Abalone 電腦遊戲推論引擎之策略研究與平台實作

鄞永傳、余啟仁、蔡憲忠、申開玄
稻江科技暨管理學院 動畫與遊戲軟體設計學系

摘要

Abalone 是一款歐美流行的主流棋藝類遊戲，由於規則簡單、變化無窮，所以很適合拿來當作開發電腦賽局程式的主題。本文介紹 Abalone 棋類遊戲的棋盤佈局與規則，研究其各項推論策略，並且實作一套 Abalone 電腦賽局競賽的遊戲平台。此遊戲平台涵蓋了三種執行模式：電腦和人對奕、人和人對奕、以及電腦和電腦對奕。利用這三個對奕執行模式的交互運作，可以驗證各種推論策略的優劣性及可行性。這個電腦賽局競賽的開發，不僅可以測試出最佳的推論策略，據以實作出可以參加奧林匹亞電腦賽局競賽的電腦賽局程式版本、也可當作研究人工智慧的測試平台、並且可以成為遊戲程式設計教學上的輔助工具。

關鍵詞: 電腦策略遊戲、人工智慧、推論引擎

The Study and Implementation of Abalone Computer Strategy Board Game

Yung-Chwan Yin, Chil-Zen Yu, Shien-Chung Tsai
and Kai-Shuan Shen
Department of Animation Design and Game Programming, TOKO University

Abstract

Abalone is a popular board game in America and Europe. Its simplicity in rule and versatility in process make it the ideal subject of developing the computer strategy board game. This paper introduces the rules and layouts of Abalone, studies several inference strategies, and implements a computer game platform by C++ language. This computer game platform contains three kinds of execution modes: computer versus people, people versus people, and computer versus computer. It would be very efficient to verify the feasibility of different game strategies by harnessing the computer versus computer mode. Utilizing proper game strategies, this computer game platform can serve as a contest version for the next Computer Olympiad tournament, a test platform for artificial intelligence, and a teaching tool kit for game programming course.

Keywords: Computer Strategy Game、Artificial Intelligence、Inference Engine

壹、 研究動機與目的

在人工智慧的研究領域上，電腦棋類競賽一向扮演著積極的角色，因為棋藝遊戲常常是規則簡易明確，在競賽的過程中棋盤局面的變化卻又有著相當程度的複雜度。以電腦棋藝遊戲來當作人工智慧研究的一個標的，是一種相當經濟可行的方式，而且又可避免去涉及到探索人類心智活動常常會引發的道德(ethical)層面之爭議[8]。歷年來，人工智慧在電腦棋藝方面的研究獲得了許多成果，除了在空間搜尋及推論演算法外，對於人類在認知方面的了解也有多少有一些收獲。

從1950年代數位電腦起步後，電腦和人的對奕就變成大家關心的議題。許許多多人工智慧的學者專家也把這個主題視為研究的重心[13]，期望能藉由這方面的研究來找到一些有效的演算法。經過多年的努力，在1989年IBM西洋棋電腦程式深藍首次擊敗了世界西洋棋冠軍 David Levy； 1997年深藍[9]又成功的擊敗當時的西洋棋世界冠軍 Kasparov。電腦程式超越人類專家的時代終於來臨。

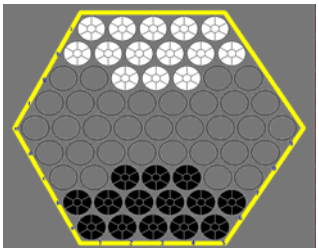
雖然電腦程式終於能超越棋藝專家，但是電腦所能扮演的思考方式卻是和人類的智慧運作方式大大不同，電腦策略遊戲程式的相對優勢是處理的速度，和近乎無限的儲存空間[14]。

在歐洲及美國， Abalone 是一款很流行的主流棋藝類遊戲，由於其規則簡單、容易上手，所以是一項適合各年齡層的人士作為休閒娛樂的活動項目。Abalone 遊戲也是奧林匹亞電腦賽局競賽的正式比賽項目之一[10]，可是以Abalone為主題的電腦賽局程式卻並不多。和西洋棋比較，有關 Abalone 賽局的策略研究也是相當少見的。

本研究的目的在於以 C++ 視窗程式開發一款 Abalone 電腦賽局競賽的遊戲平台，在此遊戲平台上實作電腦和人對奕、人和人對奕、以及電腦和電腦對奕的三種執行模式。由於能夠掌握程式架構，所以得以利用電腦和電腦對奕的執行模式，研究各種 Abalone 遊戲推理策略方式，並找尋最佳的賽局策略。

貳、 Abalone 棋盤佈局與規則

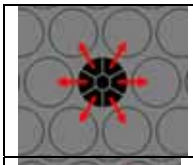
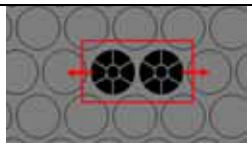
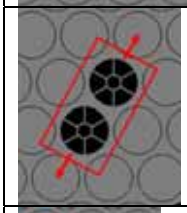
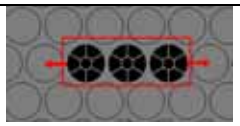
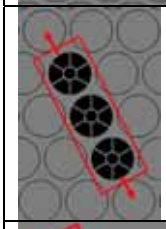
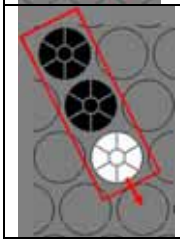
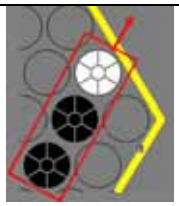
Abalone 棋盤為一個六角形，每一個周邊有五個彈珠球洞，所以全部棋盤共有61個球洞。比賽的兩方各自持有14顆黑色或白色的彈珠，持黑色彈珠者先下，可以沿線或側線一次推動至多三顆彈珠，雙方可以憑藉沿線彈珠數量的優勢把對方彈珠推出界限(2推1、或3推2)，率先把對方彈珠推出六顆者獲勝，棋盤開局盤面如圖表1 所示：



圖表 1　Abalone 棋盤開局

遊戲的規則很簡單，競賽的雙方輪流移動彈珠，並設法把對方的彈珠推出邊界外，移動彈珠的方式有兩種：沿線移動(inline)或側線移動(broadside)。

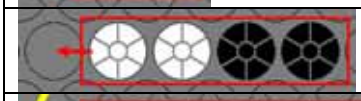
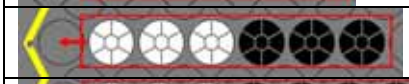
沿線移動(inline)：可一次移動至多三顆彈珠，只要移動方向的沿線上有空的球洞，或是對方的彈珠數量較少可被推動。常見的合法沿線移動如圖表2（皆以執黑棋為例）：

	移動一顆彈珠到任意六個方位 移動方向的沿線上有空的球洞		移動兩顆彈珠 移動方向的沿線上有空的球洞
	移動兩顆彈珠 移動方向的沿線上有空的球洞		移動三顆彈珠 移動方向的沿線上有空的球洞
	移動三顆彈珠 移動方向的沿線上有空的球洞		移動兩顆彈珠，並且推動對方的彈珠 移動方向的沿線上有空的球洞，並且對方的彈珠數目較少
	移動兩顆彈珠，並且推動對方的彈珠 移動方向的沿線上有空的球洞，並且對方的彈珠數目較少		移動兩顆彈珠，並且將對方推出界限 移動方向的沿線已達邊界，並且對方的彈珠數目較少

	移動三顆彈珠，並且推動對方的彈珠 移動方向的沿線上有空的球洞		移動三顆彈珠，並且將對一顆彈珠推出界限 移動方向的沿線已達邊界，並且對方的彈珠數目較少

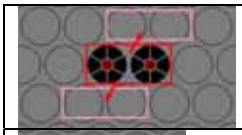
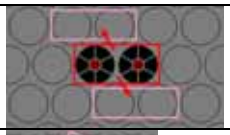
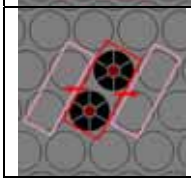
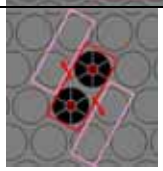
圖表 2 合法的沿線移動(inline)

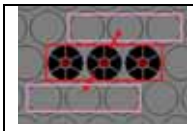
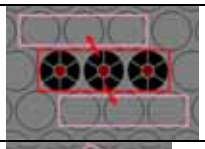
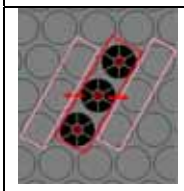
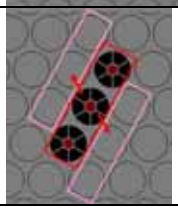
沿線移動可以趁勢推動對方的彈珠，先決的條件是要取得彈珠數量上的優勢，例如兩顆黑球推動一顆白球、三顆黑球推動一顆白球、三顆黑球推動兩顆白球等。但是三顆推動兩顆已經是上限，任何一方是不能推動己方的四顆彈珠，圖表3 列出非法的沿線移動：

	一顆黑彈珠嘗試推動一顆白彈珠，未取得數量優勢
	兩顆黑彈珠嘗試推動兩顆白彈珠，未取得數量優勢
	三顆黑彈珠嘗試推動三顆白彈珠，未取得數量優勢
	四顆黑彈珠嘗試推動三顆白彈珠，取得數量優勢，但四顆推動三顆是非法的移動

圖表 3 非法的沿線移動(inline)

側線移動(broadside)：側線移動不能推動對方的彈珠，只要移動的側線方向有完整的空球洞就可以一次側線移動二至三顆彈珠。圖表4 列出常見的合法側線移動如下：

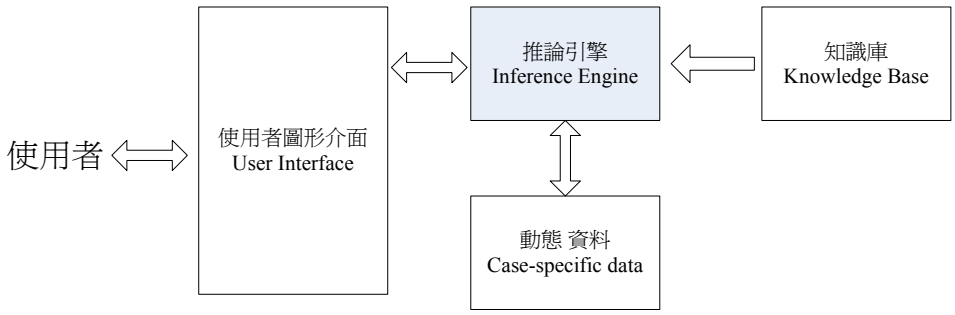
	兩顆黑彈珠側線移動，移動方向有完整的空球洞		兩顆黑彈珠側線移動，移動方向有完整的空球洞
	兩顆黑彈珠側線移動，移動方向有完整的空球洞		兩顆黑彈珠側線移動，移動方向有完整的空球洞

	三顆黑彈珠側線移動，移動方向有完整的空球洞		三顆黑彈珠側線移動，移動方向有完整的空球洞
	三顆黑彈珠側線移動，移動方向有完整的空球洞		三顆黑彈珠側線移動，移動方向有完整的空球洞

圖表 4　合法的側線移動(broadside)

參、　平台實作架構

本文實作的 Abalone 電腦賽局競賽的遊戲平台開發架構完全依照 Brad J. Cox 的軟體IC觀念[1]，以 C++ 視窗程式語言開發。在程式開發中，先以物件導向的方式建構多個相關類別元件，再搭配固定的視窗程式流程架構而完成這個遊戲競賽平台。使用軟體類別元件的架構可以簡化遊戲程式開發的工作、縮小開發時程，因為類別元件已經將重要的遊戲功能包裝成一組可再利用的 C++ 類別[7]，並且已經將複雜以及困難的功能包裝成簡單易用的介面[2]，因此在此架構下開發互動式遊戲程式將可以收到事半功倍的效果。本研究的遊戲平台開發架構如圖表5 所示：



圖表 5　Abalone 電腦賽局競賽的遊戲平台開發架構

1.使用者圖形介面：這是本遊戲平台的外觀，採用直覺式的視窗圖形介面設計，讓使用者可以直覺的操作。使用者以滑鼠點選競賽模式：電腦和人對奕、人和人對奕、或者電腦和電腦對奕模式，並且開啟新局。在對奕過程

亦可使用滑鼠操作彈珠的選取或移動企圖。

2.推論引擎：利用知識庫中的內建事實、規則、策略及賽局的動態資料來進行推論，尋找最有利的策略。本遊戲平台實作各項對奕策略，以較佳的策略來完成電腦方面的推論。

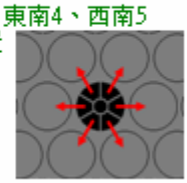
3.動態資料：儲存目前所能搜集到的賽局動態資料，包括對方佈局、我方佈局、優劣點數等資料。在對奕過程中這部份的資料是隨時會變動的。

4.知識庫：儲存 Abalone 電腦賽局競賽的規則，以及Abalone推論策略等靜態資料。

除了上述的開發架構外，實作一個電腦策略遊戲程式平台亦必須要考量三個部分：如何使用有效率的內部資料結構來表現棋盤佈局、推論引擎的運作方式、以及如何評估與決定最佳的佈局。

由於每一顆彈珠都可以朝六個方向移動，所以每一個彈珠球洞的內部資料結構必須要能記錄著六個方向的相鄰球洞編號。同樣的，當彈珠往某個方向移動時，內部資料結構也必須能夠記載著進攻方式與預測得分值，以作為評估及選定的依據。圖表6 即為每一個彈珠球洞的資料結構實作：

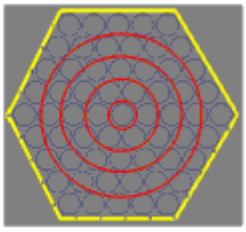
```
struct Ball_Struct
{
    short x, y;   // 球洞的座標
    short A[6];   // 相鄰球洞的號碼 西0、西北1、東北2、東3、東南4、西南5
    short status; // 球洞的目前的狀況 1 白球、2 黑球、3 空置
    short undo_status;  // 預留空間 作爲 回復的狀況
    bool  selected;     // 此球洞是否已被點選
    short G[6];         // 球往六個方向移動的得分預估值
    short ActionNo[6];  // 球往六個方向的 進攻方式
};
```



圖表 6 每一個彈珠球洞的內部資料結構

另外一個重要的資料結構就是每一個球洞的戰略位置。Abalone 電腦賽局競賽的目標就是要把對方的彈珠推出界外，所以愈靠近中心點的彈珠就愈安全。在實作程式內部將這樣的利害關係用一個陣列來表示 (圖表 7)：

```
// 球位置的重量
short Ball_Weight[61] = {      1,1,1,1,1,
                             1,2,2,2,2,1,
                           1,2,3,3,3,2,1,
                         1,2,3,4,4,3,2,1,
                       1,2,3,4,5,4,3,2,1,
                         1,2,3,4,4,3,2,1,
                           1,2,3,3,3,2,1,
                             1,2,2,2,2,1,
                               1,1,1,1,1      };
```



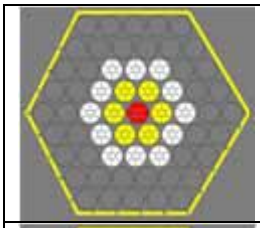
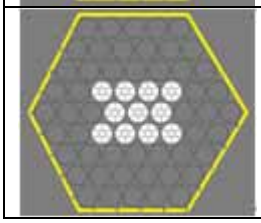
圖表 7 六十一個彈珠球洞的戰略位置結構

在遊戲進行的過程中，Abalone 推論引擎會嘗試定義出一棵遊戲樹，以遊戲狀態為一個節點 (node)，往下的每一個子節點代表移動一步之後的狀態。推論引擎儘可能的從遊戲樹的目前位置，推算到最多步的走法，並且以評估函數評估每個遊戲位置對兩方玩家的影響。推論引擎在推論的過程中會產生大量的搜尋空間(search spaces)，這些搜尋空間通常都是相當龐大及複雜，以致於需要一些技術來協助減少這些空間，這些相關的技術就稱為賽局策略。下一節探討 Abalone 的賽局策略。

肆、 Abalone 賽局策略研究

許多的人工智慧研究顯示，一般的棋藝專家往往能把棋盤佈局分解成一小塊一小塊可供辨識的小佈局，這些小佈局就是從他們的經驗累積成的一組又一組的模式(pattern)。在比賽的過程中，棋藝專家不停的辨識這些模式 (pattern)，並且隨著賽局的演變，刻意的操控這些小佈局，讓自己能夠處在有利的局面。一般的資淺的玩家，因為經驗的小模式(pattern)相當有限，所以會處於相對不利的局面。

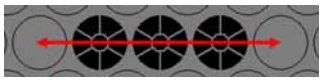
Abalone 的賽局策略同樣的以小佈局為單位，可分為佈局與中盤策略。佈局策略以佔領棋盤中央位置為主，以及儘量將己方彈珠集結成堅固的群集 (cluster)。如圖表 8：

	佈局策略以佔領棋盤中央位置為主 如左圖所示，棋盤中心點是重要的戰略位置，賽局策略應以佔據愈多的棋盤中央位置為主要考量
	佈局策略以集結成堅固的群集 (cluster) 為重要考量 如左圖所示，集結成堅固的群集，無論從任何方向推算都是堅固的三子連線隊形

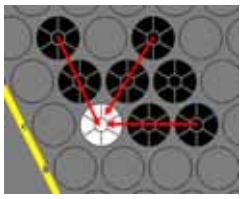
圖表 8　Abalone 佈局策略

隨著棋賽的進行，因為己方攻擊或對手進攻，兩方的隊形都將逐漸被對方切割成數塊，此時中盤策略逐漸影響盤局，包括：

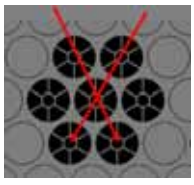
1.建立三子連線：Abalone 最強的沿線優勢是三子連線，所以中盤時儘可能保持三顆彈珠連線，單顆落單的彈珠是最易受到功擊的對象。



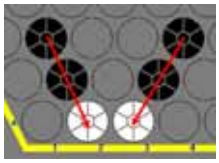
2.誘敵進入己方陣地，再三面圍攻：三面圍攻可取得最絕對的優勢。



3.呈菱形以應付來自不同方向的攻擊：菱形隊形可以抵擋正面及側面的功勢。



4.雙路攻擊兩子：雙路攻擊可讓只能對方擇一閃躲。

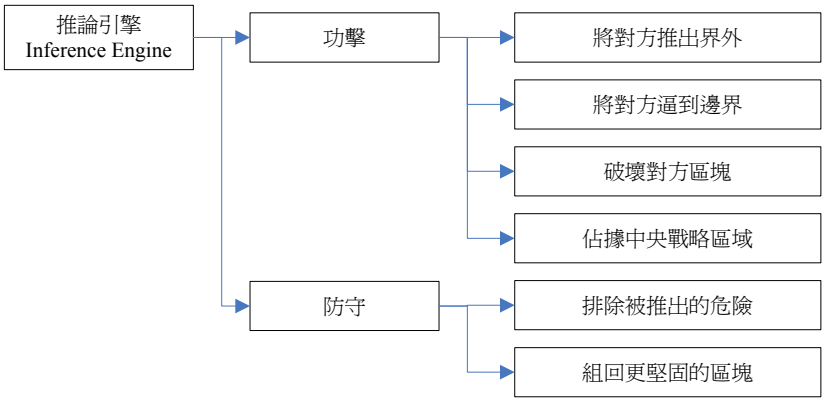


5.局勢有利或子數較多時，可以一子換一子來降低盤面複雜度：棋賽的規則是先把對方的彈珠推出六子者獲勝，所以當己方處於優勢時可用以子易子的方式，儘早結束棋局。

6.將對方棋子分散：如有可能以，儘量將對方的彈珠打散，再各個突破。

7.在劣勢時(子數落後三子以上)分散棋子以求和局。

根據上述的賽局策略，abalone 推論引擎可以劃分為兩大類：進取導向與保守導向。進取導向以功擊為主；保守導向以防守為主。如圖表 9 所示：



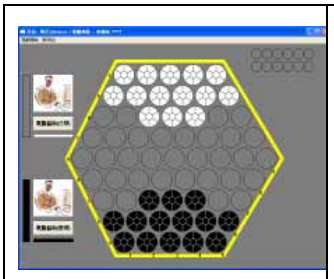
圖表 9　abalone 推論引擎

伍、　結論

Abalone棋藝競賽是一種零和遊戲 (zero-sum game)，競賽雙方的獲利都等於對手的損失，競賽的過程不斷的在選擇最佳的盤勢棋路，以掌控較佳的局面。

本文研究Abalone推論引擎的各項推論策略，並且實作完成一套Abalone 電腦賽局競賽的遊戲平台。如圖表 10、圖表 11、圖表 12 所示，透過遊戲平台的三種執行模式的交互運作：人和人對奕模式、電腦和人對

奕模式、電腦和電腦對奕模式，可以很容易的驗證各種推論策略的可行性，並且測試各種推論策略混合運用的最佳方式。這個遊戲平台的實作將可參加奧林匹亞電腦賽局競賽、提供人工智慧研究的測試平台、也可以成為遊戲程式設計教學上的輔助工具。



圖表 10 人和人對奕模式

圖表 11 電腦和人對奕模式

圖表 12 電腦和電腦對奕模式

參考文獻：

Brad J. Cox. (1986). Object Oriented Programming An Evolutionary Approach. Productivity Products International, Inc.

Boer, James. (2000). Object-Oriented Programming and Design Techniques, Game Programming Gems, Charles River Media, Inc., 8 ~ 19.

Caitlin Kelleher and Randy Pausch. (2005, June). Lowering the Barriers to Programming: A taxonomy of Programming Environments and Languages for Novice Programmers. ACM Computing Surveys. 37, No. 2.

Campbell, Murray, (1997) "An Enjoyable Game: How Hal Plays Chess," in Stork, David (ed), "Hal's Legacy," MIT Press.

David Levy and Monty Newborn (1991). How Computers Play Chess. Computer Science Press. ISBN 0-7167-8121-2.

Frey, Peter, (1997) "Chess Skill in Man and Machine," Springer-Verlag.

Gamma, E., et al. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley Longman, Inc.

Herik, H. J. van den, Uiterwijk, J.W.H.M., and Rijswijck, J.V. (2002). Games solved: Now and in the future. Artificial Intelligence, Vol. 134, pp. 277-311. ISSN 0004-3702.

Hsu, Feng-hsiung, (2002) "Behind Deep Blue," Princeton University Press.

International Computer Games Association, 11th Computer Olympiad, (2006) http://www.cs.unimaas.nl/olympiad.

Levy, David, (1976) "Chess and Computers," Computer Science Press.

Levy, David, (1980) "More Chess and Computers," Computer Science Press.

Levy, David and Newborn, Monty, (1990) "How Computers Play Chess," Computer Science Press.

Newborn, Monroe, (2003). "Deep Blue: An AI Milestone," Springer-Verlag.