整。

陪伴我度過這段日子的大家，謝謝你們的陪伴，讓研究路上的我不會太孤單。謝謝你們在口試前的打氣，讓我多了一份安心。還有學妹的幫忙，讓我可以更心無旁騖的準備，你們真是貼心，祝你們也可以順利的完成學業！

最後，我要謝謝我的家人，你們是最支持我的，總是站在我這邊為我說話，每次回家都可以感覺到家裡面的溫暖，吃到最熟悉的味道。爸爸、媽媽、弟弟你們辛苦了！也希望在天上的阿嬤可以放心。謝謝你們~

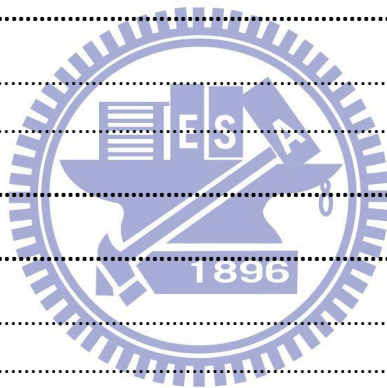
郁雯

民國百年九月予交大學習科技實驗室

目錄

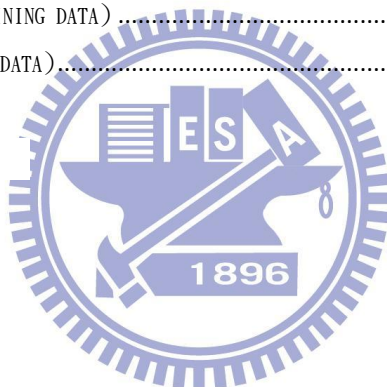
中文摘要.....	III
ABSTRACT.....	IV
誌謝.....	VI
目錄.....	VII
表目錄.....	IX
圖目錄.....	X
第一章 緒論.....	1
1.1 研究動機.....	1
1.2 研究背景.....	2
1.3 研究目的.....	9
1.4 研究問題.....	10
1.5 研究的重要性.....	11
第二章 文獻探討.....	12
2.1 益智遊戲與樂趣.....	12
2.2 認知心理學.....	20
2.3 問題求解.....	24
2.4 益智遊戲中人類解題方式.....	25
2.4.1 邏輯繪畫拼圖.....	25
2.4.2 魔術方陣.....	26
2.4.3 數獨.....	26
2.4.4 <i>Telescope Game</i>	27
2.4.5 新接龍.....	28
第三章 研究方法與設計.....	30
3.1 研究架構.....	30
3.1.1 研究步驟：.....	30
3.1.2 研究流程圖.....	31
3.2 資料蒐集.....	32
3.3 解題程式設計.....	33
3.3.1 程式架構.....	33
3.3.2 程式流程圖.....	34
3.3.3 數獨的 <i>Heuristic</i>	35

3.3.4	分支度	35
3.3.5	初始盤面狀態	35
3.3.6	Backtrack 的次數	36
3.4	分析的進行方式	37
3.4.1	參數說明	37
3.4.2	以基因演算法找出評估難度之方程式	39
第四章	結果分析	41
4.1	解題程式蒐集的數據說明	41
4.1.1	第一部分	41
4.1.2	第二部分	42
4.2	演化結果分析	43
4.2.1	第一部分 - 搜尋樹分析 (新手解題方式)	43
4.2.2	第二部分 - 解題技巧使用次數分析 (老手解題方式)	50
4.3	小結	54
第五章	結論與建議	55
5.1	結論	55
5.2	建議	56
參考文獻	57
附錄-直觀法概說	65
餘數法	65
摒餘法	70



表目錄

表格 1 H. HERNÁN MORALDO 遊戲樂趣觀點.....	17
表格 2 工作記憶.....	22
表格 3 數獨技巧 (高低).....	39
表格 4 數獨技巧 (演化參數).....	42
表格 5 方程式 1 - 因素分析.....	45
表格 6 標準差 (第一部分).....	48
表格 7 相關 (第一部分, TRAINING DATA).....	49
表格 8 相關(第一部分, TEST DATA).....	49
表格 9 標準差 (第二部分).....	52
表格 10 相關 (第二部分, TRAINING DATA).....	53
表格 11 相關(第二部分, TEST DATA).....	53



圖目錄

圖表 1 心流圖.....	2
圖表 2 HAYES 問題解決歷程流程圖.....	4
圖表 3 研究問題之架構圖.....	10
圖表 4 GENRES OF PUZZLE GAMES (KIM, 1999)	13
圖表 5 PLAYER MOTIVATION(KIM, 1999)	14
圖表 6 MODALITY (KIM, 1999)	15
圖表 7 CHRIS CRAWFORD 遊戲的互動相關性.....	16
圖表 8 信息加工系統的一般結構(車文博, 1998).....	21
圖表 9 (GATHERCOLE & BADDELEY, 1993)	22
圖表 10 邏輯繪畫拼圖說明 1.....	25
圖表 11 邏輯繪畫拼圖說明 2.....	26
圖表 12 魔術方陣例子.....	26
圖表 13 數獨解說, 摘自: 數獨大師.....	27
圖表 14 TELESCOPE GAME 解說.....	28
圖表 15 新接龍解說.....	29
圖表 16 複雜度與人類解題速度.....	30
圖表 17 研究流程圖.....	31
圖表 18 難度分類, 摘自: 尤怪之家.....	32
圖表 19 解題程式之解題流程.....	33
圖表 20 解題程式流程圖.....	34
圖表 21 猜測的解說圖.....	36
圖表 22 遊戲的難度.....	37
圖表 23 搜尋樹.....	38
圖表 24 基因型態.....	40
圖表 25 雙點配對.....	40
圖表 26 GA 適應值 (第一部分).....	47
圖表 27 難度值分布 (第一部分).....	47
圖表 28 平均難度值 (第一部分).....	48
圖表 29 GA 適應值 (第二部分).....	51
圖表 30 難度值分布 (第二部分).....	51
圖表 31 平均難度值 (第二部分).....	52

第一章 緒論

1.1 研究動機

隨著網路的盛行，各類益智遊戲也經由網路快速的傳播，但是如何讓遊戲可以歷久不衰？例如從小玩到大的井字遊戲、踩地雷、新接龍等，一直深受大眾的喜愛。其中，挑戰性是個重要的因素。當挑戰與技能達到平衡時，人就會處於心流狀態 (Moneta & Csikszentmihalyi, 1996)，專注沉浸於遊戲內並感到快樂。所以，如何界定益智遊戲的難度（挑戰）與玩家的能力（技能）相互對應關係，是令人關切的，因為「樂趣」必須是遊戲不能太難亦不能過於簡單才會存在。對於生手玩家而言，不會剛碰到一款新遊戲就覺得太難；反之，對於老練玩家來說，也可以針對自己的程度選擇他想要的難度來玩。如此一來，可以降低因為太容易而覺得無聊不想玩的意願。所以，讓一款益智遊戲包含多種難度關卡並協助玩家選擇適合的難度，可以讓這款遊戲被更多的人接受。

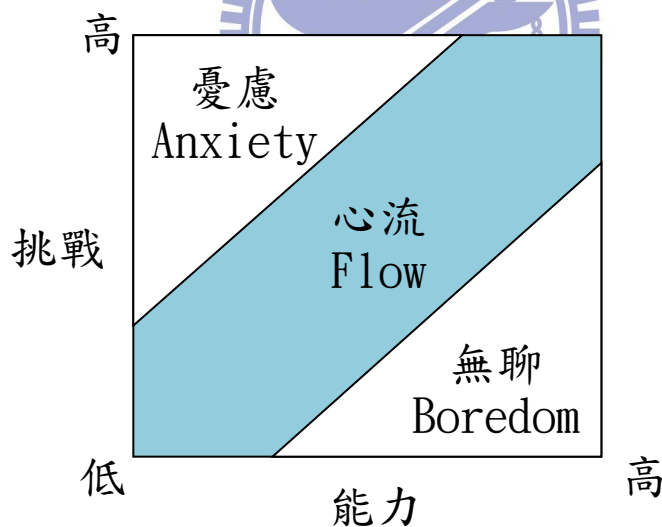
除此之外，依據遊戲的複雜度來找出對遊戲難度定義的方法，相信對於一款益智遊戲難度關卡設計會有一定的參考依據。未來在自動產生關卡的時候，也可以套用此方法來解決難度設計上的問題。

本研究企圖利用實驗方法得到遊戲複雜度與人類解題速度的高度相關性，希望可以從中得到一個合理的目標函式，使得已知當下問題的複雜度後，進一步推敲人類對於問題的難度認定，由此推論是否到達一定的上限之後，這類問題會讓人類降低去解的意願。藉由判斷這個難度適不適合設計在遊戲上，如此便可以避免設計出太過於困難的關卡導致玩家不想玩的想法產生。

1.2 研究背景

難度分析對於整體的難度平衡是必須的，必須先對難度做分析之後才能針對失衡的地方做調整，而不管是在試卷或者是遊戲設計上對於難度平衡皆有一定的需求。為何需要對難度做平衡？以心流經驗（Flow experiences）為例，當人們達到心流狀態的時候，個人精神力會完全投注在某種活動上的感覺，心流產生時同時會有高度的興奮及充實感，心理上會帶來滿足的歡愉感，並且會更加投入在目前的事物中（Csikszentmihalyi, 1975）。

在心流理論中，能力（Skill）與挑戰（Challenge）是兩個重要的因素（Moneta & Csikszentmihalyi, 1996）。其中的挑戰就如同上述的難度，當我們將難度與人類的能力清楚界定出來後，那很有可能可以間接得知目前是否處於心流狀態。



圖表 1 心流圖

然而，每一個人的能力有限，當一個人的技能無法再提升的時候，他所看的就不是像圖表1中為一個二維的圖，而是比較像一道垂直的牆，難以突破。而且，一個玩家會因為面對不同的問題而有不一樣的感覺，因此本研究嘗試要將難度做一個劃分，再讓玩家根據自己的能力去挑選適合他自己的遊戲難度，進而產生心

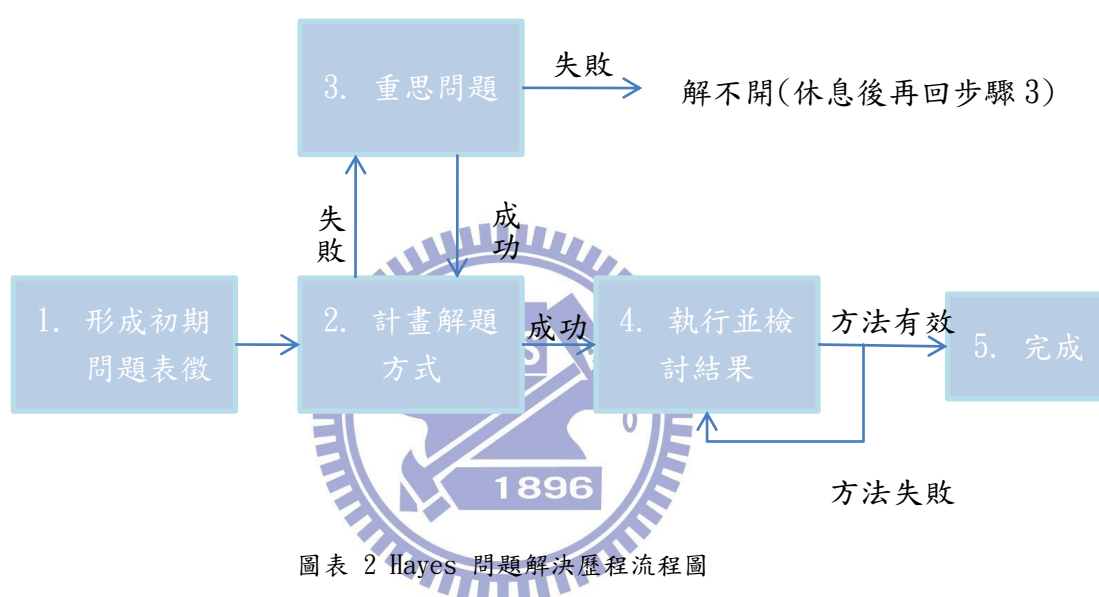
流(Flow)。

在學習過程中，除了教材的呈現方式對於學習者很重要之外，還有一項非常重要的元素，也就是測驗，每次學習到一個階段時，常常會藉由測驗的方式來衡量學習的效果，如何挑選合適的試題？大部分都是試卷上的難度係數去做選擇，針對受試者的程度給予一份適當的試題，因此，試題的難度分析也是非常受到重視的。測驗理論 (test theory) 專門來解決這方面的問題，測驗理論學者通常把它劃分成二大學派：一為古典測驗理論 (classical test theory)，主要是以真實分數模式 (true score model) 為骨幹 (DeVellis, 2006; Gullikson, 1987; Lord & Novick, 1968)；另一為當代測驗理論 (modern test theory)，主要是以試題反應理論 (item response theory) 為架構 (Hulin, Drasgow, & Parsons, 1983; Hambleton & Swaminathan, 1985; Hambleton, Swaminathan, & Rogers, 1991; Lord, 1980; Reisel, Ainsworth, & Haviland, 2005; Reisel & Waller, 2008)。

由此，我們可以了解在試題或者是遊戲設計上，皆有對難度做分析的需求。而本研究挑選「數獨 (Sudoku)」這個廣受大家瘋狂的遊戲做為本研究所要探討的遊戲難度，因之具有單人遊戲的特性以及邏輯概念很強。要對難度作分析，首先必須先了解如何去解決問題，這中間對於難度的影響牽涉到的不只是解題的步數，往往會因為所使用的解題策略或技巧的不同，造成所使用到的解題時間亦會不同。對於人工智慧領域的研究來說，如何更有效的解決問題是其優先考慮的東西，大部分的學者都將目標鎖定在找出一個更好的演算法來解決問題

(Brailsford, Potts, & Smith, 1999; Farhang, Meybodi, & Hatamlou, 2008; Kumar, 1992)。然而，人類在解決問題時，大多使用快速、直觀的判斷，而不是有意識的、一步一步的演繹。

因此，對於益智遊戲內部解題演算法的設計，本研究嘗試將人類問題解決能力套用在數獨上。藉由模擬人類解題思維，讓遊戲更進一步貼近人類的大腦，得到的解題時間可能更加貼切來描述人類對於難度的定義。我們可以將問題解決歷程分為發現問題、界定問題、尋找解決方法、付諸行動及最後的驗證結果等步驟（D’Zurilla & Goldfried, 1971）。圖 2 為 Hayes 在 1981 年提出的問題解決歷程流程圖。



由圖表2可以看到，在步驟2和步驟3之間有可能形成一個迴圈，至於會不會有無限迴圈或者說是迴圈次數的多寡，往往是根據每個人他的能力和經驗有所不同而形成差異。對於能力這方面，本研究在一開始就說明並不會針對這個來做討論。而是，從另一方面來看，對於每個人來說，他在面對不同的問題時，策略的不同會影響步驟2和步驟3所重複執行的次數，進而讓他對問題產生難或不難的感覺。因此，在解題程式設計上，本研究假設玩家具備基本的技能的前提去探討其它影響難度之因素。

在解題策略上， Klein (1996) 認為一般人在面對問題時，最常使用的問題解決策略有兩大類：演算法(algorithms methods)及捷思法(heuristics methods)。

演算法是指從起始狀態開始，有系統地檢查每一種中間狀態，直到搜尋到目標狀態為止；因為要進行一序列的心智運算，通常要花費較多的心力，所以解題時所需的時間較多且較沒有效率，但是保證可以找到解答。不過，因為演算法根據不同的問題會有不一樣的解決方式，因此牽涉到一個人的能力，所以本研究並沒有針對這方面做探討。而 Newell 和 Simon (1972) 指出捷思法是一種根據經驗而非理論所進行的選擇性作法，可以有效地縮小問題範圍，且通常可以快速而成功的解決問題，但是不保證一定可以解決問題。常見的捷思法有目的分析法、嘗試錯誤法、倒推法、差異減除法、類比法、繪圖法六種 (Klein, 1996)。

(1) 手段-目的分析法 (means-end analysis method)：在這個方法中，問題的解決是藉由重覆確認目前狀態與目標的差異，並使用一些方法來減少差異。

(2) 嘗試錯誤法 (trial-and-error method)：面對問題時，解題者不斷嘗試不同的解決問題方法，從嘗試錯誤中解決問題。

(3) 倒推法 (working backwards method)：從問題的目標狀態出發，然後返回起始狀態，進而解決問題。

(4) 差異減除法 (difference decreasing method)：解題者在面對陌生情境時，想辦法解除呈現狀態與目標狀態的差距。

(5) 類比法 (analogy method)：利用先前運用過的問題解決方法與經驗，比較新題目與舊經驗的相似處，解決一個類似的問題。

(6) 繪圖法 (diagram method)：透過流程圖呈現解決問題的步驟，可使問題更有系統的解決。

而本研究針對數獨上的解題特性，認為目的分析法和嘗試錯誤法很適合用在數獨的解題策略上。因此，將目的分析法及嘗試錯誤法此兩種人類解決問題的策略應用在本研究的解題程式當中之後，再來對解題程式的複雜度作分析。

「計算複雜度」是衡量一個問題難度和演算法好壞的工具。它常常用來比較一個新的演算法或是改良過的演算法的好壞。人們可以根據程式運算處理的過程，

將整個程序做一個複雜度的衡量。為了對難度做分類，首先，本研究將數獨的解題程式做一個複雜度的量化，把可能影響難度的因素考慮進來。

通常複雜度分為兩類：時間複雜度 (Time Complexity)、空間複雜度 (Space Complexity)。「時間複雜度」指的是要通過多少步才能解決問題，「空間複雜度」指的是在解決問題時需要多少記憶體。其中，在電腦科學和工程領域裡，以時間複雜度較為廣泛使用來衡量演算法的複雜性。

當然，除了上述衡量複雜度的方法外，還可利用回溯的次數、搜尋的深度以及廣度對整體複雜度做更精確的評估。這些東西都可以用上述計算複雜度的方法做量化。有些電腦遊戲甚至使用雜湊表 (Hash Table) 來記錄之前所走過的位子，當整個程式在跑演算法的時候，會先對雜湊表做搜尋，避免重複運算。因此，雜湊表的大小也可以視為影響整體複雜度的因素之一，所以在評估複雜度的時候亦可以將其考慮進去做為參考。

本研究討論複雜度的益智遊戲，數獨 (Sudoku) 源自於日本的一個邏輯遊戲，現在於歐洲及北美受到非常大的歡迎。1979 年，第一個數獨被創造於美國，不過那時並沒有受到注目。後來傳到日本受到人們喜愛因而於西方重新受到重視，數獨之所以受到歡迎的原因就在於它充滿著挑戰但卻有著非常簡單的規則

(Semeniuk, 2005)。直到現在，大部分研究主要都是在探討如何去解決數獨，基於不同的知識利用一些特別的演算法。數獨已經被證明是一個「非確定性多項式時間完全問題 (Non-deterministic Polynomial-time Complete Problem, NP-Complete)」(Yato & Seta, 2003)。如何去解決一個 NP-Complete 問題呢？部分學者將數獨看作滿足問題 (Satisfaction Problem, SP)，並用通用 SAT 求解器 (general SAT solver) 去找出答案 (Lynce & Ouaknine, 2006; Reeson, Huang, Bayer, & Choueiry, 2007 ; Weber, 2005)。也有學者將數獨視為限制滿足問題 (Constrain Satisfaction Problem, CSP)，並比較不同的 propagation schemes

對於解數獨的差異 (Reeson, Huang, Bayer, & Choueiry, 2007; Simonis, 2005)。

然而，在數獨的難度定義上並沒有一套標準。在報紙或是書上那些已經出版的數獨大部分都已經將難度等級劃分好，方便讓玩家可以挑選所需要的等級去作挑戰，但玩家並不清楚那些等級是依據什麼來劃分的，每位學者都有自己主張的難度劃分方式。Inkala (2007) 以圖形架構 (graph structure) 和資訊理論 (information theory) 為基礎提出一種定量機制衡量數獨的複雜度。Mantere 和 Koljonen (2006) 則是利用基因演算法去測試數獨的難度等級。Li、Song 與 Zhang (2010) 根據 Shannon (1948) 提出的定義熵 (entropy) 的概念建構一個數學模型來衡量數獨難度。從這些學者的研究當中，可以發現到幾乎都是從工程上面的角度來解決問題，甚少考慮到人認知上的角度。

對於人和電腦來說，電腦可以記住所有的解題過程但是人卻會因為能力上面的差別所記憶的東西多寡也會有所不同。在人工智慧的領域上，搜尋 (Search) 是一種常見問題解決的機制，用搜尋演算法來解決的問題大致上分為單一路徑問題 (single-agent pathfinding problems)、兩人遊戲 (two-player games)、限制滿足問題 (constraint-satisfaction problem) 三類 (Korf, 1996)。我們可以將解決問題的過程描述為狀態空間之搜尋，狀態空間中的每一個狀態即代表相對應的解題狀況。搜尋演算法有非常多種，但一般往往會有嘗試錯誤的探索概念在其中，暴力搜尋法是最普遍的搜尋演算法，在搜尋的過程中，除了能夠分辨目標狀態和非目標狀態之外，沒有其他參考資訊，常見的技巧有：廣度優先搜尋 (breadth-first search)、統一成本 (uniform-cost search)、深度優先 (depth-first search)、迭代加深 DFS (iterative deepening DFS) 及雙向搜尋 (bidirectional search)；而啟發式搜尋法則是在搜尋的過程中，可以藉由啟發函數估計目前狀態和目標狀態還有多少距離，進而增進搜尋的效率，如：

Newell、Shaw 與 Simon 在 1958 年提出 Alpha-Beta 切捨演算法常常被用於棋類對弈遊戲之中 (Knuth & Moore, 1975; Newell, Shaw, & Simon, 1958)、A* 常用於尋找最佳解路徑問題 (Hart, Nilsson, & Raphael, 1968)、迭代加深 A*(iterative-deepening A*) 解決了廣度優先搜尋時的空間問題 (Korf, 1985)。另外，有一類的通用啟發式策略稱為萬用啟發式演算法 (metaheuristic)，通常使用亂數搜尋技巧，他們可以應用在非常廣泛的問題上，如：使用 Kirkpatrick、Gelatt 與 Vecchi (1983) 提出的模擬退火 (simulated annealing) 對數獨解題 (Lewis, 2007)、使用蟻群演算法 (ant colony optimization algorithm) 在圖中尋找優化路徑 (Dorigo, 1992)、修改最陡上升爬山演算法 (hill-climbing algorithm) 降低數獨的搜尋空間 (Jones, Roach, & Perkins, 2008) 及禁忌搜索 (tabu search) (Glover, 1990) 等等。啟發式搜尋法的最大缺點是不能保證成功，但可大量節省時間是其優點，同時針對問題設計出更合適的啟發函數 (Korf, 1996)。然而，啟發法則的執行效率是無法分析的，例如在對奕程式中使用啟發法則，我們只能藉由實驗觀察評估其效率，也許在贏了一百回合之後才發現其缺點。此外，並非每一種問題都適合啟發式搜尋法。例如：某些複雜的問題可以分解成幾個小問題再分別解決，問題狀態的改變只占原狀態的一小部份，或問題本身在執行上無法回溯等。這些不適用啟發式搜尋法的問題，則可以改用規劃，規則庫，專家系統，學習等其他方法。

對於數獨到目前為止已經有為數不少的研究在討論它或是拿它當作實驗對象，大部分的學者主要還是在於開發出一個演算法來解題；有的可能是把數獨轉換成某一類問題然後用解決這類的問題去解題；有的是利用電腦的優勢 (運算快和記憶體大) 將數獨用暴力法解出來；不過就最近的研究趨勢而言，大家傾向於使用一些啟發式演算法來解題 (Pelanek, 2011; Pang, Li, Song, & Zhang, 2010; Xu, C., & Xu, W., 2009)。本研究基於這些學者的研究基礎上，從人工智慧的角度，以搜尋演算法當作本研究數獨解題程式設計的主要架構，假設此解題程式

在能夠記住先前的步數，但對於要從目前的狀態走到下一步狀態時，使用的啟發式搜尋的概念，運用不同的人類解題策略（如目的分析法、嘗試錯誤法），評估哪個節點是最有可能被選擇的，藉此模擬人類解題方式，並從中計算解題程式的複雜度並產生難度值，將其與實際數據（人類解題時間）做比較，期望得到一個客觀的難度分類依據。

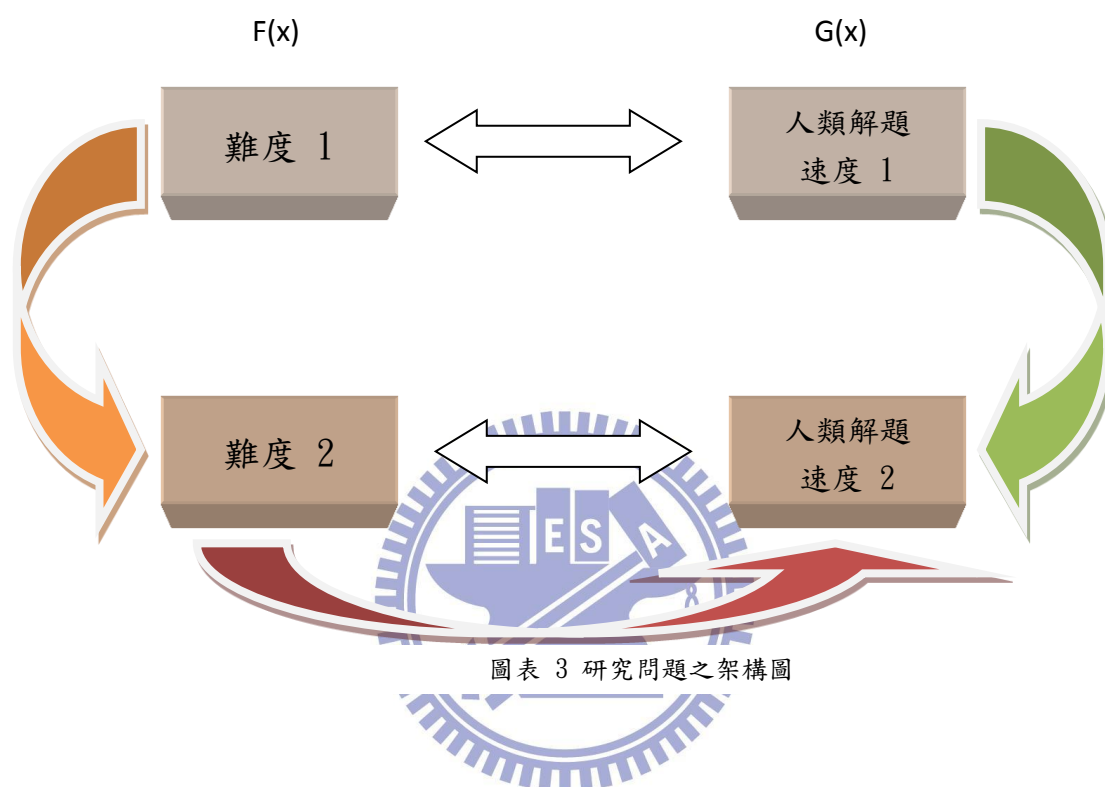
1.3 研究目的

綜觀目前網路上的益智遊戲，大部分雖然都有關卡的設計，但對於難度的分類其實並沒有一個好的方法來區別，導致玩家在挑選遊戲的時候，常常不清楚這款遊戲的難度到底適不適合自己，因此很可能因為一開始挑錯關卡，讓自己害怕、討厭或不想玩這款遊戲。

所以，本研究目的在於期望可以找到一套方法來清楚定義遊戲關卡難度的分類，而這個方法是根據與人類解題速度做比較找出來的。另外，可以藉由複雜度來判斷難度的方法，未來在設計關卡難度的時候，可以利用不同的求解演算法來設計益智遊戲解題的 AI，直接由演算法來界定遊戲的難度，提供一個更加客觀的評斷方式。

1.4 研究問題

本研究依據研究目的，提出下列架構圖及研究問題：



研究問題

- 1) 找到代表解題程式之複雜度變化的函式 $F(x)$ 。
- 2) 找到代表人類解題速度變化的函式 $G(x)$ 。
- 3) 找到 $F(x)$ 與 $G(x)$ 之間的關係。也就是說，找到一個衡量複雜程度的方式，使此複雜度跟人類解題速度成正相關。

1.5 研究的重要性

本研究最主要的貢獻在於模擬人類的解題模式讓電腦更加貼近人類想法。以往，在人工智慧這領域上，大部分還是傾向於先把問題解出來再說，如何解的更快更精準才是最後的目標，甚少有人把焦點放在先以人類解決問題的觀點上面，考慮到人類的極限是什麼？大腦運轉的速度是否能夠與電腦的CPU相同？還有解決問題的時間，多久會讓人想放棄？等等許多人類在解決問題上面會面臨到的。

最為重要的是，直接由問題的複雜度推測出人類對於關卡的難度認知。憑藉著這個結果，我們可以直接定義問題的難度，讓玩家自行選擇適合自己的遊戲關卡，以及自動推薦漸進難度的關卡給玩家，避免玩家持續玩到過於簡單的關卡而覺得無聊、或是在剛接觸到某個遊戲時就碰到難度太高的關卡而喪失興趣。

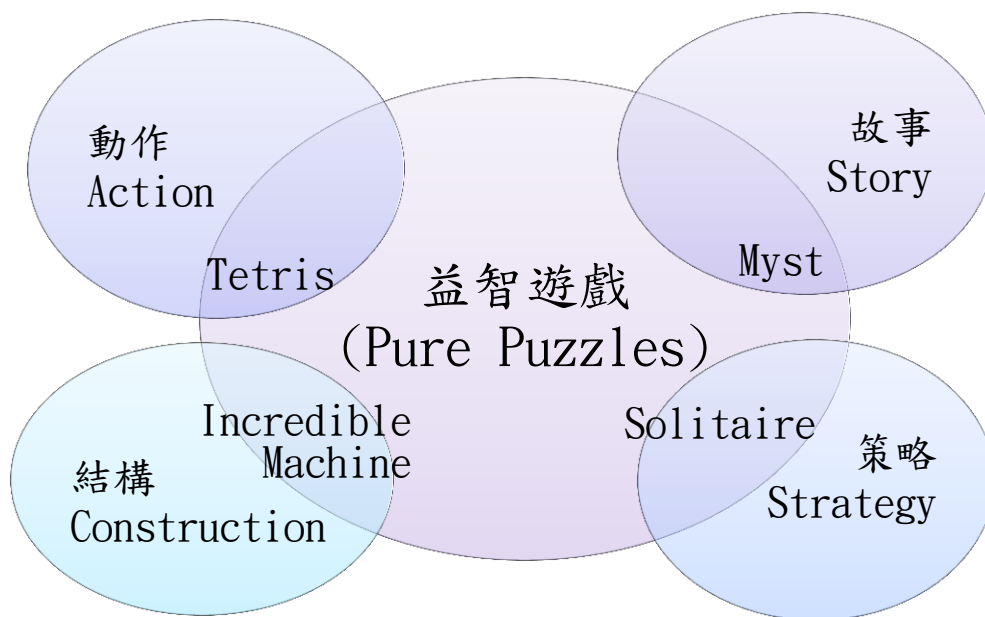


第二章 文獻探討

為了達到研究目的，本研究以數獨當作實驗對象，探討如何藉由難度讓人處於心流狀態。因此，本章節深入討論益智遊戲的特性、遊戲如何帶給玩家的樂趣、在人類認知的思維上如何解決問題、數獨或新接龍這類益智遊戲中有哪些人們常用的 Heuristics…等等。

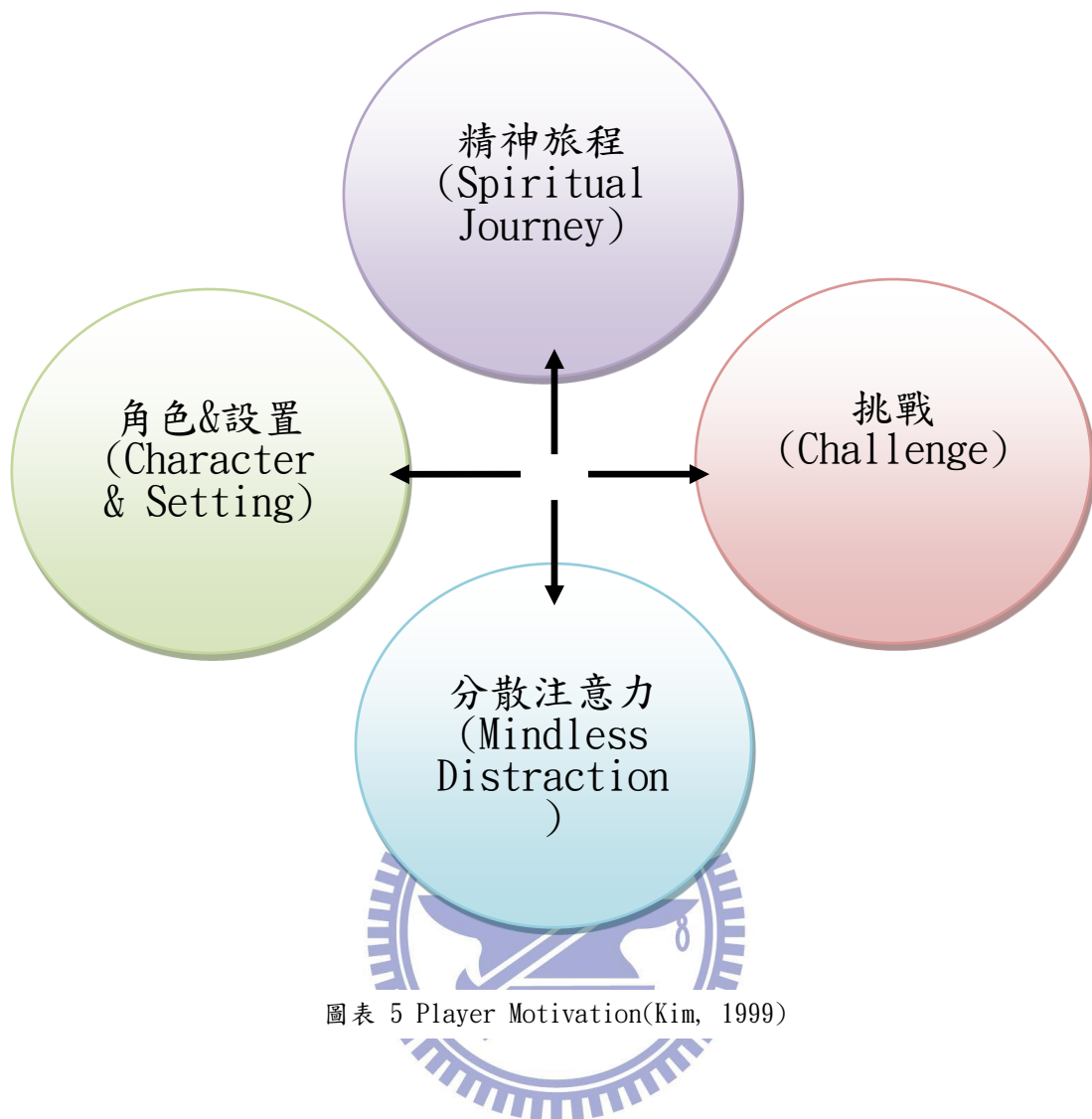
2.1 益智遊戲與樂趣

益智遊戲是以挑戰智力為主要樂趣的一類遊戲，其中的考驗涉及到記憶力、邏輯思考、策略、模式辨別、空間想像、文字等多種思維，並附加以運氣和反應等元素。由於這類遊戲鮮有暴力性內容，因此多被視作休閒類遊戲，Kim(1999)將之分為動作、故事、策略和結構幾種類型（如圖 4），有些 Puzzle 會混合上述幾種類型；有些則是純粹的益智遊戲（pure puzzle）僅僅只需要玩家的邏輯思考能力，而不牽涉到反應或手眼協調等等的能力，本研究所探討的數獨就是屬於這一類的益智遊戲。



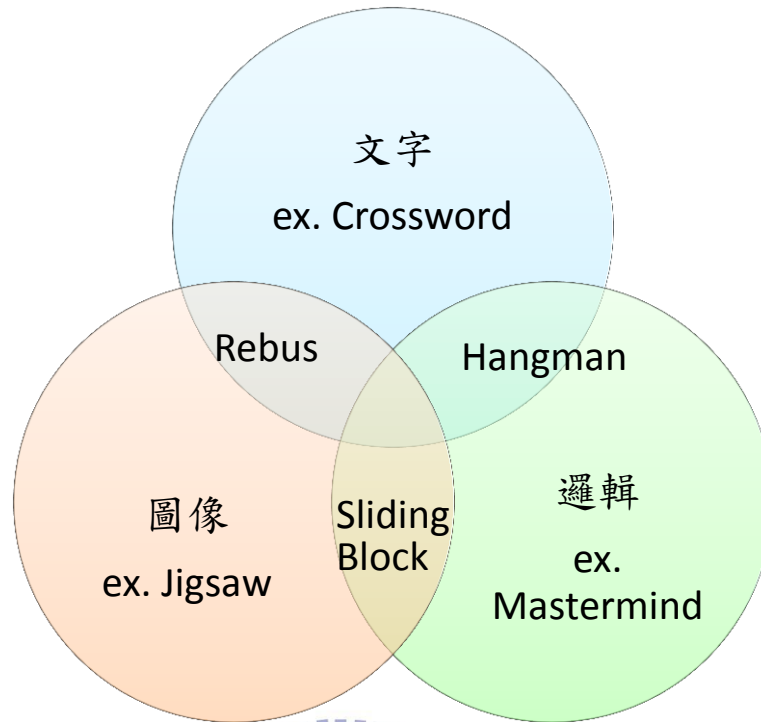
圖表 4 Genres of Puzzle Games (Kim, 1999)

不同的人玩益智遊戲其實都有不同的動機 (Motivation)。一些人將益智遊戲視為一種精神上的旅程，可以體驗不同的人生；有些人想要追求刺激、挑戰，將找答案的過程當作一種挑戰；玩俄羅斯方塊的人或許是想讓自己的腦袋休息一下，不需要花腦筋去想事情。有些人或許根本沒有甚麼理由，僅僅只是因為它有著自己熟悉的元素而去玩玩看。也就是因為這些動機，人們選擇自己覺得有意思的遊戲去玩 (Kim, 1999)。數獨吸引人的地方在於它有著不同的難度，挑戰著一個人的思考邏輯與策略，本研究試圖將數獨的難度做完全排序，讓玩家能根據自己的能力自由選擇各種等級的挑戰。



圖表 5 Player Motivation(Kim, 1999)

一般而言，人們在選擇益智遊戲的喜好有三種類型：文字 (Word)、圖像 (Image)、邏輯 (Logic) 三種益智遊戲型態 (圖 6)。每一種類型都有其適合的思考模式去解題，當然有些益智遊戲不只包含一種類型在其中，例如：Hangman 就是一種文字邏輯遊戲。而本研究所實驗的益智遊戲數獨是一個單純的邏輯遊戲，牽涉到的玩家能力僅有思考邏輯這塊。



圖表 6 Modality (Kim, 1999)

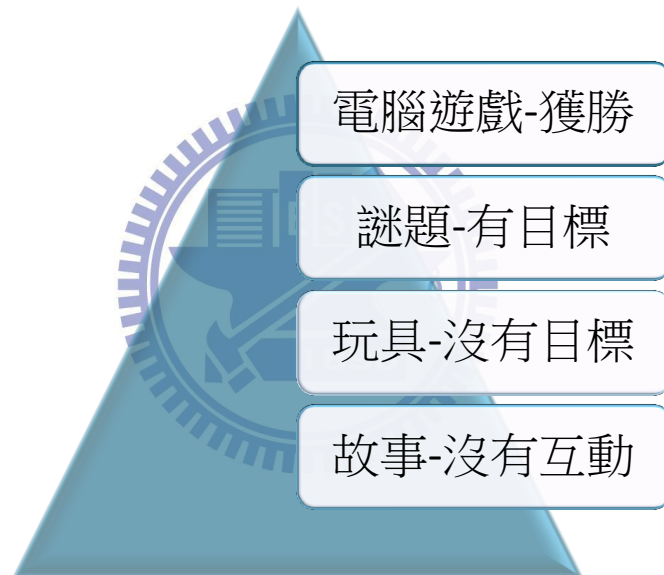
益智遊戲是有樂趣的而且它是有解答的 (Kim, 2006)。如何令一款益智遊戲變的有趣呢? Kim 提出下列幾點：

1. 新奇的 (Novel)：將現實生活中不合乎實際情況的事物，付諸於遊戲中實驗它，令玩家感受到前所未有的新鮮感，覺得這個遊戲很新穎。
2. 不能太簡單，也不能太難：關卡太簡單就會覺得無聊、提不起勁，太難會喪失信心、覺得氣餒。
3. 難以捉摸的 (Tricky)：在解題時，必須要換一個角度去想問題，問題才可能會解出來。也就是感知轉移 (Perceptual Shift)。

然而，有趣這件事是個很主觀的東西，取決於每個人的口味，有的人覺得踩地雷很好玩，但有的人卻不以為然。在遊戲設計上，樂趣是許多遊戲設計者所關心的議題，但是，在沒有一套既定標準之下，只能憑藉以往設計遊戲的經驗，遊戲設計者設計出自認為玩家會覺得有趣的遊戲，而這真的是玩家想玩的遊戲嗎？

樂趣是一個心理狀態的表現，對於每個人來說，都希望可以獲得樂趣，無論是在遊戲或工作上。本研究希望盡可能擺脫個人觀感，用更客觀的方式去衡量可能影響樂趣的遊戲難度。

Crawford(1984)在電腦遊戲設計的藝術(The Art of Computer Game Design)一書中提到由遊戲的互動相關性，由高到低，區分為電腦遊戲、謎題、玩具和故事四種類型。其中，謎題是以自我挑戰為主，因此本研究選擇數獨當作研究的對象，嘗試將關卡做完全排序推薦給玩家做選擇。



圖表 7 Chris Crawford 遊戲的互動相關性

- 1) 電腦遊戲 (Game): 是凌駕於三者之上，更有系統化的把其他三種串連在一起。遊戲是利用技術將謎題轉換成電腦遊戲(益智遊戲)，具有互動性，因此讓玩家有比較的感覺，傾向於想要獲得勝利。
- 2) 謎題 (Puzzle): 與遊戲不同的地方在於少了比較的感覺，所以傾向於對自己的挑戰，針對題目找到答案。
- 3) 玩具 (Toy): 對玩家來說是最具有彈性的，玩家可以任意操縱改變規則。
- 4) 故事 (Story): 有著強烈的連續性，不允許玩家任意更動，沒有互動性。

想要設計出一個好的益智遊戲 (Puzzle)，要先建立一個好的玩具 (Kim, 2006)。在玩家找到解答之前，必須要讓他覺得操控它是件有趣的事情。如果一個玩家能夠持續被遊戲吸引，持續感受樂趣，那麼遊戲就是成功的。Moraldo(2001) 提出創新(creativity)、驚喜(surprise)、理解(understanding)、力量(power)、挑戰(challenge)和成長(growth)等幾項影響遊戲樂趣的因素。

表格 1 H. Hernán Moraldo 遊戲樂趣觀點

創 新	我們的注意力總是被新奇和有趣所吸引。善用新點子，把東西做得看起來不同，操作起來不同等等。
驚 喜	讓玩家感到驚喜：與創新有點類似。讓遊戲存在著一些想不到的事物（但必須是玩家可以接受的範圍內）。
理 解	確保玩家可以完成遊戲：沒有什麼比希望繼續遊戲但是又無法明白應該怎麼做而更加帶來挫折感了。盡可能幫玩家理解遊戲機制。
力 量	人們喜歡力量，但必須適當的給予力量。太多的力量會反而會讓玩家很快地覺得無聊；同樣地，太少的力量會讓玩家覺得無助。不要讓玩家覺得在遊戲中，自己像個嬰兒一般。
挑 戰	當遇到挑戰時，玩家一開始想的就是要克服這個挑戰。這是很好的方式帶領玩家進入遊戲，然後，當玩家挑戰成功，他們會覺得完成了一些東西。藉由這種回饋 (rewarding experience)，體會到勝利的滋味，讓玩家更想進一步去玩這個遊戲。
成 長	讓遊戲成長（如：關卡設計），讓玩家體驗成長（如：等級提升），讓玩家的屬性成長。當玩家感受到成長的時候，無論是屬性還是遊戲情節，玩家就不會感到厭倦和無聊。

綜合來說，遊戲需要面對玩家。即便一個設計者所想出來的是一個很大的遊戲機制，但是有可能只符合他自己的胃口，其他人卻不喜歡。一款遊戲的樂趣點，需要以玩家的偏好來引導。樂趣應當盡可能讓更多的玩家認同，這是一個最大的挑戰，當然，因為一款遊戲有自己特定的類型，表現和創意等等，所有方面都只可能覆蓋特定人群而不是所有人群，有些人必然討厭。所以，在樂趣點上找到一個最佳的平衡是一個值得嘗試的挑戰。

遊戲也可以被視為是一種藝術，遊戲設計的過程，必須有讓玩家巧遇一個有意義的事件 (Salen & Zimmerman, 2003)。簡單來說就是必須要有意義的玩 (meaningful play)，這也就是在遊戲設計上的美學，如何提供一個讓玩家認為有意義的遊戲經驗，或者說是遊戲能夠玩得多有趣 (Browne, 2008)。

美感 (Aesthetic feeling) 是一種很主觀的東西，許多學者辯說美其實就像是一種對智慧的追求一樣，是感性的 (Browne, 2008)。每個人感受到的美一定不會完全一樣，但或許會有相似的地方。Gardner (2004) 形容數學家喜好抽象的美，在數學科學上，有次序 (order)、勻稱 (symmetry)、定義的限制 (definite limitation) 等等這些主要構成數學之美的特點，吸引著數學科學領域的學者之注意力。

美學被關心的問題主要有現有的藝術作品如何被描述、解釋、演化及如何創造新的藝術作品 (Stiny & Gips, 1978)。Birkhoff (1993) 提出了一些關於審美辦法 (aesthetic measure) 的概念，可以將美做量化。

$$M = f\left(\frac{O}{C}\right)$$

O代表物件基於內在品質的次序，如它的簡單 (simplicity)、對稱

(symmetry)、平衡(balance)與和諧(harmony)；C代表複雜度，如元件的數量和層次上的細節可能影響人們的感受；M代表美感的指標。

對於遊戲的美，究竟能帶給玩家多大的樂趣，這是許多遊戲設計者在設計遊戲的過程中必須要考慮的。因此，針對遊戲品質的衡量目前有不少的研究。

Thompson (2000) 提出四項主要指標：深度(depth)、明確性(Clarity)、戲劇性(Drama)、決定性(Decisiveness)。(1) 深度：讓玩家持續的學習獲得更高的技巧；(2) 明確性：清楚的遊戲規則讓玩家可以充分發揮自己的遊戲策略。深度和明確性是兩個相當接近的概念，深度會因為所利用到的遊戲策略而有所不同(Abbott, 1975)。(3) 戲劇性：弱勢的玩家有機會逆轉局勢並獲得勝利(Schmittberger, 1992)。Thompson (2000) 也指出不需要在一擊KO(single killer move)後出現，應該在延長賽(extended campaign)的時候發生比較適當；(3) 決定性：當結果已經確定了，玩家無法力挽狂瀾時，遊戲就應該迅速的結束掉。劇情和決定性兩者是互補的關係(Browne, 2008)，需要清楚了解玩家的感受才能夠判斷設計的時間點。

遊戲設計大師Kramer (2000) 認為：創意的新鮮度和可玩性、驚喜、公平的機會、獲勝的機會、不受Kingmaker影響(Kingmaker是遊戲裡其中一種玩家，他沒有勝利的希望，但卻有著決定誰是贏家的關鍵)、不能太早淘汰、合理的等待時間、具創意的控制、一致性、元件的數量、目標群體和一致的規則、有張力的學習和掌握比賽、複雜性和影響力等等，都可以用來當作評估一個遊戲的好壞之依據。Althofer (2003) 提出在單人遊戲上衡量有趣性的方法：(1) 遊戲長度(Game length)：遊戲不應該太短或太長，太短會覺得沒玩到甚麼就結束，讓人覺得失望；太長會讓人感到厭煩；(2) 和局的次數(Drawing quota)：遊戲不應該出現太多的合局，分不出勝負的遊戲往往會讓人覺得沒勁；(3) 平衡(Balance/advantage)：遊戲不應該偏袒某些玩家，遊戲必須要是公平的；(4)

變異 (Variability)：一些變異在玩的過程中是被接受的，例如在尋寶遊戲中，可能不知道會獲得什麼寶物而充滿期待；(5) 變異會導致性能損失 (Performance loss due to variability)：太多變異會導致玩起來的感覺很隨機，沒有漸進式的感覺，甚至不清楚自己在玩些什麼，如何玩；(6) 深度優先 (Deepening Positivity)：深度搜尋優於淺度搜尋 (shallower searches)；(7) 平滑度 (Smoothness)：每一次的搜尋彼此間的差異。Rolle (2003) 將這些概念應用在一般遊戲玩家 “Morphling” 去幫助遊戲設計過程衡量有趣性。

樂趣是一種非常主觀的感覺，然而對遊戲設計者來說讓自己設計的遊戲充斥著有趣性是非常重要的。正因為如此，本研究所要做的就是從挑戰這個面向去讓玩家感受到益智遊戲的樂趣，先從難度的判定，找到一般人普遍都認同的方式，讓遊戲關卡以漸進式難度的方式呈現給玩家，藉此讓玩家依據自己的能力在遊戲中選擇適合自己的關卡，達到挑戰與技能的平衡。進一步的，找出激發出玩家覺得樂趣的點。

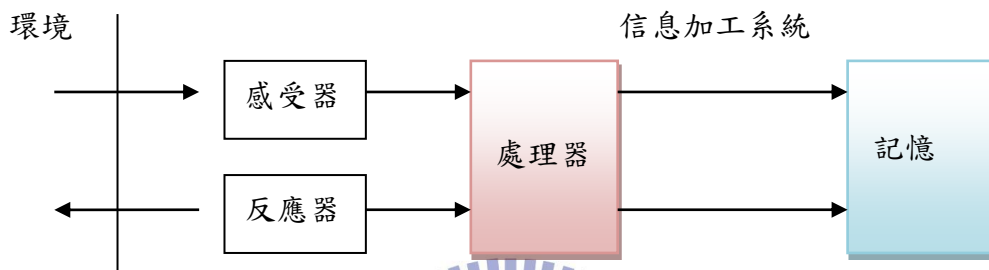


2.2 認知心理學

認知心理學是一門研究認知及行為背後之心智處理（包括思維、決定、推理和一些動機和情感的程度）的心理科學。這門科學包括了廣泛的研究領域，目的在於研究記憶、注意、感知、知識表徵、推理、創造力，及問題解決的運作。即把大腦當作資訊處理裝置的觀點。

Craik (1943) 指出基於知識的代理人三個關鍵步驟：(1) 刺激必須翻譯成知識內部的表示。(2) 認知過程藉由處理既有的表示來得到新的內部知識表示。(3) 新的知識表示被翻譯回行動。

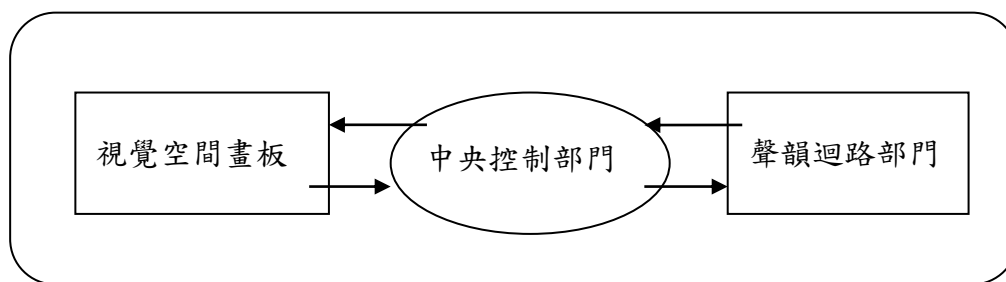
人腦的信息加工系統是由感受器（receptor）、反應器（effector）、記憶（memory）和處理器（或控制系統）（processor）四部分組成。首先，環境給予感覺系統一個訊息，即感受器接收到信息，感受器對信息進行轉換；轉換後的信息在進入長時記憶之前，要經過控制系統進行符號重構，辨別和比較。記憶系統貯存著可供提取的符號結構。最後，反應器對外界作出反應，如圖8。



圖表 8 信息加工系統的一般結構(車文博, 1998)

由於電腦的隱喻和使用，認知心理學在1960至1970年間得到許多人工智慧及其它相關領域研究成果的助益。目前已發展成為一個跨領域的認知科學，此學門整合了一系列不同取向關於心靈與心智處理的研究。

對於記憶的處理，一般可以分工作記憶和長期記憶，訊息經過工作記憶的處理後，轉存入長期記憶，而工作記憶需要動用到先前的背景知識或認知架構來幫忙處理新的訊息。「工作記憶（working memory）」是指個體在進行認知作業的歷程中，同時對訊息「短暫貯存(short term storage)」及「運作處理(processing)」的能力。Gathercole 和 Baddeley (1993) 提出工作記憶是個相當複雜的概念，至少是由「中央控制部門」、「聲韻迴路部門」、及「視覺空間畫版」等三種成分（components）所構成。



圖表 9 (Gathercole & Baddeley, 1993)

表格 2 工作記憶

中央控制部門	最重要的部份，負責分配與監管認知系統內的訊息。扮演的腳色為協調附屬部門間訊息的抑制、分配、提取、移轉等與注意力 (attention) 及統合力 (coordination) 相關之作業。
聲韻迴路部門	用來儲存語文材料的附屬系統，由「聲韻儲存」(phonological storage) 及「隱內覆誦」(subvocal rehearsal) 兩個子成分所構成。訊息是以聲音碼的形式所儲存，會隨著時間而消失，必須靠覆誦讓記憶保存下去。
視覺空間畫版	負責處理「視覺」(或影像) 和「空間」性質的訊息。

認知心理學上發現一個令人遺憾的是人類認知系統充滿了嚴重的信息處理的限制，特別是：短期記憶 (short-term memory, STM) 能力，大概只有七個組塊 (7 chunks)，也就是說，在給定的時間內只能學到一些資訊 (大約是一分鐘可以學到七個組塊)，在問題解決的搜尋效率上，可能只有一分鐘 10 個狀態 (states) (Gobet, 1993)。然而，在各個領域的專家展示給我們的卻是這些限制是可以被 (部分) 迴避，因為在實驗對象中，一位物理學家以 2 秒的速度當場解決一個困難的問題；國際象棋大師能夠蒙住眼睛玩遊戲 (Gobet & Simon, 1996)。

認知心理學的研究指出在跨領域的專業知識上，有三個重要的特點：圖形識別 (pattern recognition) 的重要性、選擇性搜尋 (selective search) 的重要性、該領域上豐富的知識。Chase 和 Simon (1973) 將這三個特點應用在記憶的組塊理論 (chunking theory of memory) 當中，並以西洋棋遊戲來實現。

在棋類研究上面，不少研究探討有關於在解決問題時記憶的配置有何差異。多半都是比較專家和新手之間的差別為何。De Groot (1966) 發現這兩者在廣度及深度搜尋上並沒有太大差異，反而主要不同的地方在於記憶位置 (positions)。Chase 和 Simon (1973) 發現在兩者想起整個盤面及每個移動組塊 (Chunk) 的順序時，組塊的數量並沒有明顯差異，但組塊的大小卻有著非常大的差距，專家所記得的組塊大小遠遠大於新手所記得的。這個結果被許多領域的人應用在不同的問題解決上，如：Egan 和 Schwartz (1979) 用於電子電路圖；Jeffries、Turner、Polson 與 Atwood (1981) 用於計算機程序中；解決代數問題、沃斯問題 (Cooper & Sweller, 1987; Sweller & Cooper, 1985)。

因此，本研究的重點在於，從認知的觀點上分析人類在玩益智遊戲時，所受到的限制為何？例如：在玩 Telescope Game 的時候，玩家並沒有獲得可以用別管子擋住其它管子或是擋住球，是否玩家會因為沒有獲得這樣資訊而導致陷在遊戲關卡之中。又或者是，再把球推到目的地前，必須先將盤面的狀態稍作調整 (先佈局)，才能夠順利完成目標等等。這些 Heuristic 如果事先告訴電腦，那對電腦而言是件很輕鬆就能解開的題目，然而，對於人類來說，要經過多少次的嘗試才能夠發現這些技巧呢？更進一步，若能夠清楚的知道說那些限制條件會影響解出答案的關鍵，藉由量化這些因素，並將其加入複雜度的衡量之中，或許能夠更貼切的反應出人類對難度的感受。

2.3 問題求解

一般而言，當一個代理人擁有多個評價未知的直接選項時，可先檢驗通往已知評價狀態的各種可能動作序列，然後從中選出最佳的序列。尋找此種序列的過程稱為搜尋 (search)。用搜尋法對問題求解，把問題作為輸入，並以動作序列的形式傳回問題的解 (solution)。一旦找到一個解，那麼他所建議的動作就可以付諸實施，也解是執行階段 (execution phase)。一個問題可以正式定義為四個部分：

- (1) 初始狀態 (initial state)
- (2) 動作 (actions)：最常用的正規方法為後繼函數 (successor function)。
- (3) 目標測試 (goal test)：用來確認給定的狀態是否為目標狀態。
- (4) 路徑成本 (path cost)：分配每條路徑一個數值化的成本。

搜尋的本質是，先處理其中一個狀態，其他的先擱置不處理，至於要挑選哪個狀態先給予處理，則是由搜尋策略 (search strategy) 而決定。

限制滿足問題 (Constraint satisfaction problem, CSP) 結合啟發式與組合搜索方法在合理的時間內解決問題。形式上，一個 CSP 的定義是由三個變數集合所組成 $\langle X, D, C \rangle$ ，其中 X 為一組變量， D 是一個域值， C 為一組限制。每個限制依次為一對 $\langle t, R \rangle$ ，其中 t 是一個元組變量 (tuple of variables)、 R 是一組元組的值，所有這些元組具有相同數目的元素，因此 R 代表的是一個關係。評估變量是把變量經由函數的轉換後所得到的值， $v: X \rightarrow D$ 。經由這樣來評價滿足限制條件 $\langle (v(x_1), \dots, v(x_n)), R \rangle$ ，如果 $(v(x_1), \dots, v(x_n)) \in R$ ，判斷滿足所有限制條件來得到問題的解 (高超群，2003)。

同樣的這些限制條件亦可以從益智遊戲中發現，例如：對於數獨而言，每列每行數字只能出現一次；對於 Telescope Game 而言，每根管子有它自己最大伸長量和固定伸縮方向；魔術方塊則是規定每列、行、對角線的合要是定值等等。這些皆為遊戲的限制條件。

但是，除了遊戲表面上規則所定的限制條件之外，往往遊戲內部隱含的技巧可能是影響遊戲難度最重要的因素。例如：在數獨中，井字型排除法是新手想不到的技巧，但偏偏有些關卡必須要知道這個方法才能夠解出來。不過，這種高階技巧是否真的就比較好呢？其實不然，有些關卡用這個方法反而會讓遊戲變的更難，這是一件很有趣的現象。本研究藉由使用某些基本的技巧解題，來探討這對於難度的界定除了技巧之外，其他的影響因素是甚麼？

2.4 益智遊戲中人類解題方式

2.4.1 邏輯繪畫拼圖



邏輯繪畫拼圖是個擁有 $N \times M$ 方格的益智遊戲，方格左方和上方提供了限制，用來定位方格中列與欄的資訊。下圖左邊為原始題目，右半部為解答。

			2	1	3	1	2				2	1	3	1	2
			2	3	1	3	2				2	3	1	3	2
		5								5					
1	1	1						1	1	1					
		3								3					
	2	2							2	2					
		5								5					

圖表 10 邏輯繪畫拼圖說明 1

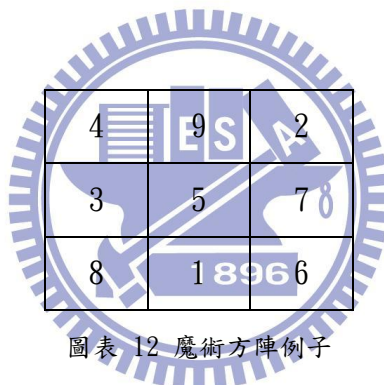
一般來說，會先找出唯一的 Pattern，在圖 10 中，(1, 1, 1) 和 (2, 2) 這兩組塗出來的樣式一定是固定的（在 5×5 中），如圖 11。依照此方法，逐一將圖拼出來。



圖表 11 邏輯繪畫拼圖說明 2

2.4.2 魔術方陣

魔術方陣是一個由 1 到 n^2 的數字所組成的 $n \times n$ 陣列，具有各對角線，各橫行與縱列的數字和（魔數）都相等的性質。



圖表 12 魔術方陣例子

一般解法是：先計算出魔數為多少，數字和=各行數字和（魔數）*行數。再來是從數字最多的地方下手，慢慢的將解答填出來。

2.4.3 數獨

「數獨」(sudoku) 名稱來自日文，但概念源自「拉丁方塊」，是十八世紀瑞士數學家歐拉發明的。遊戲規則很簡單，九個九宮格裡，每一直行與每一橫列都有 1 到 9 的數字，每個小九宮格裡也有 1 到 9 的數字，但一個數字在每個行列及每個小九宮格裡都只能出現一次。

7			5	6		9	8		7	4	3	5	6	2	9	8	1
									9	1	6	8	4	7	5	2	3
	8				1	4			2	8	5	9	3	1	4	7	6
		9			6	8			1	7	9	4	2	6	8	3	5
		4		9		6			8	5	4	7	9	3	6	1	2
		2	1			7			6	3	2	1	5	8	7	9	4
		1	2				5		4	6	1	2	7	9	3	5	8
									3	9	8	6	1	5	2	4	7
	2	7		8	4			9	5	2	7	3	8	4	1	6	9

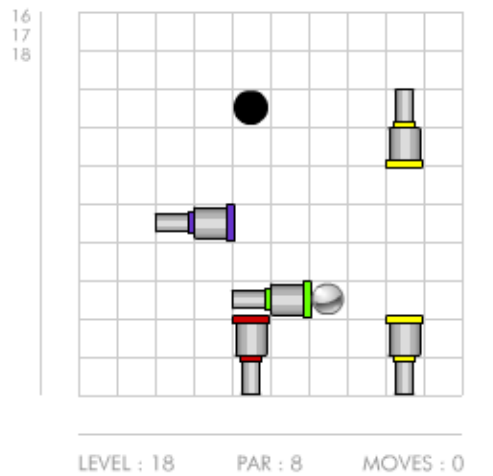
圖表 13 數獨解說，摘自：數獨大師

一般數獨的解法有直接排除法、隱格排除法（又分雙隱格、三隱格、大井字、小井字）、餘數排除法。與魔術方陣相同的地方在於，數獨也會先從數字多的那一行（列、九宮格）先填。

2.4.4 Telescope Game



Telescope Game 是由不同的管子和一個球所組成的遊戲，最終目的在於把球推入黑洞中。一般會先嘗試下一步能到把推到哪裡球，再來是思考怎麼把球推到洞裡。所以在思考把球推到洞裡的過程中，想法是哪根管子可以把球給推進去？這時就會有兩種人出現，一種是站在球的觀點要怎麼走，另一種人是要怎樣利用管子把球推進洞，因而對管子做一些前置動作（佈局）。在嘗試的過程中也會發現一些技巧，例如 18 關（下圖），若先把球經由綠色管子推到黃色管子上時，會發現球被困在那條垂直線上動彈不得，甚至懷疑是否有解。至於這個技巧，在此先賣個關子，有興趣的人可以去嘗試看看。



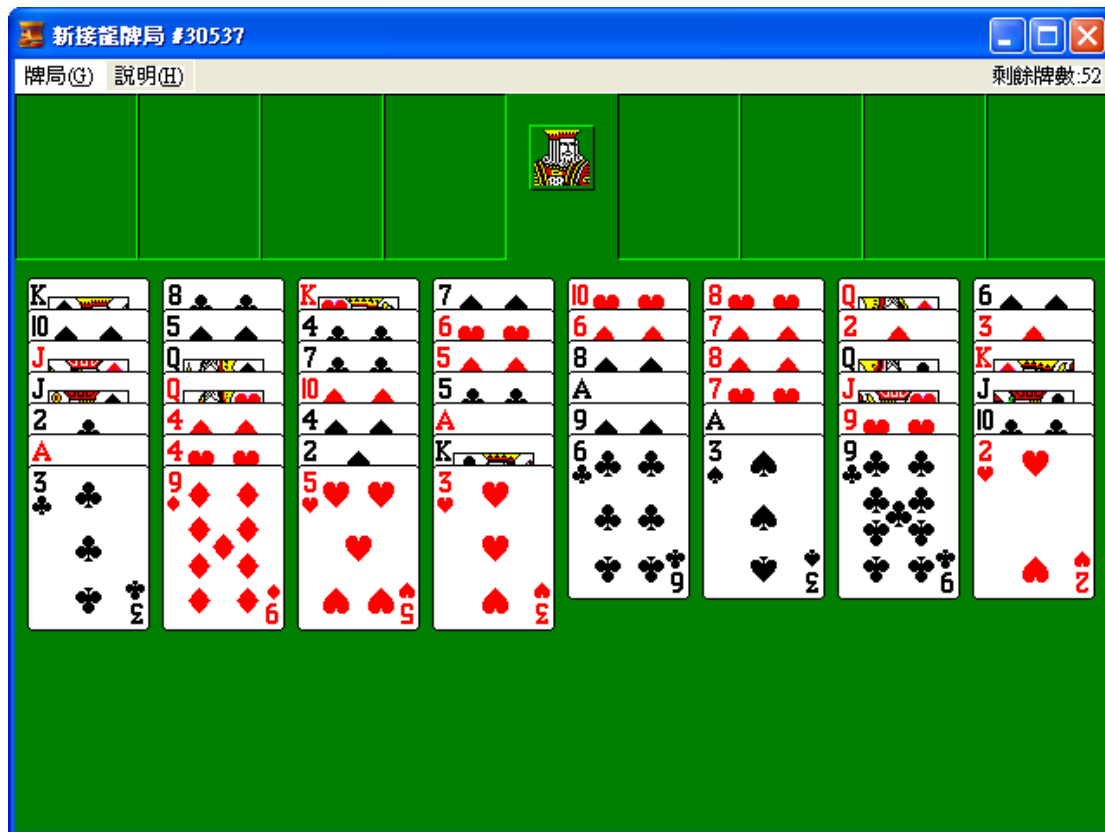
圖表 14 Telescope Game 解說

2.4.5 新接龍

新接龍，是一款使用 52 張牌進行的單人卡片遊戲，遊戲標的是將整副牌搬到右上角的四個空白欄中。將一副標準 52 張撲克牌洗牌，再分別置於 8 個欄目，其中 4 個欄目有 7 張紙牌，其餘 4 個欄目有 6 張紙牌，每張撲克牌皆要以翻開顯示。另外還有四個本位欄框、四個空白欄框，本位欄框為四個放牌位置。紙牌 A 可立即放到本位欄框，而其他同花色的牌則可以由小到大依序疊上去。只要將所有的牌都放到本位欄框中，就成功了。空白欄框為四個放牌位置。每個欄框都可任意放一張紙牌。

移動紙牌的規則如下：

- ✧ 將紙牌移動到某一欄時的順序必須為由大到小，而且是不同的花色。
- ✧ 將紙牌移至本位欄框時，必須以相同花色，將牌按照從低(A)到高(K)的順序移動。
- ✧ 某欄最底端的紙牌可移到空白欄框、另一欄的最底端或本位欄框。
- ✧ 空白欄框中的紙牌可移至某一欄的最底端或本位欄框。



圖表 15 新接龍解說

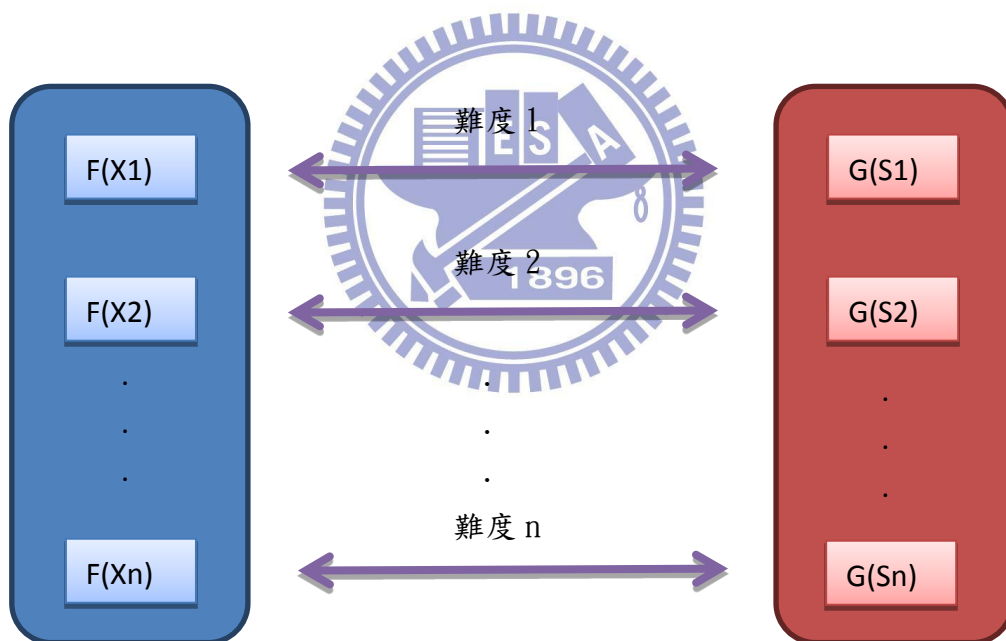
對於新接龍，一般人的想法是先找出 A 這張牌在哪裡，先想辦法把它放到本位欄框中，接著依序 A-K 這個順序逐一找到想到的牌將其置於本位欄框。其次，若是無法先將牌放到本位欄框中，則是先看看下面八行中是否有放置的地方，否則再把牌放到左上方的空白欄框（Chan, 2006）。

上述所提到的技巧，也正是先前在介紹 Heuristic 時，本研究想要量化的東西。相信這種小小的技巧，如果玩家先前就知道，那這關的難度瞬間會變的很容易；同理，對於電腦而言也是一樣，要讓電腦像人一般，人覺得難的東西電腦也這麼認為。如何適時的給予電腦知識、限度為何？這些都是值得思考的問題。

第三章 研究方法與設計

3.1 研究架構

本研究想藉由衡量益智遊戲的複雜度直接定義出遊戲的難度分類，首先，必須先找出影響難度的因素，再經由程式的分析比較 $G(S)$ 找到關係函式 $F(X)$ ，其中 X 包含各種影響複雜度的數值（如：搜尋次數、Backtrack等等）， $G(S)$ 代表人類玩某一題數獨（ S ）所獲得的解題速度的平均值。

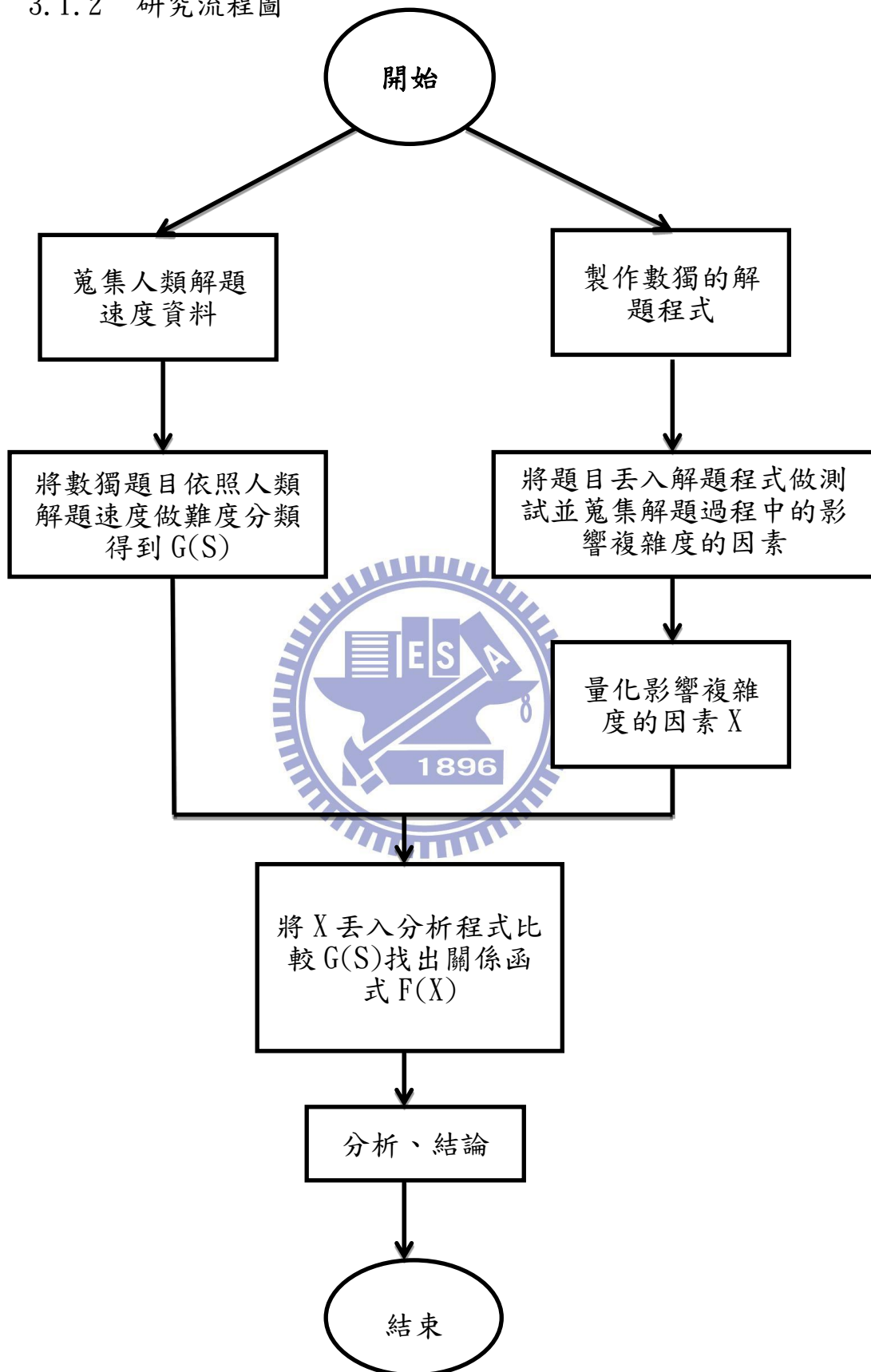


圖表 16 複雜度與人類解題速度

3.1.1 研究步驟：

- 1) 蒐集人類解題速度 $G(S)$
- 2) 量化影響複雜度因素 X
- 3) 將 X 帶入分析程式依據 步驟1) 所得到的難度分類找出關係函式 F

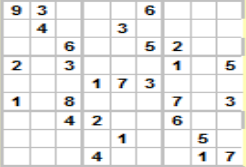
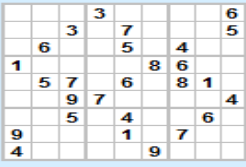
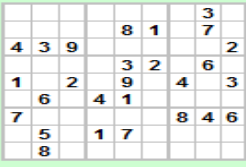
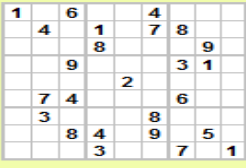
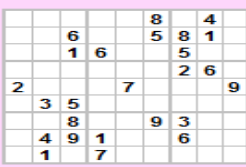
3.1.2 研究流程圖



圖表 17 研究流程圖

3.2 資料蒐集

本實驗所用到的數據來源為網路玩家在線上填寫數獨題目所統計下來的數值，根據「尤怪之家」(<http://oddest.nc.hcc.edu.tw/>)中所提供的上千題解題速度時間的蒐集，本實驗為了確保準確性，將從其中挑選出百人以上作答過的數獨題目，這邊依據人類解題速度將數獨題目分為 5 種難度，每種難度各挑 70 題共 350 題數獨作為研究分析中的訓練樣本；再從每種難度各挑 15 題共 75 題數獨作為測試樣本，並以人類解題平均用時作為參考標準，如圖 15 所示。

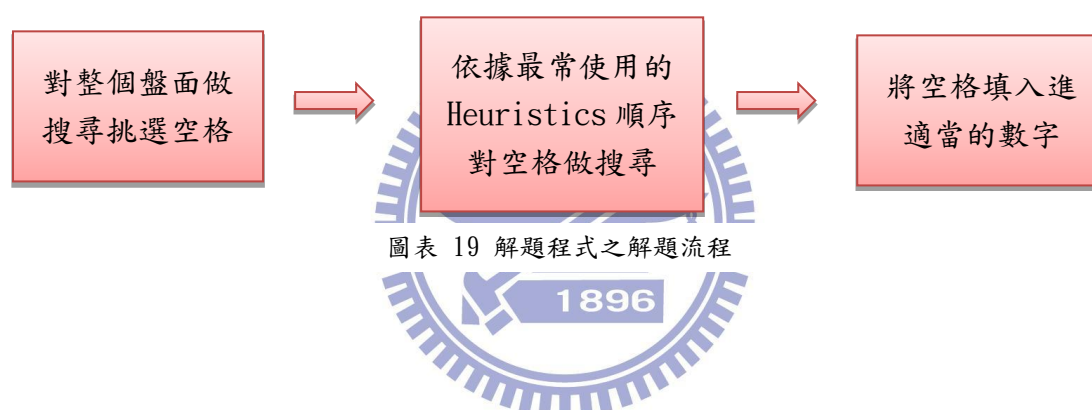
題號	sudoku	挑戰次數	成功次數	成功率	平均用時
20	 <p>★</p>	231	154	66.66%	4 '38 "
8	 <p>★★</p>	138	82	59.42%	6 '33 "
7	 <p>★★★</p>	133	88	66.16%	6 '57 "
6	 <p>★★★★</p>	141	86	60.99%	11 '23 "
1	 <p>★★★★★</p>	731	149	20.38%	25 '31 "

圖表 18 難度分類，摘自：尤怪之家

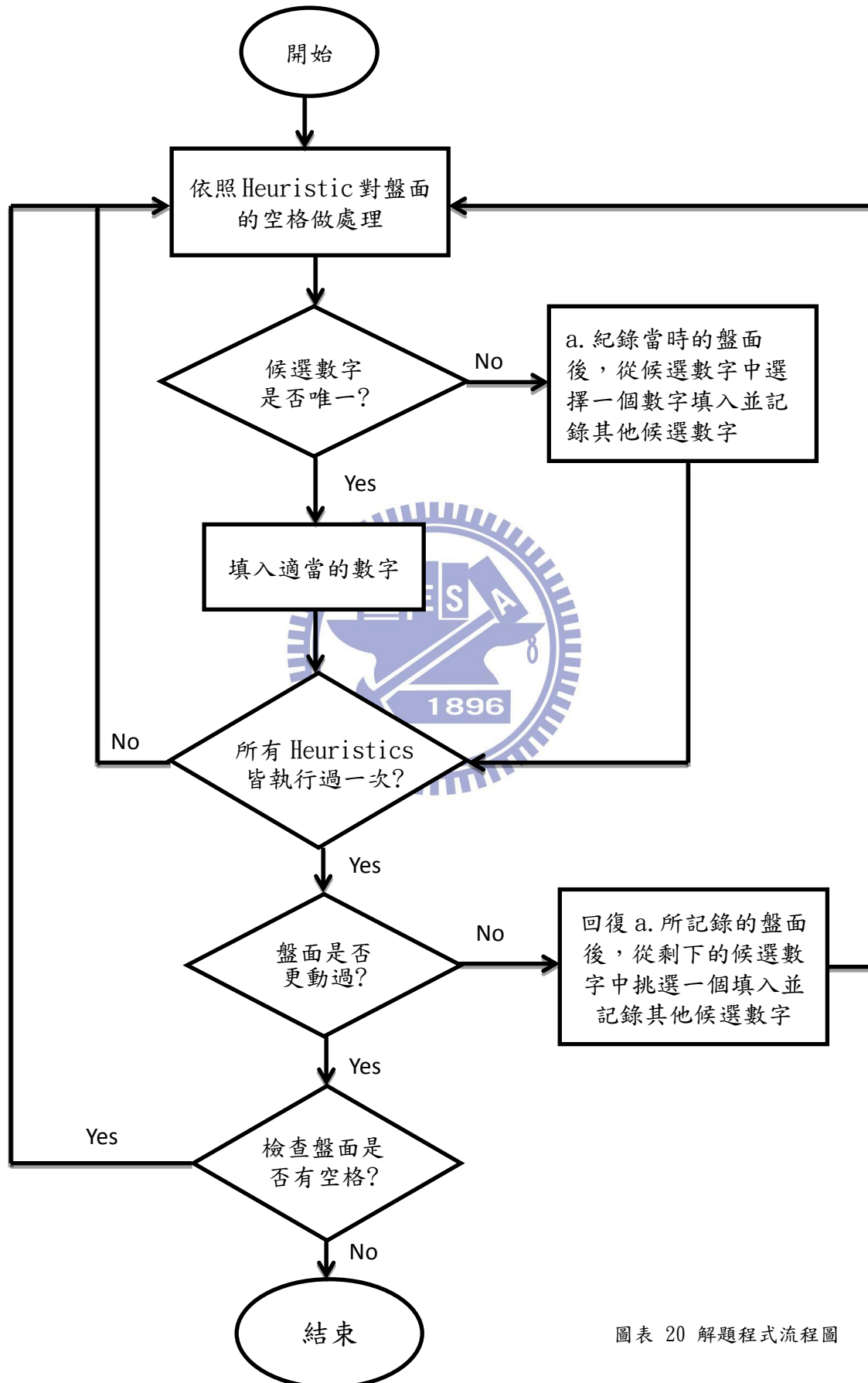
3.3 解題程式設計

3.3.1 程式架構

此解題程式從人類的角度為出發點，依循人類解數獨的方式去做設計。首先，在數獨的解題技巧中，直觀法是人類在解數獨時最常用的方法，其主要的技巧可分為餘數法及摒餘法兩大類。本程式以搜尋樹的概念為架構，並且加入一些人類解數獨的解題技巧（詳細的說明參考3.3.3）在其中，搜尋樹的分支搜尋優先順序則以候選數字的多寡做取捨，優先對候選數字數目較少的空格做處理。



3.3.2 程式流程圖



圖表 20 解題程式流程圖

3.3.3 數獨的Heuristic

直觀法的主要的技巧可分為下列兩大類：

1. 餘數法：觀察特定的空格是否有解的方法，也是初學者剛接觸數獨時，最容易理解及應用的方法。
2. 摒餘法：觀察特定的數字在某一單元是否有解的方法，也是入門後應用最廣的方法。

雖然入門後，大家在解題時必定優先使用摒餘法，但對於初學者而言，最先學會及理解的可能還是餘數法，技巧說明在附錄中有詳細的介紹。

針對於本研究的需要，在解題程式中所運用到的解題技巧，會由較低解題技巧對數獨解題，當找不到答案時會使用更高的解題技巧進一步試圖找出答案，企圖在玩家具備一定的技能條件中，找出可能影響難度的因素。

3.3.4 分支度

本研究將數獨的分支度分為可確定的分支和不可確定的分支，所謂的可確定分支是指利用唯一法或者是宮摒餘法找到某空格的唯一解答，而所謂的不可確定分支就是需要做猜測的空格。

3.3.5 初始盤面狀態

另外，本研究考慮到對於一個數獨題目來說，一開始容不容易找到可能的活路，換句話說就是容不容易一眼看出確定答案的空格，或者是確定可以走的活路多寡，是否與決定一個题目的難易是否有關，因此也將其列入考慮。

3.3.6 Backtrack的次數

在數獨中會用到 Backtracking 的地方在於，當我們使用直觀法去解題的時候，發現找不到只有一個數字可以填入空格時，這時候就必須從候選的數字中挑一個出來，由於這個是隨機挑選的，並不保證這個選擇一定是正確的，因此必須記錄當時的盤面，當一直找不出下一步解的時候就必須回到這個盤面重新做選擇，也就是猜測的步驟。

6			2	5		3	9	
			4		3	8		5
5					8			1
	1	2	8	3		5		
	9			6			8	
8		6		4			7	
3			7	2				8
2					4			
	7	4		8	5			

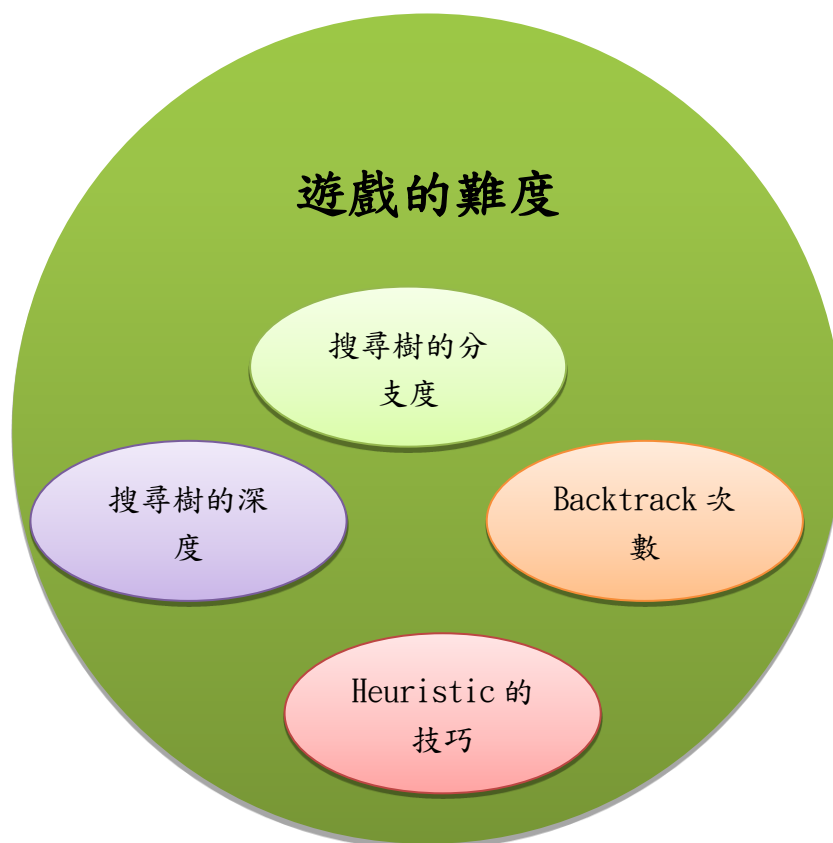
圖表 21 猜測的解說圖

由上圖可以看到第一行第一列的候選數字有 1、4、6、7 這時候就必須從這四個數字中挑選出一個數字填入且必須記錄當時的盤面以備猜測失敗後的回溯點。

3.4 分析的進行方式

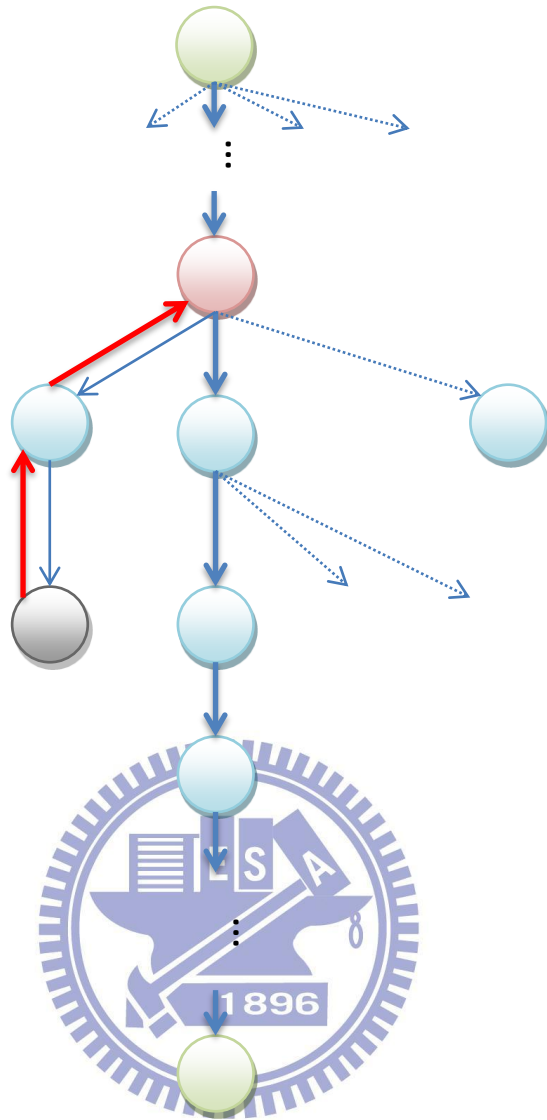
3.4.1 參數說明

遊戲的難度可能受到好幾種的因素所影響，如圖22所示。



圖表 22 遊戲的難度

本研究分析將從設計出來的解題程式中，把程式搜尋的過程中(如圖 23)，深度、廣度、執行次數、Backtrack 次數、數獨技巧(表 4)使用次數以及初始盤面中數字分布狀態等等做量化後，將這些數值給予不同的權重丟入分析程式中，以基因演算法演化找出評估難度的方程式。



圖表 23 搜尋樹

- 深度：藍色粗線的路徑長
- 廣度：藍色虛線的數目
- 執行次數：所有實線的數目
- Backtrack次數：紅色粗線的數
- 猜測次數：紅色節點數目

表格 3 數獨技巧 (高低)

數獨技巧	等級
唯一解法	低 ↓ 高
宮摒餘解法 (只使用行、列摒除)	
宮摒餘解法 (加入區塊摒除)	
宮摒餘解法 (再加入單元摒除)	
二餘解法	
三餘解法	
四餘解法	
數對摒除解法	
數對卡位或區塊 (單元)數對唯餘解法	
唯餘解法 (五、六餘法)	
行、列摒餘解法	
矩形摒除解法	
數偶摒除解法	
猜測解法	

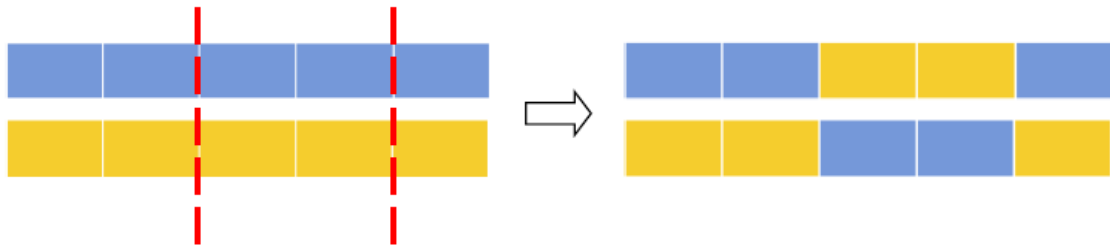
3.4.2 以基因演算法找出評估難度之方程式

基因演算法 (Genetic Algorithm) 是取法大自然的一種演算法，藉著對於演化現象的觀察，Holland (1975) 認為可以透過把問題轉為基因型 (Genotype)，利用競爭-生存以及基因交換-突變，尋找出問題的正確解答。透過競爭-生存，擁有好基因的品種會有較高的機會生存到下一代，而與生存較無關的基因則會隨著時間逐漸被淘汰。

本研究依據這套法則，首先隨機產生 500 的子代，每條基因的型態如下圖所示，再根據演化的法則先挑選出優秀的母代做交配 (採用雙點交配)，之後，再以一定的突變機率令產生出來的新子代發生突變。最後，保留母代中最優秀的基因 (前 10%)，成為最終的新子代。

$A*X$	$B*X^2$	$C*\log(X)$...	$D*X*Y$	$E*Y^3$
-------	---------	-------------	-----	---------	---------

圖表 24 基因型態



圖表 25 雙點配對

其中，適應值（誤差值）為一減去相關係數，採用的是依據斯皮爾曼等級相關（Spearman Rank Correlation）對人類解題時間與解題程式之複雜度做比較，產生斯皮爾曼相關係數，目標是盡量讓相關係數越高越好。



第四章 結果分析

本研究的結果分為兩個部分，一個部分是假設解題程式是新手的狀態，並不會用太多高深的技巧去解數獨，因能力不足而找不到答案時就用嘗試錯誤的解題策略去找出數獨的解，並以解題的搜尋過程中的統計數字做為衡量該題難度的參數；另一個部分是讓解題程式已經進階成專家的程度，利用不同的解題技巧去對數獨題目做解題，主要是以目的分析法的解題策略之概念去找出數獨的答案，並以解題的搜尋過程中用到的高階技巧數量做為衡量該題難度的參數。

4.1 解題程式蒐集的數據說明

4.1.1 第一部分

本研究將 350 題數獨，難度 1~5 星各 70 題，丟入第三章所提的程式架構之解題程式後，分別獲得以下多個參數：

- 1 執行次數， x_1 。
- 2 猜測次數， x_2 。
- 3 Backtrack 次數， x_3 。
- 4 分支度 1 (確定答案的分支)， x_4 。
- 5 分支度 2 (不確定答案的分支)， x_5 。
- 6 X_4/x_5 ， x_6 。
- 7 初始盤面上，已存在 6 個數字的九宮格個數， x_7 。
- 8 初始盤面上，已存在 5 個數字的九宮格個數， x_8 。
- 9 初始盤面上，已存在 4 個數字的九宮格個數， x_9 。
- 10 初始盤面上，已存在 3 個數字的九宮格個數， x_{10} 。

- 11 初始盤面上，已存在 2 個數字的九宮格個數，x11。
- 12 初始盤面上，已存在 1 個數字的九宮格個數，x12。

首先，會先對每項參數做正規化的動作，調整到 1~10 之間，再丟入基因演算法的程式中去演化找出能夠表達複雜度分布之非線性方程式 $F(x)$ ，並將得到的難度值作正規化，調整到 0-100 之間。

4.1.2 第二部分

統計解題程式在解題的過程中所使用到的各項技巧之次數(資料來源:尤怪之家提供的解題程式之解題過程)

表格 4 數獨技巧 (演化參數)

	使用次數
唯一解法	Y1
宮摒餘解法 (只使用行、列摒除)	Y2
宮摒餘解法 (加入區塊摒除)	Y3
宮摒餘解法 (再加入單元摒除)	Y4
二餘解法	Y5
三餘解法	Y6
四餘解法	Y7
數對摒除解法	Y8
數對卡位或區塊 (單元)數對唯餘解法	Y9
唯餘解法 (五、六餘法)	Y10
行、列摒餘解法	Y11
矩形摒除解法	Y12
數偶摒除解法	Y13
猜測解法	Y14

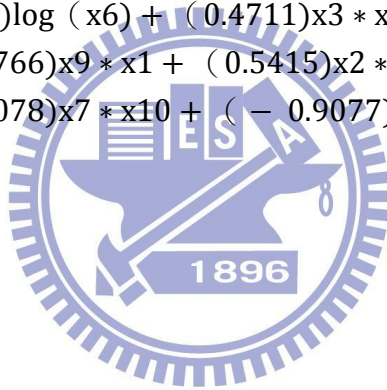
這部分並沒有做正規化的動作，因為每個變數之間的差異不會很大。同樣的丟入基因演算法中找出表達複雜度分布之線性方程式。

4.2 演化結果分析

4.2.1 第一部分 - 搜尋樹分析 (新手解題方式)

將 350 題數獨樣本當作訓練資料(training data)並使用基因演算法找出衡量難度的方程式 (方程式 1)，大約演化了五十世代後，適應值漸趨穩定，整體的平均誤差接近 0.3，最低誤差降至 0.2 (圖 26 所示，誤差值 = 1 - 斯皮爾曼相關係數)。將 350 題數獨樣本估算其難度值，如圖 27 所示。整體看來每種難度等級間有著明顯的分隔，但在低難度數值端與高難度數值端，有少數的樣本無法很清楚的區隔開，如：難度 2 和難度 3 在左邊有一段兩條線之間交錯。將難度 1~5 星，分別有 70 個樣本數，取其難度值的總平均來看，呈現漸進的趨勢，與預期的想法是符合的 (圖 28 所示)。本研究將找到的方程式對每一個數獨樣本作難度評分後，對其做排序後並與相對應的人類解題時間做相關關係的衡量，發現此方程式與實際解題時間的相關性有 0.8，呈現高度相關的關係 (表 7)。另外，將完全沒訓練過的 75 題數獨難度值做相關係數衡量，其相關性為 0.69 (表 8)，呈現顯著相關。因此，本研究在第一部分對難度的定義相當符合人類感知上的難度。本研究並將方程式 1 做因素分析之後得到表格 5，發現不確定分支度、猜測次數和回溯的次數對整體的難度值的影響較大，因此本研究推論，對於新手玩家來說，在具備較少的能力 (數獨技巧) 下，解題過程中失敗的機會較大，在反覆的嘗試錯誤當中，會讓玩家產生一定的壓力，可能會覺得繁複，因而對問題有了難或簡單的認知。

$$\begin{aligned}
F(X) = & (0.7960)x_5 * x_5 + (-0.5159)x_4 * x_6 + (0.5731)\log(x_{11}) \\
& + (0.0477)x_2 * x_3 + (0.3438)x_{10} * x_1 + (0.6370)x_5 * x_1 \\
& + (-0.2190)x_2 * x_2 + (-0.8851)x_1 * x_2 + (0.9298)x_3 * x_1 \\
& + (0.0218)x_8 * x_{10} + (-0.7694)x_4 * x_{11} + (0.1988)x_8 * x_4 \\
& + (-0.3555)x_3 * x_{12} + (-0.3610)x_2 * x_3 + (0.2330)x_2 \\
& * x_4 + (0.5954)x_9 * x_8 + (-0.6981)x_6 * x_2 \\
& + (-0.8951)\log(x_2) + (-0.5309)x_2 * x_{10} + (0.7112)x_5 \\
& * x_3 + (0.5081)x_2 * x_1 + (-0.4237)x_3 * x_5 + (0.0754)x_6 \\
& * x_3 + (0.5023)x_3 * x_{11} + (-0.6704)x_9 + (-0.7332)x_8 \\
& * x_{10} + (-0.8771)x_9 * x_7 + (-0.4292)x_8 * x_1 \\
& + (-0.3457)x_8 * x_3 + (-0.3573)x_2 * x_{11} + (-0.7831)x_9 \\
& * x_8 + (-0.2756)x_8 * x_8 + (0.8528)x_5 * x_{11} + (0.0374)x_{10} \\
& * x_6 + (-0.2866)x_2 * x_2 + (0.6344)x_7 * x_{12} \\
& + (-0.6276)x_8 * x_7 + (0.6590)x_3 * x_3 + (-0.1202)x_6 \\
& * x_{10} + (0.0766)x_{11} * x_3 + (0.8478)x_2 * x_2 \\
& + (0.2464)\log(x_6) + (0.4711)x_3 * x_8 + (-0.6767)x_1 * x_{11} \\
& + (-0.6766)x_9 * x_1 + (0.5415)x_2 * x_7 + (0.5617)x_5 * x_4 \\
& + (-0.9078)x_7 * x_{10} + (-0.9077)x_2 * x_8 + (-0.0639)x_5 \\
& * x_{11}
\end{aligned}$$



方程式 1 搜尋樹參數演化所得

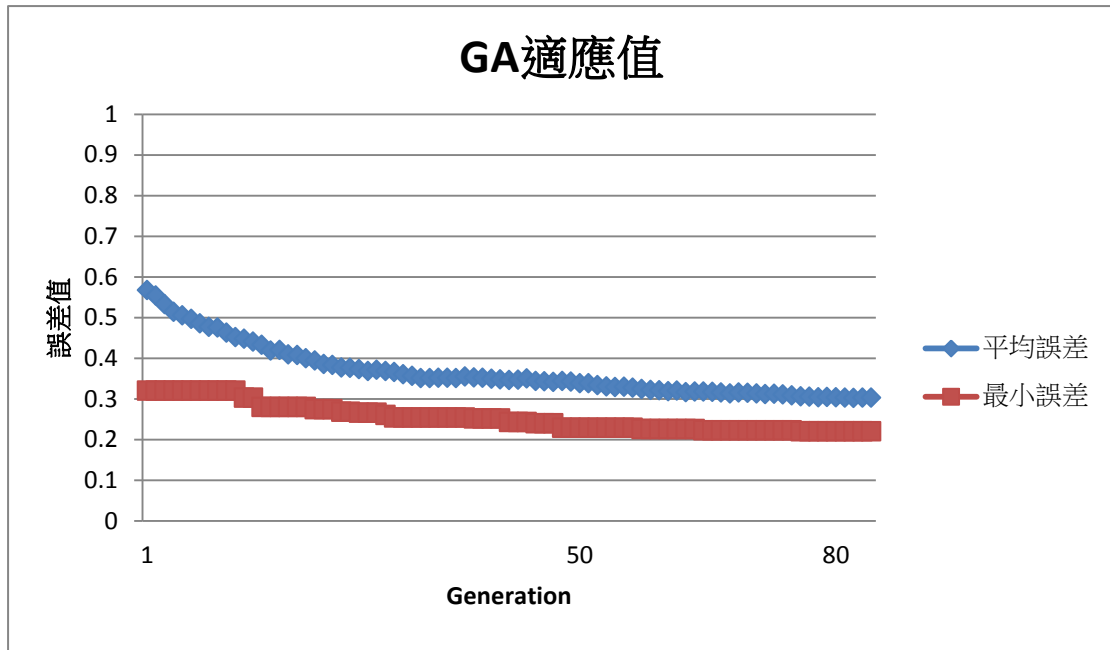
表格 5 方程式 1 - 因素分析

解說總變異量

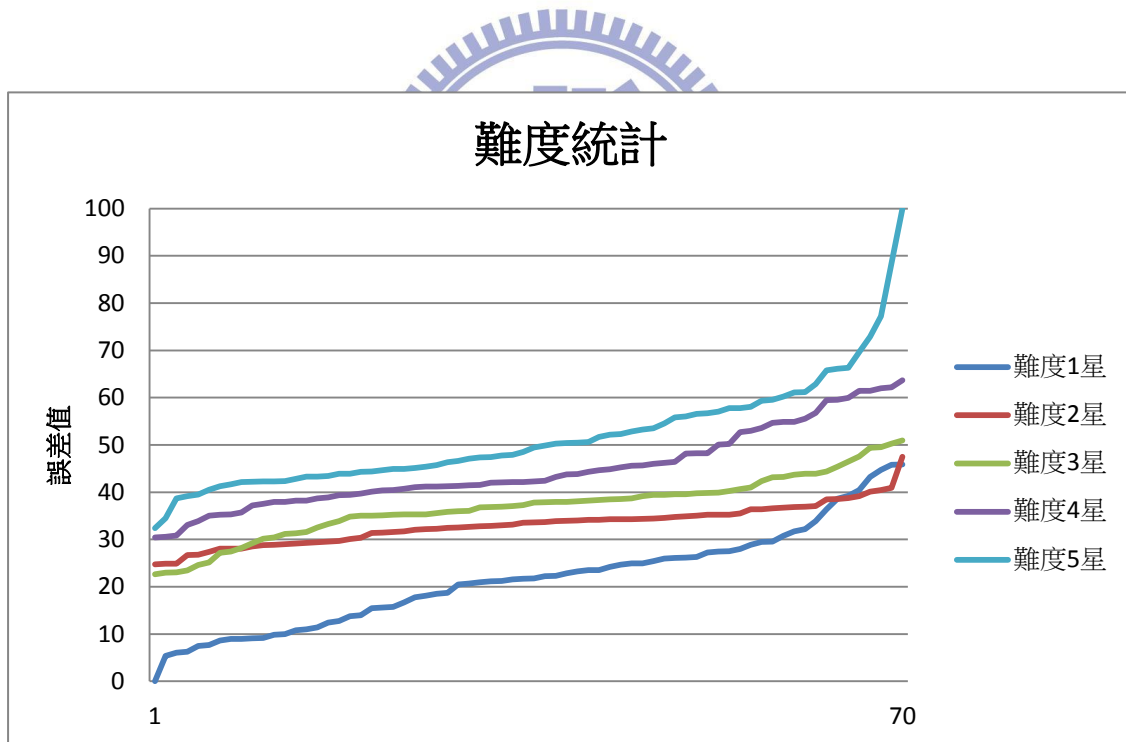
元件	初始特徵值			平方和負荷量萃取		
	總數	變異數的 %	累積%	總數	變異數的 %	累積%
(0.7960)x5*x5	19.951	39.902	39.902	19.951	39.902	39.902
(-0.5159)x4*x6	9.603	19.206	59.108	9.603	19.206	59.108
(0.5731)log (x11)	5.379	10.759	69.867	5.379	10.759	69.867
(0.0477)x2*x3	5.009	10.019	79.886	5.009	10.019	79.886
(0.3438)x10*x1	3.477	6.954	86.840	3.477	6.954	86.840
(0.6370)x5*x1	2.349	4.698	91.538	2.349	4.698	91.538
(-0.2190)x2*x2	1.670	3.340	94.877	1.670	3.340	94.877
(-0.8851)x1*x2	.512	1.024	95.901			
(0.9298)x3*x1	.426	.852	96.753			
(0.0218)x8*x10	.380	.760	97.513			
(-0.7694)x4*x11	.271	.543	98.056			
(0.1988)x8*x4	.245	.490	98.545			
(-0.3555)x3*x12	.126	.251	98.797			
(-0.3610)x2*x3	.107	.213	99.010			
(0.2330)x2*x4	.097	.195	99.205			
(0.5954)x9*x8	.082	.163	99.368			
(-0.6981)x6*x2	.071	.142	99.510			
(-0.8951)log (x2)	.053	.106	99.617			
(-0.5309)x2*x10	.047	.094	99.710			
(0.7112)x5*x3	.042	.084	99.795			
(0.5081)x2*x1	.033	.065	99.860			
(-0.4237)x3*x5	.022	.044	99.904			
(0.0754)x6*x3	.018	.036	99.940			
(0.5023)x3*x11	.014	.027	99.967			
(-0.7332)x8*x10	.010	.019	99.987			
(-0.6704)x9	.003	.006	99.993			
(-0.8771)x9*x7	.002	.003	99.996			
(-0.4292)x8*x1	.001	.003	99.999			
(-0.3457)x8*x3	.000	.001	100.000			

$(-0.3573)x2*x11$.000	.000	100.000		
$(-0.7831)x9*x8$	4.159E-5	8.317E-5	100.000		
$(-0.2756)x8*x8$	1.552E-5	3.103E-5	100.000		
$(0.8528)x5*x11$	5.579E-6	1.116E-5	100.000		
$(0.0374)x10*x6$	3.486E-6	6.972E-6	100.000		
$(-0.2866)x2*x2$	8.410E-7	1.682E-6	100.000		
$(0.6344)x7*x12$	4.623E-7	9.247E-7	100.000		
$(-0.6276)x8*x7$	3.119E-7	6.239E-7	100.000		
$(0.6590)x3*x3$	2.432E-8	4.865E-8	100.000		
$(-0.1202)x6*x10$	9.612E-9	1.922E-8	100.000		
$(0.0766)x11*x3$	4.962E-9	9.925E-9	100.000		
$(0.8478)x2*x2$	2.398E-9	4.796E-9	100.000		
$(0.2464)\log(x6)$	1.448E-9	2.897E-9	100.000		
$(0.4711)x3*x8$	1.290E-9	2.580E-9	100.000		
$(-0.6767)x1*x11$	9.601E-10	1.920E-9	100.000		
$(-0.6766)x9*x1$	3.548E-10	7.097E-10	100.000		
$(0.5415)x2*x7$	2.341E-10	4.682E-10	100.000		
$(0.5617)x5*x4$	5.567E-11	1.113E-10	100.000		
$(-0.9078)x7*x10$	3.756E-11	7.512E-11	100.000		
$(-0.9077)x2*x8$	3.476E-11	6.952E-11	100.000		
$(-0.0639)x5*x11$	2.000E-11	4.001E-11	100.000		

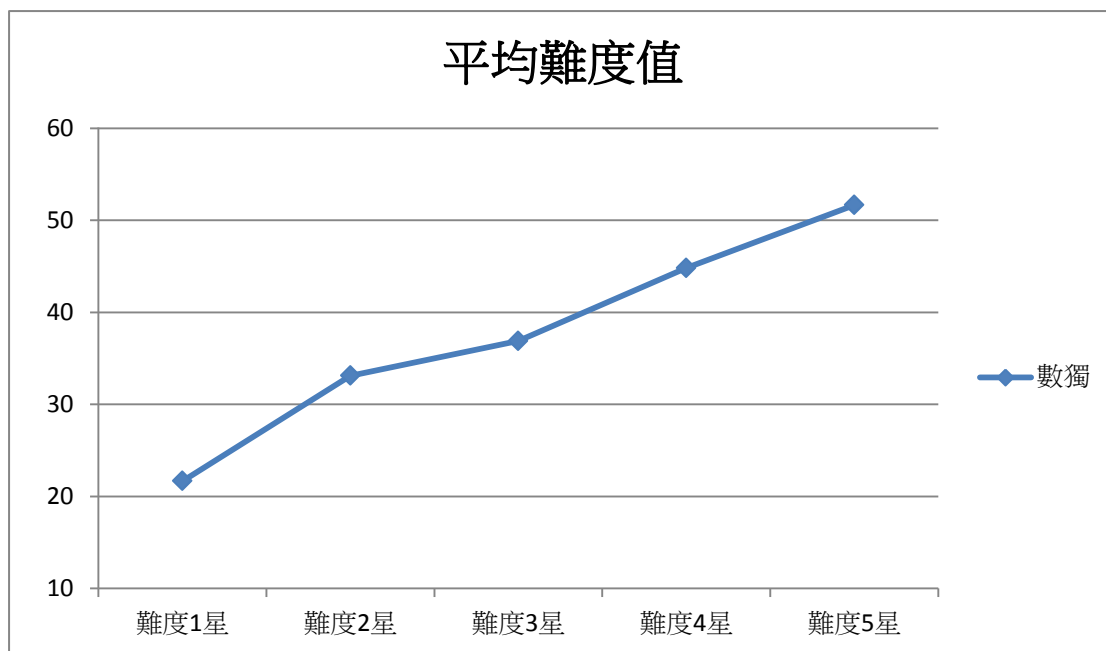
萃取法：主成份分析。



圖表 26 GA 適應值 (第一部分)



圖表 27 難度值分布 (第一部分)



圖表 28 平均難度值 (第一部分)

表格 6 標準差 (第一部分)

難度等級	標準差 (350 個樣本)
1	21.30560
2	8.40205
3	13.31934
4	16.96801
5	23.40265
Total	27.00876

表格 7 相關 (第一部分, training data)

			人類解題時間 排序	本研究難度評 估方程式排序
Spearman' s rho 係數	人類解題時 間排序	相關係數 個數	1.00 350	.80** 350
	難度評估方 程式排序	相關係數 個數	.80** 350	1.00 350
**, 相關的水準度為 0.01 (雙尾)				

表格 8 相關 (第一部分, test data)

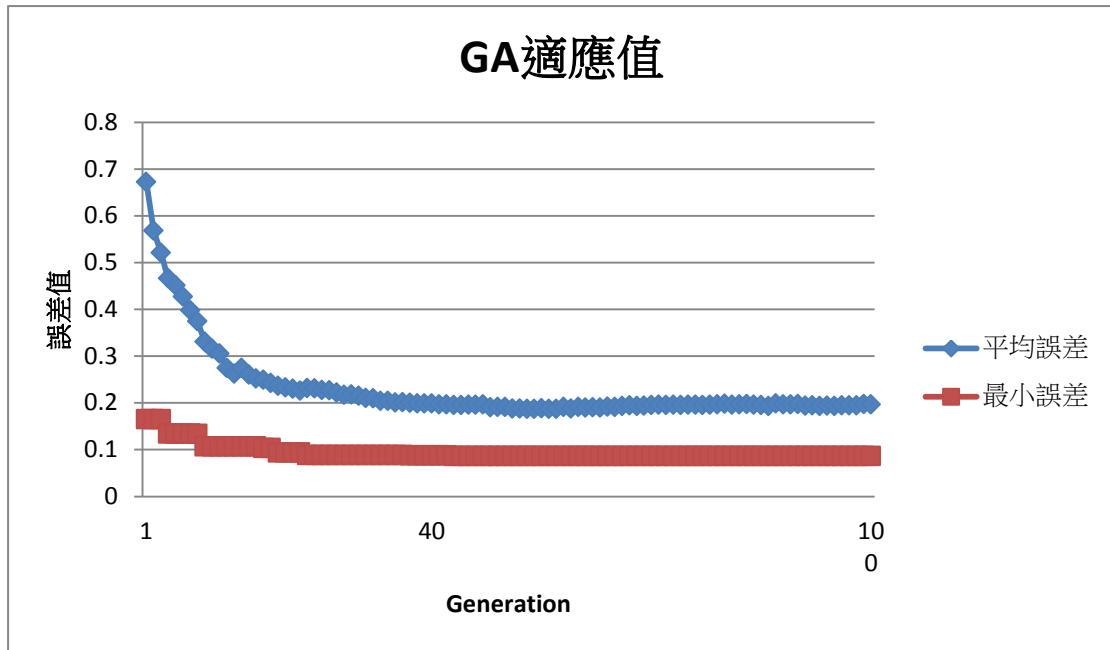
			人類解題時間 排序	本研究難度評 估方程式排序
Spearman' s rho 係數	人類解題時 間排序	相關係數 個數	1.00 .	.69** .000
	難度評估方 程式排序	相關係數 個數	75 .69**	75 1.000
**, 相關的水準度為 0.01 (雙尾)				

4.2.2 第二部分 - 解題技巧使用次數分析 (老手解題方式)

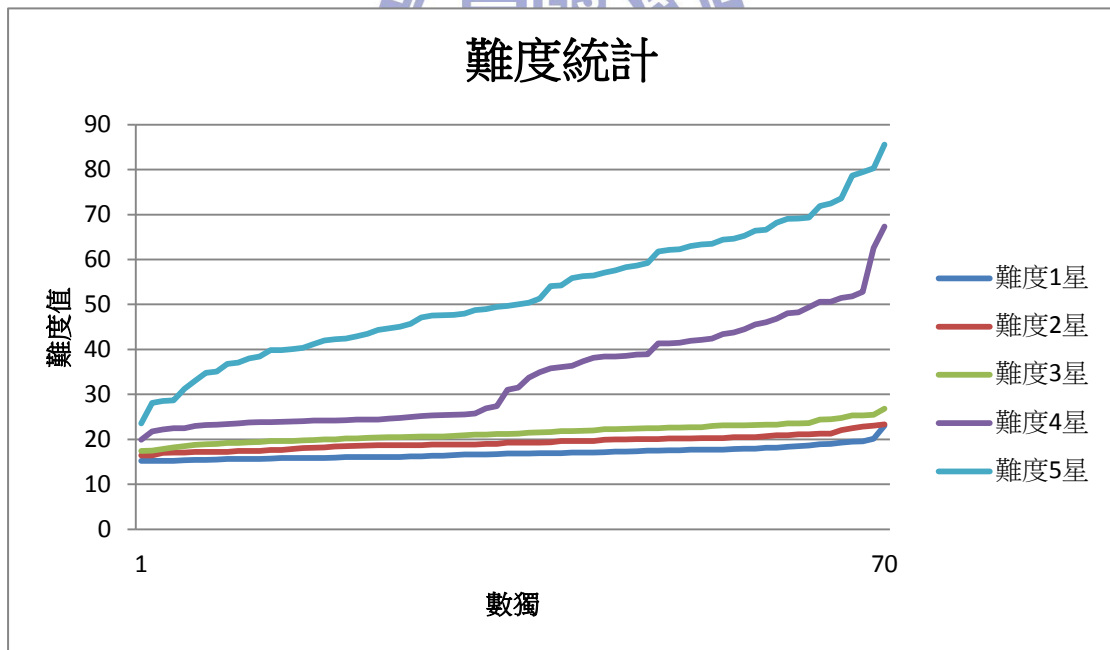
將 350 題數獨樣本當作訓練資料(training data)並使用基因演算法找適應函式找出衡量難度的方程式 (方程式 2)，大約演化了四十世代後，適應值漸趨穩定，整體的平均誤差接近 0.2，最低誤差降至 0.1 (圖 29 所示，誤差值 = 1 - 斯皮爾曼相關係數)。將所有數獨樣本估算其難度值，如圖 30 所示。整體看來每種難度等級間有著明顯的分隔，但在低難度的數獨樣本間彼此並沒有太大的差異，有少數的樣本無法很清楚的區隔開，如：難度 1、難度 2 和難度 3 在左邊有一段彼此之間幾乎快要重疊。將難度 1~5 星，分別有 70 個樣本數取其難度值的總平均來看，呈現漸進的趨勢，與預期的想法是符合的 (圖 31 所示)。本研究將找到的方程式對每一個數獨樣本作難度評分後，將其由低到高排序並與對應的人類解題時間做相關關係的衡量，發現此方程式與實際解題時間的相關性有 0.92，呈現高度相關的關係 (表 10)。另外，將完全沒訓練過的 75 題數獨難度值做相關係數衡量，其相關性為 0.77(表 11)，呈現非常顯著相關。因此，本研究在第二部分對數獨難度的定義非常符合人類感知上的難度。以數獨專家的角度對數獨解題的時候，可以清楚的從圖 30 看到，難度 1~3 星的題目對一個已經是專業等級的玩家來說並沒有太大差異 (彼此之間難度值不會差太多)，因為在解題的過程中並不需要用到太高難度的技巧去解題，用宮摒餘法來解數獨對於其來講，有如一個數學家用加減乘除解 $1+1=?$ 的問題，那樣容易。

$$F(Y) = (0.5736)Y_1 + (0.1672)Y_2 + (0.9720)Y_3 + (0.8140)Y_4 + (0.5957)Y_5 \\ + (-0.6099)Y_6 + (2.5141)Y_9 + (1.6968)Y_{10} + (-0.4094)Y_{11}$$

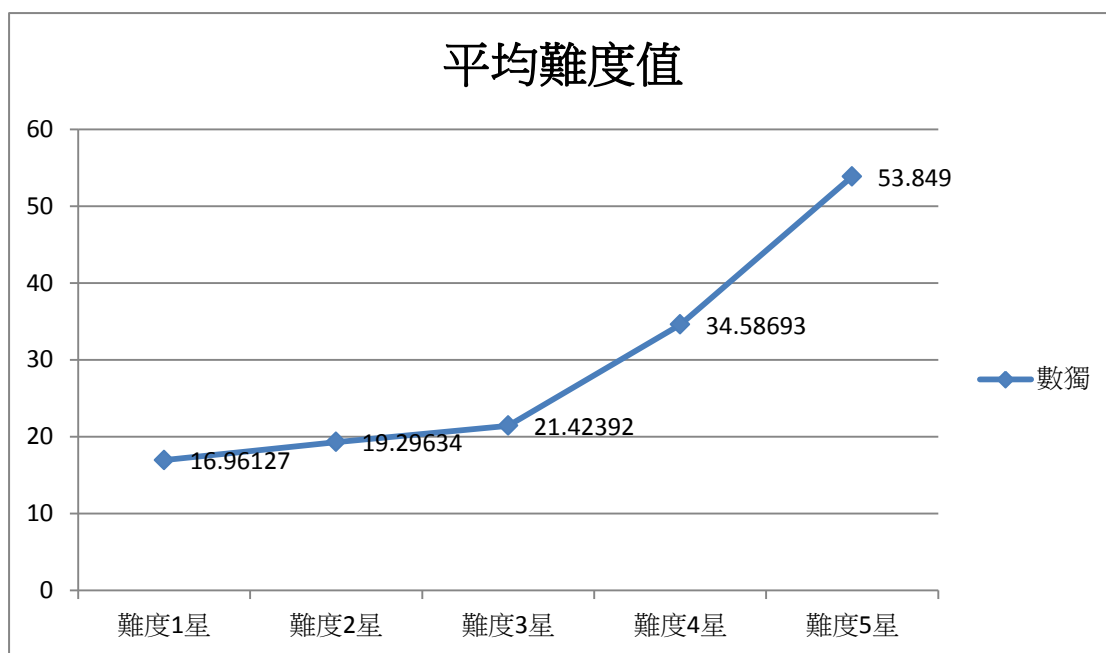
方程式 2 解題技巧次數演化所得



圖表 29 GA 適應值 (第二部分)



圖表 30 難度值分布 (第二部分)



圖表 31 平均難度值 (第二部分)

表格 9 標準差 (第二部分)

難度等級	標準差 (350 個樣本)
1	1.39715
2	1.58009
3	2.02716
4	11.25786
5	14.15430
Total	15.49094

表格 10 相關 (第二部分, training data)

			人類解題時間 排序	本研究難度評 估方程式排序
Spearman' s rho 係數	人類解題時 間排序	相關係數 個數	1.00 350	.92** 350
	難度評估方 程式排序	相關係數 個數	.92** 350	1.00 350
**, 相關的水準度為 0.01 (雙尾)				

表格 11 相關(第二部分, test data)

			人類解題時間 排序	本研究難度評 估方程式排序
Spearman' s rho 係數	人類解題時 間排序	相關係數 個數	1.00 .	.77** .000
	難度評估方 程式排序	相關係數 個數	75 .77**	75 1.000
**, 相關的水準度為 0.01 (雙尾)				

4.3 小結

由第一部分和第二部分的分析看來，以目的分析法的解題策略（老手）來對數獨做難度衡量時，所得到的相關度會更高於以嘗試錯誤法的解題策略（新手）。可見，針對數獨來說，這個遊戲還是傾向於以解題技巧為重的益智遊戲，難度會隨著玩家使用到的技巧高低以及次數而有所變化，但嘗試錯誤法其所得到的相關度也有高達 0.8，因此在預設一個新遊戲還尚未被人們所熟悉時，遊戲設計者在衡量難度時，解題過程中，不確定分支數和回溯次數亦有一定的參考價值。



第五章 結論與建議

5.1 結論

根據第一章第二節 Hayes (1981)所提出來的問題解決歷程，再比較第四章中分析影響複雜度的因素，發現在一個問題的形成過程當中，研究問題目前所處的狀態與研究者預期達到的理想狀態中間所產生的歧異，和問題解決歷程中計畫解題方式與重思問題或許有很大的相關性，因為在這反覆地思考過程中，可能會影響一個人的解題時間和步數。這就如同本研究在第四章中所發現的，對於數獨解題過程中，它的不確定分支會使玩家進行猜測的動作，當猜錯時，執行次數以及回溯的次數會因而增加，這些與決定難度有著很大的相關性。

本研究主要貢獻在於找出衡量對人類所感知的數獨關卡難度的方程式。此方程式利用回溯次數、不確定的分支度、計算次數等跟人類解題難度較具直接相關性的量化資料，將數獨關卡的難度數值化，讓我們可以依此難度數值排列數獨關卡。實驗結果顯示，此數值與人類解題的時間呈高度相關性，表示此數值確實可以衡量人類所感知的數獨關卡的難度。

依據第一章第二節 Klein (1996)所提出來的問題解題策略，本研究利用最常見的兩種捷思法去對解題程式做設計，發現對於這樣的設計出來的結果，也就是第四章所提到的人類解題時間與解題程式的複雜度的相關度可以達到八成左右。由此推測，對於一般人（新手）而言，當碰到新問題的時候，他們可能習慣用嘗試錯誤的方式去把問題給解決掉；而對於專家而言，反而會嘗試用各種技巧去解決問題（手段-目的分析法），因而使得難度會因解題技巧的高低而產生差異。

5.2 建議

首先，本研究實驗第一部分之所以不使用太過於複雜的數獨解題技巧，而挑選大部分益智遊戲在解題過程具有的特性（如：猜測、Backtrack）。其中一個原因就是希望這個難度評估的方式可以應用到與數獨類似的益智遊戲上面（如：新接龍、踩地雷等等）。因此，對於未來如果要評估這類遊戲的難度，或許可以將不確定分支度、回溯的次數以及執行的次數當作一個參考依據。在遊戲關卡設計上，通往目標的路口越多，越會讓玩家覺得複雜，也就是說關卡的難度會提升；除此之外，設計某些關卡需要某些特定技巧才能解決並想辦法在前幾個關卡中讓玩家學習、發覺，如此一來，玩家在破關的過程中就不會因為太無聊而覺得無趣或者是因為不知道有什麼方法可以破關而覺得焦慮。

其次，雖然本研究的目的不在探討人類的實際解題歷程，而是使用到人實際去玩數獨的解題時間直接當作對數獨難度的分類，在這個部分並沒有實際去問玩家玩的心情以及玩家所認為的難度應該為何？所以未來如果是想針對數獨做更深入的了解，可以在人類解題歷程上去做更詳細的探討。

參考文獻

一、中文部分

巫光禎 (2006)。數獨解題技巧，直觀法【部落格文字資料】。取自
<http://oddest.nc.hcc.edu.tw/>。

高超群(譯)(2006)。人工智慧：現代方法(原作者：S. Russell and P. Norving)。台灣：全華圖書。(原出版年：2003)。

二、英文部分

Abbott, R. (1975). Under the strategy tree. *Games & Puzzles*, 36. Retrieved from <http://www.logicmazes.com/games/tree.html>

Althofer, I. (2003). Computer-aided game inventing. *Technical Report*. Retrieved from http://www.minet.uni-jena.de/preprints/althofer_03/CAGI.pdf.

Birkhoff, G. D. (1933). *Aesthetic measure*, Harvard University Press, Cambridge.

Brailsford, S. C., Potts, C. N., & Smith, B. M. (1999). Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research* 119, 557-581.

Browne, C. B. (2008). *Automatic generation and evaluation of recombination games*, PhD thesis, Queensland University of Technology.

Chan, C. (2006). *Helping human play freecell*, University of Kent.

Retrieved from

<http://www.cs.kent.ac.uk/pubs/ug/2006/co620-projects/revrole/HelpingHumanPlayFreecell.pdf>

- Chase, W. G., & Simon, H. A. (1973). Perception in chess. *Cognitive Psychology*, 4, 55-81.
- Cooper, G., & Sweller, J. (1987). Effects of schema acquisition and rule induction on mathematical problem-solving transfer. *Journal of Educational Psychology*, 79, 347-362.
- Craik, K. (1943). *The nature of explanation*. Cambridge: Cambridge University Press.
- Crawford, C. (1984). *The art of computer game design*. Osborne/McGraw-Hill, Berkeley, CA.
- Csikszentmihalyi, M. (1975). *Beyond boredom and anxiety*. San Francisco: Jossey-Bass.
- DeVellis, R. F. (2006). Classical test theory. *Medical care*, 44(11), S50-S59.
- De Groot, A. (1966). Perception and memory versus thought: Some old ideas and recent findings. In B. Rleinmuntz (Ed.) *Problem solving*. NY: Wiley.
- Dorigo, M. (1992). *Optimization, learning and natural algorithms*, Ph. D. Thesis, Politecnico di Milano, Italy, EU.
- D'Zurilla, T. J., & Goldfried, M. R. (1971). Problem solving and behavior modification. *Journal of Abnormal Psychology*, 78(1), 107-126.
- Egan, D. E., & Schwartz, B. J. (1979). Chunking in recall of symbolic drawings. *Memory & Cognition*, 7(2), 149-158.
- Farhang, Y., Meybodi, M. R., & Hatamlou, A. R. (2008). Improving the

efficiency of forward checking algorithm for solving constraint.
Eighth International Conference on Intelligent Systems Design and Applications., 1, 240–245, 26–28.

Gardner, M. (2004). Theory of everything. *The New Criterion*, 23, 65.
Retrieved from
<http://www.newcriterion.com/articles.cfm/theory-of-everything-1136>.

Gathercole, S. E., & Baddeley, A. D. (1993). *Working memory and language*. Hillsdale, NJ, England: Lawrence Erlbaum Associates, Inc.

Glover, F. (1990). Tabu Search: A Tutorial. *The Practice of Mathematical Programming*, 20(4), 74–94.

Gobet, F. (1993). A computer model of chess memory. *15th Annual Meeting of the Cognitive Science Society*, 463–468. Hillsdale, NJ: Erlbaum Associates.

Gobet, F., & Simon, H. A. (1996). Templates in chess memory: A mechanism for recalling several boards. *Cognitive Psychology*, 31(1), 1–40.
Retrieved from <http://bura.brunel.ac.uk/handle/2438/1339>.

Gullikson, H. (1987). *Theory of mental tests*. Hillsdale, NJ: Lawrence Erlbaum Associates. (Originally published in 1950 by New York: John Wiley & Sons)

Hambleton, R. K., & Swaminathan, H. (1985). *Item response theory: Principles and applications*. Boston, MA: Kluwer-Nijhoff.

Hambleton, R. K., Swaminathan, H., & Rogers, H. J. (1991). *Fundamentals of item response theory*. Newbury Park, CA: SAGE.

Hart, P. E., Nilsson, N. J., & Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *Systems Science and Cybernetics*,

IEEE Transactions on, 4(2), 100-107.

Hayes, J. R. (1981). *The complete problem solver*. Philadelphia: Franklin Institute Press.

Hulin, C. L., Drasgow, F., & Parsons, C. K. (1983). *Item response theory: Application to psychological measurement*. Homewood, IL: Dow Jones-Irwin.

Inkala, A. (2007). *AI Escargot - The most difficult Sudoku puzzle*. Lulu.com Publisher, Finland.

Jeffries, R., Turner, A. A., Poison, P. G., & Atwood, M. (1981). The processes involved in designing software. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition*, 255-283. Hillsdale, NJ: Erlbaum.

Jones, S. K., Roach, P. A., & Perkins, S. (2008). Construction of heuristics for a search-based approach to solving Sudoku. *Research and Development in Intelligent Systems XXIV(3)*, 37-49.

Kim, S. (1999). The art of puzzle game design **【Notes from talks at the 1999 Game Developers Conference】**. Retrieved from <http://scottkim.com/thinkinggames/GDC99/index.html>

Kim, S. (2006). *What is a puzzle*. Retrieved from <http://www.scottkim.com/thinkinggames/whatisapuzzle/index.html>.

Kirkpatrick, S., Gelatt, C. D., Jr., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598), 671-680.

Klein, S. B. (1996). *Learning: Principles and applications*. New York: McGraw-Hill.

Knuth, D. E., & Moore, R. W. (1975). An analysis of alpha-beta pruning. *Artificial Intelligence*, 6, 293-326.

- Korf, R. E. (1985). Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 27(1), 97-109.
- Korf, R. E. (1996). *Artificial intelligence search algorithms*. Algorithms and Theory of Computation Handbook, CRC Press.
- Kramer, W. (2000). What makes a game good? *The Games Journal*. Retrieved from <http://www.thegamesjournal.com/articles/WhatMakesaGame.shtml>.
- Kumar, V. (1992). Algorithms for constraint-satisfaction problems: A Survey. *AI Magazine 1992*, 13(1), 32-44.
- Lewis, R. (2007). Metaheuristics can solve Sudoku puzzles. *Journal of Heuristics*, 12(4), 387-401.
- Lord, F. M. (1980). *Applications of item response theory to practical testing problems*. Hillsdale, NJ: Lawrence Erlbawn Associates.
- Lord, F. M., & Novick, M. R. (1968). *Statistical theories of mental test scores*. Reading, MA: Addison-Wesley.
- Lynce, I., & Ouaknine, J. (2006). Sudoku as a SAT problem. *Ninth International Symposium on Artificial Intelligence and Mathematics*, Fort Lauderdale.
- Mantere, T., & Koljonen, J. (2006). Solving and rating Sudoku puzzles with genetic algorithms. *New Developments in Artificial Intelligence and the Semantic Web Proceedings of the 12th Finnish Artificial Intelligence Conference*, 86-92.
- Moneta, G. B., & Csikszentmihalyi, M. (1996). The effect of perceived challenges and skills on the quality of subjective experience. *Journal of Personality*, 64(2), 275-310.
- Moraldo, H. H. (2001). Fun factor for game developers. *Game Design*.

Retrieved from

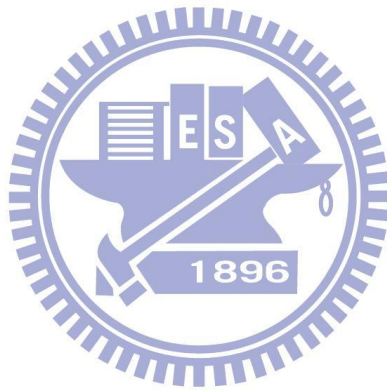
http://www.gamedev.net/page/resources/_/reference/103/general-game-design/222/fun-factor-for-game-developers-r1828.

- Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ., Prentice Hall.
- Newell, A., Shaw, J. C., Simon, H. A. (1958). Chess-playing programs and the problem of complexity. *IBM Journal of Research and Development*, 2(4), 320-335.
- Pang, S., Li, E., Song, T., & Zhang, P. (2010). Rating and generating Sudoku puzzles. *2010 Second International Workshop on Education Technology and Computer Science*, 3, 457-460.
- Pelanek, R. (2011). Difficulty rating of Sudoku puzzles by a computational model. *Twenty-Fourth International Florida Artificial Intelligence Research Society Conference*, 434-439.
- Reeson, C.G., Huang, K. C., Bayer, K.M., & Choueiry, B.Y. (2007). An interactive constraint-based approach to Sudoku. *Twenty-Second AAAI Conference on Artificial Intelligence*, 1976-1977.
- Reisel, S. P., Ainsworth, A. T., & Haviland, M. G. (2005). Item response theory, fundamentals, applications, and promise in psychological research. *Current Directions in Psychological Science*, 14(2), 95-101.
- Reisel, S. P., & Waller, N. G. (2009). Item response theory and clinical measurement. *Annual Review of Clinical Psychology*, 5, 27-48.
- Rolle, T. (2003). *Development of a multi-game engine*, Diploma Thesis, Friedrich-Schiller University Jena, Faculty of Mathematics and Computer Science, Jena.

- Schmittberger, R. W. (1992). *New rules for classic games*. John Wiley & Son, New York.
- Semeniuk, I. (2005). Stuck on you. *New Scientist*, 24(31), 45-47.
- Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27, 379-423, 623-656.
- Simonis, H. (2005). Sudoku as a constraint problem. *Proceedings of the 4th International Workshop on Modeling and Reformulating Constraint Satisfaction Problems*, 13-27.
- Stiny, G. & Gips, J. (1978). *Algorithmic aesthetics: Computer models for criticism and design in the arts*, University of California Press, Berkeley.
- Sweller, J. & Cooper, G. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction*, 2, 59-89.
- Thompson, J. M. (2000). Defining the abstract. *The Games Journal*. Retrieved from <http://www.thegamesjournal.com/articles/DefiningtheAbstract.shtml>.
- Weber, T. (2005). A SAT-based Sudoku solver. *Twelfth International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, 11-15.
- Xu, C., & Xu, W. (2009). The model and algorithm to estimate the difficulty level of Sudoku puzzles. *Journal of Mathematics Research*, 1(2), 43-46.
- Yato, T., & Seta, T. (2003). Complexity and completeness of finding another solution and its application to puzzles. *IEICE Transactions*

on Fundamentals of Electronics, Communications and Computer Science,
86(5), 1052–1060.

Zimmerman, E., & Salen, K. (2003). *Rules of play: Game design fundamentals.*
MIT Press, Cambridge.



附錄-直觀法概說

餘數法

餘數法就是確認空格應填數字的方法。首先，檢查空格的同一群組（同一行、同一列、同一九宮格）已出現的數字有多少，當出現的數字已有 8 個時，剩下的那一個數字就是這個空格的正解。這種尋找正解的方法，是一般人在未接受教學時，第一個最容易想到的解題技巧。以下就用一個的數獨題來做示範：

8		7	2		1		5	6
			4	6		1		
2					8	3	7	4
			1	8		4		
	6	8	7	4	9	2	3	5
		5		2	3			
7	5	3	9					8
		4		5	6			
6	8		3		2	5		9

在第一列第七行的空格，可以清楚地發現 1、2、3、4、5、6、7、8 已在同一行或同一列出現過，剩下數字 9 可以填入此空格中，這所使用的方法就是餘數法。餘數法在實際解題時可細分為唯一法、二餘法、三餘法、四餘法、數對摒除法、唯餘法、區塊數對唯餘法，如下：

1. 唯一法：數獨謎題中的某一個宮格因為所處的單元（列、行或宮）已填入 8 個數字時，那麼這個宮格所能填入的數字，就只剩下那個還沒出現過的數字。利用這個方式找到正解的方法就是唯一法。
2. 二餘法：找到一個看起來已填數字似乎較多的單元，例如下圖中的第 7 行。

	4	7	3	1	2	8		
8				7	9	2		
		9			8		7	4
		3		6	4	9		
2		5		8		7		
		8	2	9		3		
9	1		8			6		
		4	9	3				1
		6	7	2	1	4	9	

由上圖看到第七行，首先，開始點算本單元已出現過的數字：找不到數字 1、 $(2, 7)=2$ 、 $(6, 7)=3$ 、 $(9, 7)=4$ 、找不到數字 5、 $(7, 7)=6$ 、 $(5, 7)=7$ 、 $(4, 6)=8$ 、 $(4, 7)=9$ ，所以本單元僅剩數字 1 和數字 5 有可能填入。再來，開始點算 $(3, 7)$ 這個空格的其他群組數字，在此空格之同一列及同一九宮格中找不到數字 1 或 5，本格的可能填入的數字仍為兩個 (1、5)，所以無法確認何者為正解。因此找尋下一個空格 $(8, 7)$ ，開始點算 $(8, 7)$ 的其他群組數字，在此空格之同一列中找到 $(8, 9)=1$ ，所以本格的正解 $(8, 7)=5$ 。又因為數字 5 已經填入空格，所以 $(3, 7)$ 這個空格應填入的數字為 1。

使用唯一法時，只需要點算本單元的數字即可，不必管群組中其他兩個單元有什麼數字，所以得解率百分之百；使用二餘法時，要先點算本單元尚缺哪兩個數字，然後在群組中的另兩個單元去找，如果可以找到任何一個，就可以確認空格的正解，且一次可得到兩個空格的正解；使用三餘法時，要先點算本單元尚缺哪三個數字，然後在群組中的另兩個單元去找，如果可以找到任何兩個，就可以確認空格的正解。通常會再繼續使用二餘法的技巧嘗試找解，因為可省去點算的過程了；使用四餘法時，要先點算本單元尚缺哪四個數字，然後在群組中的另兩個單元去找，如果可以找到任何三個，就可以確認空格的正解。通常會再繼續使用三餘法的技巧嘗試找解，因為可省去點算的過程。

使用唯餘法時，首先，要先點算本單元尚缺哪些數字，然後再從群組中的另兩個單元去找，如果只有一個數字找不到，就可以確認空格的正解。

點算本單元的數字之後，記住哪些數字沒有出現，然後到群組中的另兩個單元去找，這樣的過程中，當然是需要記住的數字越少越好了，所以餘數法之運用通常是唯一法優先，接著是二餘法，三餘法、四餘法再次之，唯餘法僅在不得已的時候使用之。

3. 數對唯餘法，下面就以實例來說明數對唯餘法的使用：

6				3	2			5
	7				9		2	
	1	2	5				9	
1		7			4			2
	6		3		1		7	
4			6			9		1
	4				6	5	1	
	5	6			3		4	
8		1	9	4	5			7

如上圖示，這個數獨已找不到摒餘解了，所以只好使用餘數法，假設我們以列、行、宮的順序，直接跳用四餘法，則將是以下過程：

- 先搜尋列，一直到第 9 列才出現空格數為 4 以下的情形，但找不到三餘解，只能在 $(9, 2)$ 及 $(9, 8)$ 註記雙候選數。
- 再搜尋行，在第 2 行可使用四餘法，一樣找不到餘數解，只能在 $(1, 2)$ 註記雙候選數。
- 在第 6 行可使用二餘法，一樣找不到餘數解，只能在 $(3, 6)$ 及 $(6, 6)$ 註記雙候選數。
- 在第 8 行可使用四餘法，找到餘數解 $(1, 8)=8$ ，另在 $(9, 8)$ 可註記雙候選數，如下圖。

6	89			3	2		8	5
	7				9		2	
	1	2	5		78		9	
1		7			4			2
	6		3		1		7	
4			6		78	9		1
	4				6	5	1	
	5	6			3		4	
8	23	1	9	4	5		36	7

當我們把找到的四餘解 $(1, 8)=8$ 填入後，因為第一列已有數字 8 了，
 $(1, 2)$ 的雙候選數 8, 9 之數字 8 應被刪去，如下圖：

6	9			3	2		8	5
	7				9		2	
	1	2	5		78		9	
1		7			4			2
	6		3		1		7	
4			6		78	9		1
	4				6	5	1	
	5	6			3		4	
8	23	1	9	4	5		36	7

之前的餘數法點算中， $(1, 2)$ 的候選數只餘 8, 9，現在再刪去一個，這表示 $(1, 2)$ 只剩下數字 9 可填了，所以 $(1, 2)=9$ ，因為是在註記了雙候選數（數對）後再刪除其中一個而得到的解，故稱為數對唯餘解。

摒餘法

摒餘法就是幫未填的數字尋找應填位置的方法。利用一些方法，將未填數字在指定單元中不可能填入的空格一一摒除，如果可以摒除到只餘一個空格時，這個未填數字當然就是該空格的正解了。而進行摒除的方法有基礎摒除法、區塊摒除法、單元摒除法、矩形摒除法、數對摒除法、數偶摒除法等幾種方式，以下就舉最基礎且較常用的摒餘法做介紹。

1. 基礎摒除法

利用各種摒除的方法將指定單元中不可能填入的空格一一摒除，所得到的解叫做「摒餘解」。如果以某數對某宮摒除之後，發現某數在該宮只餘一個空格可填，稱之為「宮摒餘解」；如果以某數對某行摒除之後，發現某數在該行只餘一個空格可填，稱之為「行摒餘解」；如果以某數對某列摒除之後，發現某數在該列只餘一個空格可填，稱之為「列摒餘解」。

宮摒餘解的系統其尋找是由數字 1 開始一直到數字 9，週而復始，直到解完全題或無解時為止；每個數字又需從上左九宮格起，直到下右九宮格，週而復始，同樣要不斷重複到解完全題或無解時為止，以下用一個簡單的例子做示範。

		3		8			7	6
	6		4					1
9					2			
		9			1		3	
1				6				9
	7		5			4		
			3					8
8					7		9	
6	4			1		5		

從上圖中可以發現，對於數字 1 來說，其在第五列、第九列、第六行和第九行皆已填入，因此可以清楚的知道，對於最右邊中間那個九宮格（紅色方框）來說，數字 1 只剩下一個空格可以填入，也就是圖中藍色的空格，這就是宮摒除法。

行、列摒餘解的尋找和宮摒餘解的尋找一樣，列摒餘解的系統其尋找是由數字 1 開始一直到數字 9，週而復始，直到解完全題或無解時為止；每個數字又需從第 1 列起，直到第 9 列止，週而復始，同樣要不斷重複到解完全題或無解時為止。同理，行摒餘解的系統尋找也是一樣的作法。大部分的人都會十分習慣應用宮摒餘解的尋找，而完全忽略了行、列摒餘解的尋找。對某些題目而言或許可行，但對某些題目而言，不運用此二種法的話可能會找不到答案。附錄中有此兩種方法的詳細解說。

在下圖中，使用列摒餘解，首先可以看到第五列只剩下 3、7 沒有填入，所以 (5, 3) 和 (5, 5) 這兩個空格皆有可能填入數字 3、7。

		8	7	6	1			
		6	2				7	
	3	7	5	8		1		6
		2	6					9
4	9		8		5	2	6	1
6					2	7		
	6	9		2		3	5	
	5				6			
			9	5	7	6		

且由於第三行數字 7 已經存在，所以 (5, 3) 不可能填入數字 7，因此 (5, 5) 出現唯一解數字 7，而 (5, 3) 也由於數字 7 的填入出現唯一解數字 3。最後得到下面的答案，如下圖所示。

		8	7	6	1			
		6	2				7	
	3	7	5	8		1		6
		2	6					9
4	9	3	8	7	5	2	6	1
6					2	7		
	6	9		2		3	5	
	5				6			
			9	5	7	6		

使用列摒餘解，如下圖所示。首先可以看到第三行對於數字 7 來說，因為第二列、第七列已有數字 7 存在，所以 (2, 3)、(7, 3) 不可填入數字 7。

				6	1	3		
		X		3	7	4		6
6	3	8		2				
		3		5			6	
7	6			1				4
	4			9		8		
		X		7		1		5
		6	3	4				
		2	1	8		6		

再來，看到最左邊中間那個九宮格，因為數字 7 已經存在，所以 (5, 3) 和 (6, 3) 也不能填入數字 7。因此，可以發現到第三行出現唯一解的空格 (1, 3)，此空格 (藍色區塊) 可填入的數字為 7。

				6	1	3		
		X		3	7	4		6
6	3	8		2				
		3		5			6	
7	6	X		1				4
	4	X		9		8		
		X		7		1		5
		6	3	4				
		2	1	8		6		

直觀法的基石就是基礎摒除法，而基礎摒除法中最常用的又是宮摒餘解的尋找。有些人只有在所有數字的宮摒餘解尋找已觸礁時，才做行、列摒餘解的尋找；有些人則是在每一個數字的宮摒餘解尋找完畢後，先做行、列摒餘解的尋找，然後再進行下一個數字的摒除。

2. 區塊摒除法

宮摒餘解的系統尋找是由數字 1 開始一直到數字 9，週而復始，直到解完全題或無解時為止；每個數字又需從上左九宮格起，直到下右九宮格，週而復始，同樣要不斷重複到解完全題或無解時為止。

使用區塊摒除法，只要在宮摒餘解的系統尋找時，注意是否有區塊摒除的成立條件即可，當區塊摒除的條件具備了，就等於多了一個摒除線，找到解的機會自然多了一點，將感覺順手多了。

例如下圖中，如果不使用或不曾使用區塊摒除法，是找不到 1 的宮摒餘解的，但如果用上了區塊摒除法，將可找到四個數字 1 的填入位置。

	5	7		4			6	
			2				5	
	1	8	5			9		
				1				9
				9		8		
3		5						6
								4
9		4	6				2	
				5		1		

在圖中，先從數字 1 開始尋找宮摒餘解，當找到最左邊中間的九宮格時，由於(3, 2)、(4, 5)的摒除，將使得數字 1 可填入的位置只剩下 (5, 1) 及 (5, 3)，因為每一個九宮格都必須填入數字 1，既然最左邊中間的九宮格的數字 1 一定會填在 (5, 1) ~ (5, 3) 這個區塊，那表示包含這個區塊的第 5 列，其另兩個區塊就不能填入數字 1 了，因為同一列中只能有一個數字 1，所以可將第 5 列另兩個區塊填入數字 1 的可能性摒除。

	5	7		4			6	
			2				5	
	1	8	5			9		
				1				9
				9		8		
3		5						6
								4
9		4	6				2	
				5		1		

第 5 列的區塊摒除，配合 (4, 5) 及 (9, 7) 的基礎摒除，使得 (6, 8) 出現了最右邊中間九宮格的宮摒餘解，此空格 (藍色區塊) 應填入的數字為 1，如下圖所示。

	5	7		4			6	
			2				5	
	1	8	5			9		
				1				9
				9		8		
3		5						6
								4
9		4	6				2	
				5		1		

使用單元摒除法，只要在宮摒餘解的系統尋找時，注意是否有單元摒除的成立條件即可，當單元摒除的條件具備了，就等於多了兩個摒除線，找到解的機會自然多一點。以下就用一個簡單的例子示範單元摒除法：

在下圖中，由於 (6, 5) 的列摒除，使得數字 4 可填入中左九宮格的位置只剩下 (4, 2)、(5, 1) 及 (5, 2)，另外，由於 (9, 4) 的列摒除，使得數字 4 可填入下左九宮格的位置只剩下 (8, 1)、(7, 2) 及 (8, 2)，因為這 6 個宮格恰好集中在相同的兩行上。如果中左九宮格數字 4 填在第 1 行的 (5, 1)，因為第 1 行只能有一個數字 4，所以下左九宮格的數字 4 就只能填到 (7, 2) 或 (8, 2)；如果中左九宮格數字 4 填在第 2 行的 (4, 2) 或 (5, 2)，因為第 2 行只能有一個數字 4，所以下左九宮格的數字 4 就只能填到 (8, 1)。

不論哪一個狀況產生，第 1 行及第 2 行的數字 4 都只能填在 (4, 2)、(5, 1) 及 (5, 2)、(8, 1)、(7, 2) 及 (8, 2) 這 6 個位置中的其中兩個，不可能填到其它宮格去，且恰好佔去了第 1 行及第 2 行，所以可以將第 1 行及第 2 行其他宮格填入數字 4 的可能性摒除。於是在運用單元摒除了第 1 行及第 2

行後，再配合 (1, 6) 及 (2, 7) 的基礎列摒除，使得 (3, 3) 出現最左邊上方九宮格的宮摒餘解。

	1		2		4	5		
		2				4		9
	8			6	9	1		
1		9		5				
		3	6		2	7		
				4		6		1
		7	3	9			1	
3		1				9		
		5	4		1		3	

