黑白棋自我學習的改進策略

賴泳伶 國立嘉義大學資訊工程學系

張孝凡 國立嘉義大學資訊工程學系

摘要

本文提出一個黑白棋程式的自我學習策略。我們首先以動態程序（dynamic programming）的方式改進 alpha-beta search 的效率，再利用動態搜尋時間的方式進行自我對奕，作開局資料庫與圖樣辨識（pattern recognize）的學習，讓程式的自我學習能夠在微幅的失真度下，大幅提昇學習速度。

經過調整圖樣辨識區域與測試後，證實我們的自我學習策略能在小幅度的失真下，顯著提昇自我學習效率。這個策略經過修正後，也可以應用在其它棋戲程式上，作爲原演算法的輔助，提昇棋戲程式的棋力。

關鍵詞：黑白棋、機器學習、自我學習、人工智慧

1. 研究動機與目的

電腦棋戲程式運用人工智慧演算法，經過各方先進多年的研究與改進，在部份的棋戲中，已設計出許多遠勝於人腦的程式系統。黑白棋程式的演進，就屬其中之一。在黑白棋的規則中，限制落子後必須能翻對手棋子，才是合法棋步[5]，因此每一手棋的合法棋步侷限在 1（即使是 pass，也是一種合法棋步）至 15 步上下(超過 15 較少出現)，平均約在 8 步左右。這樣的分支度相較於其它的棋戲（如圍棋平均超過 200 步、象棋概估至少超過 50 步…），可說遠低得多。因此一般以 min-max 搜尋法，配合 alpha-beta cut 來設計的黑白棋程式，即使審局函數（evaluation function）的設計並不十分講究，在現今電腦的高速執行下，經過多層的搜尋後，均能有良好的棋力表現。此外由於黑白棋每下完一手棋，必須翻轉對手被包夾的棋子，因此棋面上最多可產生 20 顆棋子的變化（自己下一顆棋子+

翻轉對手 19 顆棋子），如此複雜的盤面演化，使得人腦難以推算多步以後的盤面，更難以分析多步以後的利弊得失。因此，若以目前世界頂尖的黑白棋程式，與人類黑白棋冠軍對奕，結果可說程式將會獲得壓倒性的勝利。

從搜尋的角度分析，黑白棋從開局到終局，平均雙方共落子 60 手，若要搜尋出第一步棋的最佳解，其複雜度爲 $O(n^{60})$，依前述平均每一手分支度爲 8 代入 n，則約爲 $1.5*10^{54}$。這樣的高複雜度告訴我們，程式無法在短時間內，以搜尋的方式，完成最佳解的搜尋（事實上即使搜尋幾百萬世紀也搜尋不完）。再從盤面變化的組合來分析，黑白棋使用的是 8*8 的棋盤，每一格可以有空點、黑棋與白棋三種變化，因此盤面最多有 3^{64}（約爲 $3.4*10^{30}$）種組合，即使考慮對稱性，並將部份不可能出現（不合理）的盤面扣除掉，其空間複雜度仍會是個天文數字。因此在黑白棋的世界裏，雖然人腦的棋力早已遠不如電腦，但在先手或後手必勝還沒有被證明出來之前，程式的棋力，仍舊有再進步的空間。

爲追求程式棋力的精進，最直接的做法，就是改進審局函數的精確度，與增加搜尋深度[2]。這二種做法是魚與熊掌，無法得兼。舉例來說，能佔得棋盤的角與邊的棋步，可能是比較好的棋步；或是能迫使對方無棋可下的棋步，也可能是較佳的選擇。於是藉由在審局函數中，加入這些分析，使局勢判斷更爲精確。但是這些被額外加入的分析，勢必會增加審局函數的時間複雜度。也就是說，要達到原有的搜尋深度，必須耗費更多的搜尋時間；或是在相同的搜尋時限內，付出搜尋深度減少的代價。

使程式棋力提昇的另一種做法，則是製做開局資料庫系統。這個做法在象棋[1]與在黑白棋[3]的應用上都已被提出，且已知在象棋上有良好的表現。這類資料庫系統的目的，在大量蒐集棋譜，藉由資料的累積，統計出較好的著手，作爲開局時的輔助策略。由於這種做法必須取得大量的對奕記錄，因此多在網路上取得，或設計成主從式的模組，讓網路上的棋友進行對奕。然而此法受限於所取得

的棋譜必須是”高手”的對奕結果，否則即使累積了大量中低棋力者的對奕記錄，參考與運用後的結果，恐怕也不會有較佳的表現。基於這個限制，我們不禁要質疑，在黑白棋中既然人腦棋力遠不如電腦程式，那麼大量蒐集人腦對奕後的棋譜，對程式棋力的提昇，是不是真的有幫助？

目前已有許多黑白棋的愛好者，將一些頂級的黑白棋程式自我對奕後的棋譜，與近年來多次世界比賽的對奕棋譜，以 wthor 資料庫的格式[4]儲存，並在網路上公佈提供棋友下載[6]。這些公開的資料庫的確是一個很好的學習資料來源，但當電腦黑白棋程式比賽時，各程式系統都採用了這些開局資料庫，在相同的立足點之下，要能夠展現出更好的棋力，除了在審局函數的設計上力求精確，和改進搜尋效率外，就是要加強自我學習的能力，才能使棋力更加登峰造極。

現在已有許多頂級的黑白棋程式，如 WZebra、Herakles、Forest 等等，都已運用了自我學習的技術，並多方蒐集 wthor 資料庫的棋譜，轉換爲自己的學習經驗。這些世界知名的程式除了具有低複雜度、高精確度的審局函數外，還大幅改良了搜尋效率，使程式本身就具備了高水準的棋力表現。然而這些程式在自我學習的應用上，僅單純地記錄對奕棋譜，用來作爲開局資料庫，而沒有在這方面多做改進與利用，著實可惜。

本文的目的，在提出一個增進黑白棋自我學習效率的策略。程式以 alpha-beta search 爲設計主軸，利用動態程序的技術強化搜尋效率，再令程式進行自我對奕，累積棋譜，進行棋路修正。爲了強化自我學習的效率，我們在自我學習的過程中，加入了動態搜尋時間的設定，與圖樣辨識 （pattern recognition）的學習，讓程式的學習能夠在微幅的失真度下，最大增加四千萬倍的學習速度。如此”以子之矛、攻子之盾”地反覆進行對奕與學習，能使程式更快找出落敗的關鍵棋步，從而補強原搜尋法的弱點，使程式棋力獲得提昇。

2. 方法

我們的黑白棋程式，是以 Borland C++ builder 6 設計完成，並在 MS Windows 系列平台上執行。方法分析如下：

2.1. 棋力基礎：

搜尋演算法是利用 Alpha-Beta cut，在指定的搜尋時限內進行搜尋。由於黑白棋的搜尋複雜度遠大於盤面的空間複雜度（如前一小節中所分析），因此經由不一樣的搜尋路徑，卻產生相同盤面的情形勢必會經常發生。故爲了增進搜尋效率，我們利用動態程序的技術，在搜尋過程中記錄曾經出現的盤面，並儲存該盤面被搜尋展開後所傳回的評估值。此後當搜尋到曾經展開過的盤面，就可以直接套用先前的評估值，以省去重複向下展開所需花費的時間。

2.2. 自我學習：

如同其它知名程式的開局資料庫，我們的程式也是將自我對奕的棋譜，儲存作爲日後對奕的開局知識庫。所不同的是，我們以 2 秒爲基礎搜尋時限，進行自我對奕，並記錄自我對奕戰績，再動態調整搜尋時限。調整方式如下：

$$\begin{cases} T_W = 2, T_B = 2 + (S_B - S_W)/2 & if \quad S_W > S_B \\ T_B = 2, T_W = 2 + (S_W - S_B)/2 & if \quad S_W < S_B \\ T_W = T_B = 2 & if \quad S_W = S_B \end{cases}$$

其中 T_W 與 T_B 分別代表白棋方與黑棋方，每下一步棋所允許使用的思考時限，而 S_W 與 S_B 則分別代表白棋方與黑棋方自我對奕後的勝場數。

動態調整搜尋時限有三大優點，第一個是在於逐漸增加搜尋的時間，相對地很有可能使搜尋深度增加，棋路的正確性也就大有可能因此提高。這會使得落敗的一方，有較大的機會能以較高的搜尋深度，找到真正導致落敗的較差棋步，作出正確的修正。

第二個優點，在於搜尋未能觸及終局前，盤面的優劣都難以正確判定，因此修正結果也有可能是錯誤的。在這樣的情形下，錯誤的修正導致再度落敗的機會就會加大，於是在再度落敗後，程式會自動再增加搜尋時間，就有機會在更高的

搜尋深度下，將錯誤的修正重新更正回來。

第三個優點，在於黑白棋的對奕中，前幾步的優劣事實上是很難下定論的。若是任由程式依固定的演算法進行對奕，那麼對奕結果有可能黑白勝負場次數會產生很大的落差。這會導致可能在開局時最初幾步棋的判斷上，認定某些棋步是最佳棋步，而每次都作出相同的選擇；或是認定某些棋步是最差棋步，就再也不走出那些棋步。結果當程式以這個自我對奕的資料庫，去跟別的對手對奕時，對手下出程式從未見過的棋形的機會就會大幅增加，自我對奕的效果也就相對地大打折扣。這個問題雖然可以加入一些隨機（random）選擇的設定來避免，但是與其讓程式去隨機作決定，不如一開始就以動態搜尋時間進行自我對奕，來平衡開局時的勝負場次數，拓展自我學習的廣度，會較為合理。

2.3. 圖樣辨識學習：

由於黑白棋的空間複雜度甚高，因此為了加速自我學習的效果，我們在程式中加入了圖樣辨識的學習。我們觀察了大量的對奕棋譜後，分析出對棋局影響最大的區塊，和部份較不具影響力的區塊。如圖 1 中，紅色區塊內，最對棋局最不具影響力的區塊，藍色區塊則是具小幅度影響力的區塊，其餘部份，才是對棋局佔有決定性影響的區塊。

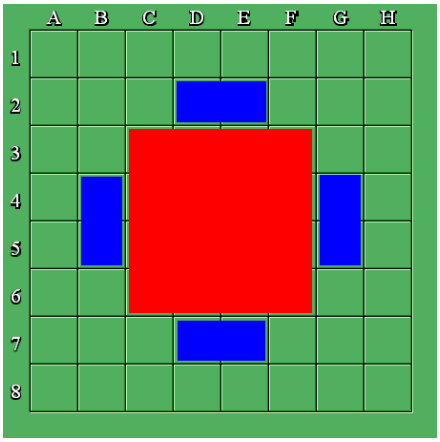


圖 1：影響力區塊示意圖

於是我們將整個棋盤中，扣除掉紅色區塊以外的區域，視為一個樣本（pattern），讓程式在自我對奕的過程中，同時進行圖樣辨識的學習。這樣的學習行為與自我學習很類似，一樣是將累積下來的對奕結果，作為樣本的評估值，

加入到審局函數中，視爲盤面的評估之一。

圖樣辨識的運用不同於棋譜比對的地方，在於棋譜比對是將搜尋第一層的棋形比對棋譜權重後，將權重套用於搜尋展開該棋形後的所有棋形上；而圖樣辨識則是在搜尋到限制最底層時，才比對圖樣的權重。（如圖 2）

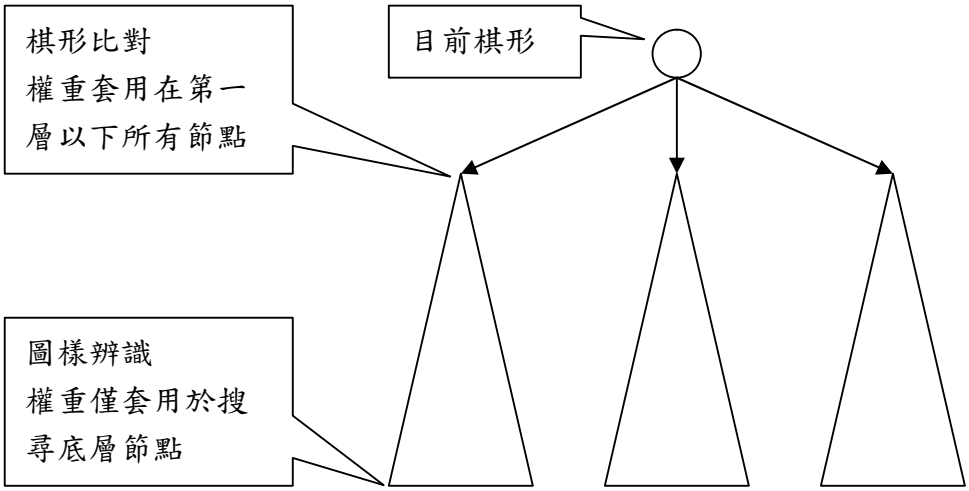


圖 2：棋形與圖樣辨識在搜尋時，套用點差異處示意圖

我們不把棋形比對應用在搜尋底層的原因，在於自我學習即使給予數十年，甚至上百年的長時間學習後，在對奕中也可以在開局後二十手棋之內，就出現不曾見過的棋形。若再將棋形比對放在搜尋底層，那麼扣除掉搜尋深度，可能在開局後十步棋內，就會搜尋到不曾見過的棋形。這樣的應用對於平均要下 60 手才終局的黑白棋而言，棋形比對的效果若只能運用在整局棋的前四分之一，似乎就顯得太微乎其微。因此我們將它應用在搜尋的第一層，較能作更深入的應用。

反觀圖樣辨識，由於忽略了中央 16 格的變化，因此每記錄一個圖樣，就等於是記錄了 3^{16} 種不同的棋形，若再考慮對稱性，並扣除掉少部份不可能出現的棋形，記錄一個圖樣，也相當於記錄近 4 千萬個棋形的學習效果。由於圖樣辨識忽略了中央 16 格的變化，因此在少部份的盤面評估上，會有些微的誤差，因此我們將圖樣辨識應用在搜尋最底層，作爲審局函數的一個權值，並以略低於棋形比對的權重作評估，就能利用圖樣的累積，大幅提昇學習效果，改進學習效率。

3. 結果與討論

在 Alpha-Beta search 中，運用動態程序的技術後，我們設定每一步的搜尋時限爲 2 秒，進行自我對奕與效果評估，成果如表 1。

表 1：運用動態程序（dynamic programming）加速的搜尋效果

	第 7 手	第 14 手	第 21 手	第 28 手	第 35 手	第 42 手	第 49 手
Alpha-beta search 深度	5	5	4	5	5	5	9
dynamic programming 加速後搜尋深度	6	5	5	6	6	7	12
平均 hit rate (%)	7.4	6.6	5.3	7.1	9.4	12.8	19.6

從表 1 中可以看得出來，應用動態程序法加速搜尋後，在開局與終局的搜尋效果上，有明顯的改進，因此能有效的節省下多餘的展開時間，作更深入的搜尋。尤其在接近終局附近的搜尋中，平均有約二成的棋形是搜尋過的，這對終局前的完整搜尋（perfect searching），可說具有相當大的助益。附帶一提的是，由於發現搜尋過的棋形時，該棋形就會直接套用評估值，以下也就不會再多作展開，因此實際上重複的棋形，在比例上會比表 1 中的正確率（hit ratio）大得多。

在動態調整搜尋時間方面的測試，經過 4 星期的自我學習後，能有效平衡開局時前幾步的勝負比例，擴展初期的分支度。表 2 爲每學習 4 天，與知名程式 WZebra 對奕時，學習資料能運用到的最深層數比較。

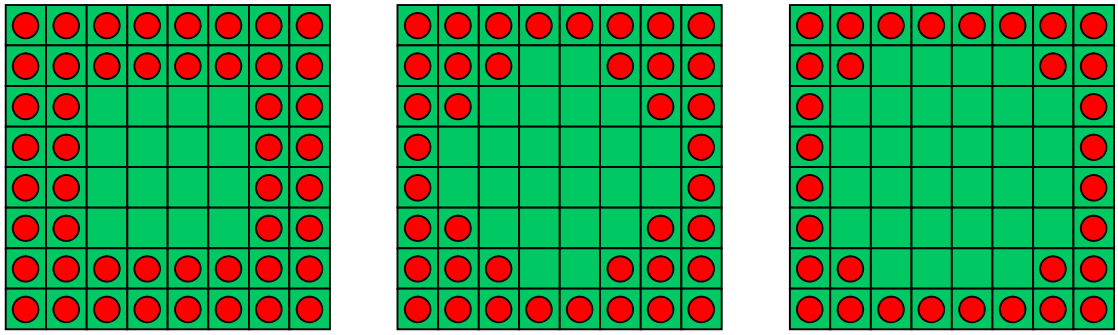
表 2：與 WZebra 對奕時開局資料運用深度

	第 4 天	第 8 天	第 12 天	第 16 天	第 20 天	第 24 天	第 28 天
自我對奕思考時限固定 3 秒	6	7	7	6	8	8	9
動態調整自我對奕思考時限	7	8	9	10	10	11	11
已完成廣度學習的層數	5	5	5	6	6	6	6

表 2 中顯示，利用動態調整自我對奕思考時間的方式，可以使自我對奕的廣度平均而穩定地增加，同時由於動態調整思考時限，使程式在面對增加思考時間

的對手時，就等於是在和棋力較高的對手下棋，進行學習，也就能汲取更好的學習經驗，使棋力提昇。

在圖樣辨識方面的學習，我們以圖 3 中的三種圖樣區塊，進行分析與測試。目的在確認出失真度最低、可忽略區塊最大的圖樣區塊。



(a) (b) (c)

圖 3：三種不同的圖樣辨識區塊

圖 3 中紅色圓點表示被選用的辨識區域，其餘空點爲被忽略變化的區域。我們從網路上隨機選取 40 個棋譜，再將棋譜分別以圖 3 中的 3 種不同的辨識區域抽取出來，最後在被忽略區域中，以隨機產生的 100 種盤面組合填入，分別進行完整搜尋。其中正確率的計算，是取將前述的 100 種盤面組合中，每一個盤面的完整搜尋後，黑、白雙方勝負結果一致的比率，而失真度則是正確率的補值。取平均值後，失真率測試結果列於表 3。

表 3；辨識區塊失真率測試結果。

	第 43 手	第 46 手	第 49 手	第 52 手	第 55 手
區塊 a	30%	27%	26%	19%	15%
區塊 b	42%	40%	41%	33%	19%
區塊 c	47%	43%	44%	42%	37%

從表 3 中可以清楚地看出來，區塊 a 的失真率最低，且越接近終局，正確率就越高。因此，我們將圖樣辨識放在搜尋的底層，可在越接近終局時，獲得良好的評估效果。

最後，我們將另一知名黑白棋程式 Deep green 設定搜尋深度爲六層，審局函

數的難度設定爲最高，並將我們的程式設定搜尋時限爲 2 秒，再進行對奕測試。我們發現 Deep green 除了在開局前 6~8 手，與接近終局前 10~12 手棋，搜尋能在 1~2 秒內完成，其餘中局的思考時間都超過 2 秒，最多還會達到 8~10 秒，平均搜尋時間約需 4 秒。而我們的程式設定固定搜尋時間 2 秒，在開局時搜尋深度約在 8~10 層，中局降至 5~7 層，終局前 12 手可完成完整搜尋。這表示我們的程式能以平均較短的搜尋時間，達到比 Deep green 更高的搜尋深度，顯示我們的程式搜尋效率比 Deep green 佳。在學習效果評估方面，初期還沒有進行自我學習前，我們的程式就能在 50 次先、後手各半的對奕中，取得 35 次的勝績。在進行 2 星期的自我學習後，再進行對奕測試，我們的程式在 50 次的對奕中，取得了 42 次的勝績，顯示自我學習已使程式的棋力，獲得了明顯的進步。另外，在對世界級的程式 Wzebra 的對奕測試中，我們的程式初期還沒有進行自我學習前，50 次的先、後手各半的對奕中，我們取得 12 次的勝績；經過 2 星期的學習後，我們的程式就能進步到取得 17 次的勝績。這個結果顯示，我們的程式在評估函數的設計上不如 Wzebra 精確，以致基礎棋力不如 Wzebra，但在進行學習後，同樣能獲得顯著的棋力提昇。

4. 結論

我們利用動態調整搜尋時間的方式，與圖樣辨識的技術，研擬出一個更有效率的黑白棋自我學習策略。經過測試後，我們證實利用這樣的策略，能使整體自我學習效率大幅提昇。未來我們將再調整審局函數的精確性，使程式的基礎棋力再提昇，將有希望挑戰世界級的 Wzebra 等黑白棋程式。對於其它的棋戲程式，可應用我們的自我學習策略，配合不同棋戲規則，作適度的修正，能強化其原有演算法的弱點，進而獲得更高的棋力。

參考文獻

[1] 許舜欽，「電腦象棋開局知識庫系統之研製」，*台大工程學刊*，第 53 期，Oct. 1991，pp.75-86

[2] 許舜欽，「電腦對局的搜尋技巧」，*台大工程學刊*，第 51 期，Feb. 1991，pp. 17-31

[3] 陳志昌、徐明煒：「Othello 開局資料庫系統之研製」。台灣大學資訊工程研究所，TAAI2002。

[4] par Sylvain Quin, *"Format de la base Wthor"*, available at http://www.ffothello.org/info/Format_WThor.pdf

[5] 黑白棋規則，available at http://www.othello.org.hk/

[6] 近年 Othello 棋戲程式比賽記錄之 Wthor 資料庫格式儲存檔案，available at http://www.disco.com.hk/