

電腦圍棋中盤著手策略

Move Strategies in Middle Game of Computer Go

顏士淨 嚴初麒 許舜欽
國立東華大學資訊工程研究所
國立台灣大學資訊工程研究所

sjyen@mail.ndhu.edu.tw jcchen0213@yahoo.com.tw schsu@mail.cju.edu.tw

摘要

圍棋是目前唯一在世界各地有廣大愛好者及職業制度，但電腦程式只停留在業餘中等水準的對局遊戲，所以電腦圍棋可說是人工智慧的一項新的重大挑戰。本文中將提出電腦圍棋中盤著手策略，包括棋步產生及評估的各種方法。這些方法已成功的運用在電腦圍棋程式吉米上，並且有很好的表現。

1. 引言

電腦對局是人工智慧領域之中相當引人注目的一門學問，在電腦西洋棋方面，自從 1950 年 Shannon 提出基本理論之後，歷經五十餘年的研究發展，1997 年電腦西洋棋程式「深藍」擊敗人類世界冠軍，可說是立下了劃時代的里程碑。而在研究過程中，也開發出許多知名的搜尋 (search) 或切捨 (cut-off) 方面的演算法 [5] [6]。

電腦圍棋發展的歷史約有三十年左右，目前世界上最強的電腦圍棋程式棋力約有台灣業餘六級，程式進步的速度有趨緩的現象。在本文中，我們結合高段棋士下棋的觀念以及電腦對局的各種理論，提出在中盤時，電腦圍棋下棋的方法，我們已經把這些方法成功的用在電腦圍棋上，並且有很好的表現。 [4] [15] [9]

在本篇文章中，我們首先闡述電腦圍棋的基本原理以及與本篇文章相關的圍棋知識表示法，之後在第四節說明中盤棋步產生及評估的基本方法，在第五節中，我們將詳細說明中盤時著手策略。

2. 電腦圍棋的基本原理

運用電腦來下圍棋，似乎是一個很直接的想法，因為圍棋的規則很簡單，勝負定義也很明確，這些都和電腦本身的特性相符合。另一方面由於它已被古今中外的許多專家研究了數千年，許許多多的戰術觀念及思考方法已被開發出來，這些都是可以在發展電腦圍棋時去應用或參考的。

但是電腦圍棋的發展過程，卻沒有想像中順利，雖然圍棋規則很簡單，但是由於盤面廣大 (一般的對局棋盤是 19×19)，對局時的變化卻比其它的棋戲複雜得多。例如西洋棋或象棋，已能藉由一些簡單的推理與深度的搜尋思考而達到相當高的棋力，但這種方法卻不太適合應用在圍棋這種高複雜度的棋戲中。A. Samuel 估計 checker 的複雜度大約是 10 的 40 次方，而 A. Newll 估計西洋棋的複雜度大約是 10 的 120 次方 [16]。這兩種棋戲的複雜度雖然已是天文數字，但比起圍棋的複雜度則要小得多了，Brown 及 Dowsey 估計圍棋所有可能的變化大約是 10 的 700 次方 [7]。

目前電腦圍棋程式的棋力依舊不甚理想，截至目前 2002 年為止最強的圍棋程式棋力約為台灣業餘六級，僅相當於業餘棋手的中等水準，更不是職業棋手的對手。比起西洋棋或象棋，圍棋程式棋力進步緩慢的原因有以下三點：

- a. 圍棋的搜尋分支較大、深度較深。圍棋的盤面為 19×19 比起西洋棋的 8×8 或象棋 9×10 都要大很多，而且完成一場對局所需的手數通常也是圍棋較多，所以圍棋需要更多的搜尋資源，諸如運算速度與記憶體空間等。目前的電腦硬體尚不足以滿足這些需求，有待電腦硬體的發展來配合。
- b. 圍棋的勝負判斷較為複雜，西洋棋與象棋只要吃掉對方的「王」就可以獲勝。反觀圍棋，需要在最後階段計算雙方所圍得的領土大小來決定勝負，而何謂領土、以及如何有效地圍取領土都是抽象複雜的概念，這些概念都需要發展新的資料結構及技術來表示。
- c. 圍棋只有一種棋子。西洋棋與象棋都有多種棋子，而其中有幾種棋子的功能特別強，使得盤面情勢能夠由雙方所擁有的強力棋子的數量來簡單判斷。然而圍棋中單一棋子難以判斷價值，通常必須將數個至數十個相接近的棋子組成一個價值單位 (棋塊) 來判斷，而這種階層結構的規則卻又是抽象而複雜的概念。

由於圍棋的複雜度太高，如果僅用窮舉搜尋的技巧，並不能得到我們要的結果，因此必需發展其它策略來製作電腦圍棋程式。直觀上來說，最直接的方式，就是讓電腦去模擬人類下棋的思考方式。而就人們下棋思考方向而言，選擇著點時大都根據該點是否利於佔地、是否利於攻防、是否有關死活等，這也是現今的電腦圍棋程式最常用的方法 [13][12][18][19][9][6]。

3. 圍棋知識的表示法

圍棋是一個複雜而抽象的問題。為了解決這個問題，我們除了使用各種搜尋技巧之外，更希望能運用到人類長久以來下圍棋所累積的龐大知識。為了將這些抽象知識對映到電腦中，讓數位化的程式能夠利用，我們需要一套清楚而明確的定義來描述圍棋的各種知識。首先我們介紹幾個描述棋局進行現況的用詞。

- a. **盤面**: 棋盤上的棋子分布狀況。有時也明確稱呼其為「原始盤面」。傳統上盤面是以一個二維陣列來表示，陣列中的元素只能是三種值的其中一個：「黑棋」、「白棋」或「空點」。在對局的過程之中，盤面是最原始的資料型態，其他一切資料與物件都必須透過解析盤面而得。
- b. **盤面資訊**: 針對某一特定盤面的所有圍棋知識的總合。它是一組龐大的資料結構，由各式各樣的物件所組成，這些物件彼此間有複雜的階層關係與交互作用。當然，盤面也是盤面資訊中的一筆資料，並且可以作為存取盤面資訊時的唯一鍵值。盤面資訊中的所有物件與資料，理論上都可以直接由盤面推算而得。
- c. **著方**: 圍棋的對局過程是由兩人交互放置棋子（又稱著子）。著方是指在當對局進行到某個特定盤面時具有著子權力的一方，其值只可能是黑棋或是白棋。
- d. **棋步**: 下棋的動作，通常是指黑棋或白棋於棋盤上的某一空點放入棋子，經過氣盡提取等處理過程之後產生出新的盤面。不過圍棋規定對局雙方在必要時也可以「虛手」，而一個虛手棋步並不會改變盤面。

接下來是現今各種圍棋程式普遍使用的階層式物件，包括了棋子、棋串、棋塊等。[18][9]

- a. **棋子或空點**: 這是棋盤上所放置的最基本物件。如果是棋子的話，除了記錄棋色還必須記錄所屬的棋串編號。如果是空點的話，則會記錄用於棋形比對的棋形碼。
- b. **棋串**: 棋串由同色且互相連接的棋子所構成。圖 1 右下角標示為 B 而互相連接的黑子就是一個棋串，其特質包含氣數及氣點等資訊。當兩個以上的棋串互相連接時，即合併為一個棋串。每個棋串可用一個結構變數來表示。內含此棋串的顏色、氣數、第一個棋子的位置，以及棋塊狀態等資料。較複雜的結構還包含氣點及相鄰接的棋串等資訊。
- c. **棋塊**: 同顏色的棋串以某種關係位置互相接近但未連接，而且不易被對方切斷時，這些棋串便構成棋塊(Group)。圖 1 中標示為 A、B、C 的棋子分別形成三個棋塊，關於棋塊詳細的定義可參考[6]，棋塊是圍棋戰術運用上最基本的結構。大部分的圍棋戰術就建立在棋塊的攻防上。在佈局時，尤其是拆邊要點的選擇，必須考慮如何拆邊圍地、防守自己的棋塊，或是考慮如何打入對方的地，予以破壞，並攻擊對方的棋塊。因此我們的佈局系統必須有正確的棋塊資訊，方可正確運作。每個棋塊可用一個結構變數來表示。因為棋塊的資訊直接被戰術系統所使用，所以結構較為複雜且不易計算。其基本的結構包括此棋塊的顏色，內含棋串數，棋塊圍取的地域點數及位置、附近的其它棋塊、勢力的影響、棋塊安危程度、棋塊本身的價值以及棋塊其它的狀態等資訊。

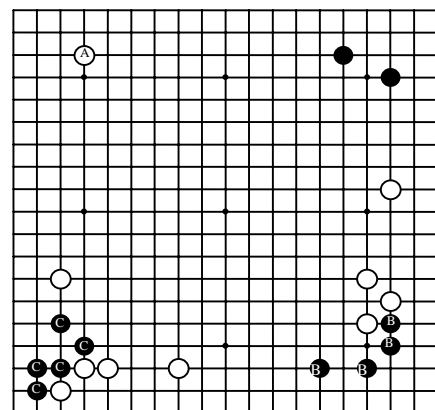


圖 1 棋子、棋串與棋塊

4. 中盤產生及評估棋步的基本方法

我們的電腦圍棋程式是採 Type-B Search，所以如何產生及評估所要的棋步，是很重要的。我

們的目標是產生的棋步越少越好，但是當然一定要包含最佳棋步。而產生棋步之後，必須能夠用靜態方式大略評估每個棋步，如此最後以全局搜尋來評估棋步時，才能參考此大略評估值，達到最好的效果。

為了能夠正確而快速的評估每個棋步，我們首先仔細分析人類高段棋士評估棋步的方式，發現高段棋士通常是先將棋步分類，根據各部分的特有的手法以及棋型、戰略等等細節，辨識出此棋步的功能，再分析此棋步對大局的影響(若是官子，則通常可省略此步驟)，來分析此步棋的好壞。然後一一評估(量化)這些候選棋步的著手價值，最後再根據量化所得的值來挑選所要的棋步。

系統產生及評估棋步的流程大致如下所述：

1. 根據棋形、棋理、棋串攻殺結果產生棋步。
2. 計算出大略量化值。大略量化值的結構可以用 (max, min) 表示，其中包括兩個數值，“ max ”是此候選棋步量化值的可能最大值，“ min ”就是此候選棋步量化值的可能最小值，此 min 值可以是負數，代表下此著手有可能使對方的地域增加。
3. 根據此大略量化值，由全局搜尋系統來找出下一步著手。

量化棋步價值系統的主要構想是將以往處理收官的觀念應用在中盤方面。在棋局進入收官時，由於合法步變得很少，形勢判斷也比較容易，故電腦圍棋程式可以運用組合對局理論 (combinatorial game theory) 來判斷最佳的著手，而且可以達到相當好的效果[15]。但事實上，我們發現，數學家們所開發出來的組合對局理論思考方式和人類棋手某些處理終盤(官子)的思考方式竟然非常類似。近年來許多圍棋專家棋士如大竹英雄及依田記基等[1]，也都嘗試將這些思考方式(官子理論)更進一步運用在處理佈局及中盤方面，而且有一些不錯的成果。而由於我們在電腦圍棋的棋塊分析、形勢判斷和棋形辨認方面的技術已有所進步，因此我們可以參考這些思考方式，將量化棋步的觀念應用在中盤部分。

上述的方法好處是可以很容易的將專家知識分析出來，應用在系統中。我們估計此方法約可到達棋力初段。其所不及人類高段棋士的是，人類通常具有全盤戰略的考量，例如人類可能會在某一部份只求定型，以求得能搶占別處大場。但對程式來說，由於只要求在局部下出最好的結

果，所以無法產生此構想。解決之道是去做深達十多步，甚至三四十步的全局搜尋。但這對目前的電腦圍棋程式來說，是一件不可能的事。

根據電腦圍棋的需求，我們將圍棋的著手依其目的分為佈局，中盤，收官的棋步。佈局部分我們已發表[19]，在本文中，我們將繼續探討中盤部分。以下我們將分析中盤部分的特性，並敘述如何處理這些狀況。

5. 中盤分析及處理

中盤泛指雙方棋塊短兵相接，攻擊一方藉由攻擊對方不安定棋塊取利，或殺死對方棋塊而圍成大空。而防守一方則要處理自己不安定棋塊，將損失減少到最小。中盤的攻防大致發生在佈局邊上開拆結束，到雙方所有棋塊皆進入安定狀態，開始收官為止。

中盤為棋塊的攻防，但由於棋串的攻防通常都是非常緊急的棋步，所以我們特別將棋串的處理獨立出來，而將中盤系統區分為棋串攻防與棋塊攻防。

5.1 棋串攻防

救出自己危險的棋串或吃掉對方危險的棋串，通常是最緊急的棋步。較早的圍棋程式，一發現有危險的棋串，通常都是馬上就把它吃掉或救出它。但若要提昇圍棋程式的棋力，正確的量化棋串攻殺的棋步是很重要的一件事。

在我們的系統之中，棋串攻殺的棋步是由棋串攻殺系統產生的，此子系統會輸出有哪些棋串是危險的，並且如何去逃及如何去吃它。詳細的棋串攻殺系統可參考[18]。

在量化一個棋串攻殺的棋步時，我們必須考慮兩件事情，首先是計算出此棋串的價值，再來則是必須考慮此棋步本身除了吃掉獲救出此棋串之外，其餘的價值(如圍空或影響勢力等)。以下我們將討論如何計算之：

(a) 棋串價值的計算方法

棋串的價值包括棋串本身被提取時，對雙方可得到多少目數以及對雙方周圍棋塊的影響。前者比較簡單，除了打劫或是比較複雜的情形(如打二還一)之外，只要把棋串中棋子的數目乘以二，就是雙方可得到的目數。

後者就比較複雜了，由於棋串的被提取，不但可讓周圍對方棋塊的眼位增加，也可使棋塊連接，斷點消失。所以對於危險的棋塊有很大的幫助，這些都是必須計算在棋串價值之中的。

(b) 其餘價值的計算方法

此棋步的其餘價值包括此棋步本身可形成的好形、可圍到或可破到的實空和對勢力的影響等等。由於有時候要吃一個棋串的方法不只一種，此時，就要考慮那一種吃法可以得到最大的利益。

5.2 棋塊攻防

中盤時對於對局雙方不安定棋塊的處理，是電腦圍棋棋力強弱的關鍵點，也可說是目前電腦圍棋的最大難處之一。我們系統的處理流程大致如下：

1. 根據資料結構中的資訊，找出所有不安定的棋塊。
2. 找出這些棋塊附近所有可影響死活的著手。
3. 量化這些著手。

由於我們資料結構的完備，步驟一很容易達成。而步驟二及步驟三的詳細做法如下兩小節所述。

5.2.1 找出這些棋塊附近所有可影響死活的著手

為了解決這個難題，我們將中盤系統細分為三個子系統：眼位判斷系統，封鎖與逃脫系統與好點判斷系統等。在此我們可以看出中盤系統主要產生候選棋步的目的如下所列：

要防守一個不安定棋塊，共有下列方法：

- a. 攻擊附近比此棋塊弱的對方棋塊。
- b. 攻擊對方棋串或防守己方棋串。
- c. 擴大此棋塊做活空間。
- d. 下在此棋塊的眼位急所。
- e. 逃脫此棋塊。

- f. 將此棋塊和其他己方棋塊連接在一起。
- g. 下在重要的棋形。

要攻擊一個不安定棋塊，共有下列方法：

- a. 防守附近比此棋塊弱的己方棋塊。
- b. 攻擊對方棋串或防守己方棋串。
- c. 縮小此棋塊做活空間。
- d. 下在此棋塊的眼位急所。
- e. 封鎖此棋塊。
- f. 將此棋塊和其他對方棋塊切斷。
- g. 下在重要的棋形。

上述的防守及攻擊的 a 方法，為參照另一方的攻擊及防守方法。而 b 方式，在上一節棋串攻防中處理。所以我們在此只討論如何進行上述之 c、d、e、f、g。這五個方法中，除了 c 已有拆邊系統及官子系統作為輔助外，其餘都需要獨特的子系統協助計算。我們將 c 的處理方式統稱為地域處理系統，其餘子系統分別為：眼位判斷系統、封鎖與逃脫系統、切斷與連接系統、好點判斷系統。這些系統的運作方式如下：

(1) 地域處理系統

此方面的著手為棋塊攻防時，最常用到的著手。在攻擊對方時，藉由攻擊而圍空，相反的，防守時也要儘量減少損失，防止對方藉由攻擊而獲得利益。由於拆邊圍空與官子圍空一樣，都是計算地域的出入，因此，在此僅以討論拆邊棋步作為說明。

在拆邊棋步的價值量化方面，一般來說，棋局到了要考慮拆邊時，通常棋盤上已有相當多的棋子，各棋塊亦已有了強弱的分別。所以我們在考慮每一個拆邊的棋步的大小時，除了要考慮雙方佔地目數的出入之外，也要考慮此一棋步對雙方棋塊的影響[1]。實際量化的值就是考慮上述兩項所得之利益的值的總和。以下我們就分別敘述如何計算這兩個值。

在計算佔地目數的出入方面，首先是找出對方在此候選棋步周圍的對方最佳著手。我們的做法是，將這個資訊預先儲存在棋形資料庫中，當系統以棋形比對的方式找出某一候選棋步時，同時也可從這個棋形資料中知道對方在此候選棋步周圍的對方最佳著手。第二點則是所謂佔地目數，其實也必須包括對周圍空點的影響(圍棋術語稱為厚勢或薄味)，在我們的系統中，可以根據影響力評估值的增減，來計算此部份的值。圖六是一個

量化拆邊棋步的例子：當黑棋下在 A 點時，黑棋可圍到 4 目的空並增加黑棋約 2 目的影響力。而白棋下在 B 點時，白棋也可圍到 4 目的空並增加白棋約 2 目的影響力。所以在這裡計算佔地目數的出入方面，黑棋下在 A 點的價值為 12 目(4+2+4+2)。

在對雙方棋塊的影響方面，這也是中盤系統所需特別注意的，我們首先根據形勢判斷系統找出雙方的不安全的棋塊，因為拆邊時，所佔的目數一定會增加，所以拆邊棋步對於支援或攻擊雙方的不安全的棋塊必定會有一定的效果。所以當某一候選棋步落在雙方的不安全的棋塊附近時，我們就再將其對周圍的不安全的棋塊的影響程度加入此候選棋步的量化值。例如在圖 2 中，棋塊 C 的價值為 10 目左右，棋塊 D 的價值也在 10 目左右。在此例子中，由於棋塊 C 或棋塊 D 在被對方攻擊時，對方都必須在其周圍連下兩手，才可吃掉這棋塊。所以在考慮對雙方棋塊的影響方面，黑棋下在 A 點的價值大約是 10 目 $((10+10)/2)$ 的價值，加上先前算出的佔地目數的出入方面的 12 目，此拆邊著手的量化值為 22 目。

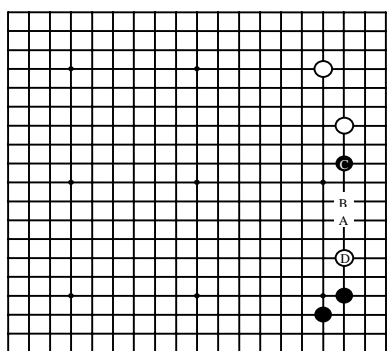


圖 2 量化拆邊棋步的例子

(2)眼位判斷系統：

關於眼位判斷的研究，相關的文獻很多 [21][22]。目前為止，最為詳盡的研究應該是陳志行教授與陳克訓教授所作的研究 [21]。這些文獻的特點為所分析的棋塊，大都已經做過了一些前置處理，例如必須先確立那些點是棋塊的地域，那些棋子是死子等等。而這些資訊，在我們的資料結構中都有，所以我們可以很容易的應用這些相關文獻的結果。但是在這些文獻中，對於在棋塊的地域之中有死子或是較複雜的情況的情況，還是會有錯誤。而為了彌補這些錯誤，我們是以製作眼位棋形的方式補救，也就是說，同樣的失誤，我們用人為的方式補救，所以下次就不會再錯了。而對於某些無法確定的情況，我們也標注說此點可能可以做活，而實際上可不可以真的防守棋塊，可以再交由搜尋系統來判斷。例如圖 3 為吳清源詰棋集中作品 86 [2]，圖中系統判斷 A 點可能可以殺死白棋，故其大略評估值為(-2,30)，也就

是說，當可以殺死白棋時，可以獲得 30 目的地，但是殺不死時，因為此著手被吃掉，所以反而使對方增加兩目的空。

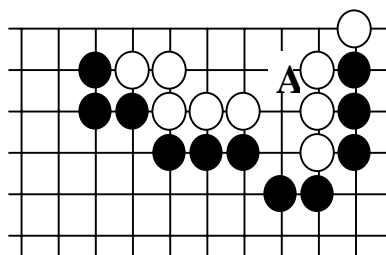


圖 3 吳清源詰棋集中作品 86

(3)封鎖與逃脫系統：

我們整理出封鎖與逃脫的棋形，這些棋形包括一間跳、二間跳、飛封、鎮等等。此系統的著手幾乎都是由棋形產生。由於棋塊在封鎖或逃脫時，通常還未確定是否活定或死定，所以也必須藉由全局搜尋系統的再確認。

(4)切斷與連接系統：

切斷與連接的著手除了由棋形產生之外，也要考慮棋串攻殺所提的資訊。例如吃掉某些棋子而連接，或是救回某些棋子而切斷等。與封鎖與逃脫較不相同的是，通常切斷與連接某棋塊後，就可算出確實的量化值。

(5)好點判斷系統：

除了以上所述的著手之外，還有一些著手，對棋塊的狀態可能有影響。我們將這些著手歸納為好點的著手。

5.2.2 量化這些著手

產生候選棋步之後，接下來最重要的工作就是量化棋步。除了一些很簡單而可以馬上算出的量化值之外，必須用搜尋的方式來計算量化值。這個量化棋步的基本原理也是根據官子理論來運作，其主要原則有下列幾項：

- a. 若下此候選棋步後，可殺死或救活某不安定棋塊，則此候選棋步的價值為此不安定棋塊的價值加上周圍對方已死棋塊的價值，以及此棋步所造成的棋形的價值。

- b. 若下此候選棋步後，未能確實殺死或救活某不安定棋塊，需要在對方手拔的情形下，再下一手才能確實殺死或救活某不安定棋塊，此時計算方式應為此不安定棋塊的價值加上周圍對方已死棋塊的價值再除以 2 (因為需要下兩手才可得到此價值)，以及此棋步所造成的棋形的價值。

根據以上的原則，我們只要用局部搜尋去判斷要下幾手才能確實殺死或救活某不安定棋塊，就可正確的估計出某候選棋步的量化值。

6. 結論

在本篇文章中，我們針對中盤階段，提出了一個有效解決這個問題的方法。我們首先分析圍棋高段棋士對於中盤的各種策略及分析方式，然後由電腦計算的角度，一一分析並實作出這些概念。在每一部份中我們除了研究如何發現好的棋步之外，並根據專家棋士的理論，設法量化每一個棋步。從每一個棋步的量化值加以整合，使得我們的電腦圍棋程式可以正確地整合及運用此系統。

目前此中盤系統還需增強的部分為對於棋塊死活的判斷，一個棋塊死活的資訊將大大影響中盤的策略。這部分的問題應該歸納於電腦圍棋形勢判斷的問題上，也就是說，我們必須再加強形勢判斷系統的準確度，才能使得中盤處理的能力更為加強。

參考文獻

1. 大竹英雄九段，大竹英雄圍棋系列 4 – 圍棋大局觀，6至17頁，理藝出版社，1992。
2. 吳清源，吳清源詰棋集、世界文物出版社，台灣台北，1986。
3. 許舜欽，電腦圍棋在台灣的回顧與前瞻，中國工程師學會，日本分會，1989年學術研討會論文集，1989。
4. 顏士淨、許舜欽，電腦圍棋的發展概況，Communications of IICM，Vol. 1, NO. 2, April, 1997, pages 23 -- 30。
5. L.V. Allis, Van Den Herik, and H.J. Herschberg. Heuristic Programming in Artificial Intelligence 2, Ellis Horwood 1991.
6. Bouzy, B, T. Cazenave, Computer go: An ai-oriented survey, Artificial Intelligence 132(1),2001, pages 39-103.
7. D.J. H. Brown and Dowsey, S. The challenge of Go. New Scientist, 1981, pages 303--305.
8. Ken Chen. Group identification in Computer Go. Heuristic Programming in Artificial Intelligence, Levy & Beal (Eds.), Ellis Horwood 1989, pages 195--210.
9. Keh-Hsun Chen, Computer Go: Knowledge, Search, and Move Decision, ICCA, Vol. 24, No. 4, 2001, pages 203 – 216.
10. Keh-Hsun Chen, Some Practical Techniques for Global search in Go, ICCA, Vol. 23, No. 2, 2000, pages 67 – 74.
11. K. Chen, “Soft Decomposition Search in the Game of Go”, Proceedings of the 6th International Conference on Computer Science and Informatics, 461-464, 2002.
12. S.C. Hsu, J.C. Yan, and H. Chang. Design and implementation of a computer Go program Archimage 1.1. Journal of Information Science and Engineering 10, pages 239--258, 1994.
13. S.C. Hsu and D.Y. Liu. The design and construction of the computer Go program Dragon 2. Computer Go, No. 16, pages 3--14, 1991.
14. Richard J. Lorentz. Pattern matching in a Go Playing Program. Game programming workshop in Japan, pages 167--174, 1995.
15. M. Muller, News, Information, Tournaments, and Reports, ICCA, Vol. 22, No. 2, 1999, pages 104-112.
16. A. Newell A., J.C. Shaw, and H.A. Simon. Chess playing programs and the problem of complexity. IBM Journal of Research and Development, Vol. 4, No. 2. Pages 320--335, 1958.
17. Ryder, J. L.,”Heuristic Analysis of large Tree as Generated in the Game of Go”, AIM-271, Stanford Univ., 1972.
18. Shi-Jim Yen, Shun-Chin Hsu, “A Positional Judgment System for Computer Go,” Advances in Computer Chess 9 Conference, June 16 to June 18, 1999 in Paderborn Germany.
19. Shi-Jim Yen, Wen-Jyh Chen, Shun-Chin Hsu, “Design and Implementation of a Heuristic beginning system for computer Go,” JCIS '98 Proceedings VolII, Association for Intelligent Machinery, pp 381-384, Oct. 23-28, 1998 in Sheraton Imperial Hotel & Convention Center, RTP, N.C., USA.
20. Zobrist, A. L. Feature Extraction and Representation for Pattern Recognition and the Game of Go, Ph.D. Dissertation, University of Wisconsin, 1970.
21. Ken Chen and Zixing Chen, “Static Analysis of Life and Death in the Game of Go,” Personal Comm., 1997.
22. Popma, R. and Allis, L., Life and Death Refined, Heuristic Programming in Artificial Intelligence 3, edited by J. Herik & V. Allis, pp. 157-164, Ellis Horwood, 1992.