益智遊戲「L 棋(L-Game)」之電腦解法及分析

蔡明原　林順喜
國立臺灣師範大學　資訊教育系

摘　要

由英國劍橋大學 Edward De Bono 教授所發明的 L-Game，是一個兩人對奕的有趣遊戲，本文嘗試利用電腦來分析這個遊戲是否存在必贏的策略？先下手的一方是否有占便宜？是否存在纏鬥最久的盤面？經過我們的分析，發現此遊戲雙方總共有 36736 種盤面(每一方 18368 種)，其中有 240 種輸的盤面，而且先後下棋的雙方擁有一樣多對應盤面，沒有任何一方占便宜，而且並不存在必贏的方法，下棋的雙方如果都沒有下錯，依一定的下棋策略，這個遊戲就不斷進行下去，也就是不斷地纏鬥下去。最後我們將研究的成果設計成 CGI 程式，放在網頁上供大家與我們的程式下棋。

一、 緒論

1. L 棋(L-Game)介紹：

我們偶然在網際網路上尋找一些有趣的的遊戲網站時，發現了一個 GamePuzzle 的網站，http://www.gamepuzzle.com/abstrct3.htm#L3，在其中我們發現了 L 棋這個遊戲，這個遊戲是由英國劍橋大學 Edward De Bono (黎波諾) 教授所設計的遊戲。

這個遊戲的玩法如下：

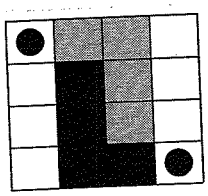


圖 1.

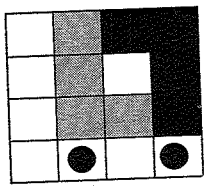


圖 2.

說明：(圖 1)爲 4× 4 棋盤，上置兩個 L 形棋子及兩顆圓形棋子，其玩法爲一人一個 L 形棋子，決定先後手後，一方拿起他自己的 L 形棋子，可任意翻轉後放

回棋盤，但不可放回原位，然後可以移動一顆圓形棋子或者不移，如此輪流移動，直到有一方拿起他自己的L形棋子後無處可放則爲負方。舉例來說，(圖2)持黑色的一方沒法再動了，就輸了這場棋賽。

查詢相關資料，發現民國74年11月27日的報紙上有一則有關L棋的報導：三個師大附中的學生寫了一個以人工智慧方法設計的程式來玩L棋，程式從與人的對戰中學習，不重複錯誤的下棋盤面，並建立一個資料庫儲存盤面，而在下棋時，選擇過去下棋經驗中離失敗最遠的盤面來下，這樣訓練的結果，他們的程式越來越會下L棋，而且總有一天這個有關L棋的資料庫會學習到所有的盤面與相對應的下棋方法，在當時他們能以高中生的身份踏入人工智慧的研究，實在不容易，可惜他們並未證明 L 棋是否存在必贏的下棋方法？先下L棋的一方是否擁有較多的勝算？所以我們希望能以不同的方式來研究L棋。

後來我們在圖書館找到了水牛出版社於民國七十七年所出版的「水平思考五日訓練法」一書，這是黎波諾教授出版的書，敘述面對一些問題時，我們應該如何以水平思考的方式去分析問題，其中第三個單元就是以L棋爲例來分析互相競爭下最容易發生的問題及如何思考下棋的策略。黎波諾教授指出，與其他需要集中精神去移動許多棋子的遊戲比起來，L棋是一個構造簡單卻需要仔細思考的遊戲，L棋有五個特點：

(1) 每個人只有一顆子(L形棋子)

(2) 棋盤不大(16格)

(3) 規則簡單容易記

(4) 下棋需要仔細的思考(因爲棋子的移動方法很多)

(5) L-Game不像井字遊戲，受到盤面的限制，當雙方把所有的格子填滿，遊戲就結束，而L-Game的棋子不會增減，必定要有一方的L形棋子不能移動，才會結束。

(6) 勝負無限制，先下手或後下手無關勝負。

雖然黎波諾教授對於他所設計的遊戲給予很高的評價，但是從文獻中我們並沒有發現黎波諾教授如何證明這個遊戲真的就如他的分析，沒有好的方法致勝？也沒有說明爲何當初要要設計那樣的啓始盤面，而且文中有一段黎波諾教授指出：「下棋的一方有50種以上下棋的方法，對方也有 50 種以上對應的方法，要一一檢討優劣是困難的，如果要預先考慮個兩三手，既使是巨大的電子計算機也不可能做到。」真的是如此嗎？所以我們希望以程式來證明我們能以電腦來分析此棋戲，而且能找出最佳的下棋方法。

2. 人腦玩此遊戲和電腦玩有何不同之處？

以人腦來玩這樣的對奕遊戲，除了要考慮自己的攻守策略外，還要考慮對方的可能採

取的攻守策略，但是人常常會考慮得不夠周延，而且對於重複的盤面不能有效的記錄，所以不能歸納出有效的下棋策略，常常都是從不斷的練習中減少自己犯錯的機會，但是也不能保證自己一定不會再出錯。而這個遊戲中，由於盤面是正方形的，雙方的棋子數目及形狀也具有對稱性，所以一個盤面有八種對稱的相同盤面，對於人腦的記憶是蠻大的負擔。此外，人類在下棋時，常常採取所謂的「經驗法則」，就像之前師大附中的學生所設計的人工智慧程式一樣，從不斷的下棋經驗中去找自己最佳的下棋策略，但是經驗未必代表就是最佳解，可能只是還沒失敗過而已。

如果我們要讓電腦來解這樣的對奕遊戲，和人腦最大的不同，就是電腦能有效地記錄所走過的盤面，並且能從展開 Game-tree 的過程中尋找最佳的對應盤面，而且能判斷盤面的對稱性，不過這樣的假設前提是要有一個好的方法去記錄盤面及分析盤面，否則只是浪費大量的記憶體及時間。此外，電腦在面對許多可選擇的盤面時，可以用機率的方式計算每一個可以走的盤面有多少勝利的機率，尋找最佳的盤面。另一方面由於電腦可以建構 Game-tree 窮舉所有的盤面，所以我們也能透過分析 Game-tree 的勝負分佈來找出好的下棋策略，幫助往後人們在玩這樣的棋類時，有更明確的判斷依據。而這個研究最後的目的，我們希望能讓我們的程式在與人玩 L-Game 的時候，能立於不敗的情況。

二、 電腦解題之演算法

面對這個遊戲，如果要讓電腦解題，第一步就是要有效的記錄盤面，我們設計以下的方法[(x1,y1,dir1),(x2,y2,dir2),(n1,n2),who]來表示，其中 x1、y1、dir1、x2、y2、dir2 都是 0 到 3 的整數，n1、n2 是 0 到 6 的整數，who 是 0 或 1，所以一個盤面占用 19bits 來記錄。雖然盤面只有 4*4，但是每一個 L 形的棋子有八個可能的形狀(圖 3)，但是在盤面的一個位置上不可能八種形狀都能放入，分析的結果發現如果以 L 形突出的那一點(x,y)爲記錄點，可能的擺放方式剛好是上(dir=0)、下(dir=1)、左(dir=2)、右(dir=3)四種可能性。而兩個中立的圓形棋子的記錄方法如果以一個二維的座標(16 個可能位置)來記錄，兩顆棋子就可能有 8*15 種組合，其實當兩個 L 形的棋子放好時，盤面上只剩 8 個空位，兩個圓形棋子只可能有 4*7 種排列方式，所以我們只要記錄這兩個圓形棋子之前各別的空白格子數 n1 及 n2，便可以推算它在盤面中的位置，而且可以節省記錄空間。最後加上一個 bit(who)記錄下一步該那一方下手，因此記錄一個盤面只需要 19 個 bit 的空間。

以下面(圖 4)的盤面爲例，這樣的盤面記錄爲[(x1,y1,dir1),(x2,y2,dir2),(n1,n2),who] =[(2,1,0),(0,1,3),(1,5),0]，(2,1,0)是指灰色的 L 形突出來的那一點在座標(x1,y1) =(2,1)，其它三格在它上方，所以記錄爲 dir1=0，同樣的，黑色的 L 形突出的那一點在(x2,y2) =(0,1)，

其它三格在它的右方，記錄爲 dir2=3，而兩顆圓形的棋子，(3,1)的那一顆，之前只有(3,0)這一個空白，所以記錄爲 1，(2,3)的那一顆之前有五個空白，所以記錄爲 5，最後的 who=0 表示下一步該持灰色 L 棋子的一方下手。

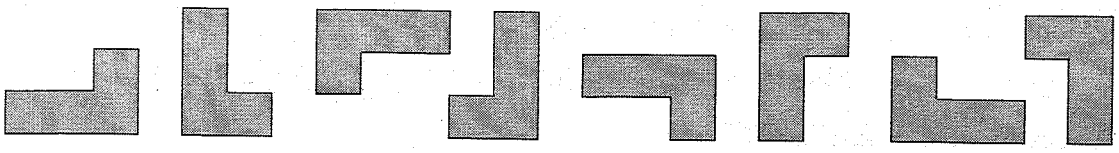


圖 3.　L 形棋子 8 種可能形狀

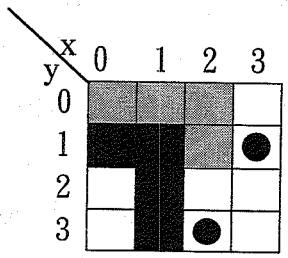


圖 4.

雖然 19 個 bit 的記錄空間已經算滿小的，但是還是需要 $3*2^{19}$ bytes(1536K bytes)的記憶體空間(因爲每一個盤面要 3 個 bytes 來做 pointer)，以前 16 位元的 DOS 作業系統，在 64K 的限制之下，記憶體配置不可能達到 1536K，所以我們整個程式是在 32 位元的作業系統下執行 Borland C++ Builder 3.0。完成了盤面記錄的設計後，便讓程式去產生 Game-tree，並把雙方的所有可能盤面及輸贏盤面記錄下來，並從 Game-tree 的追蹤將每一個盤面的最佳對應盤面找出，並尋找此遊戲是否存在最佳的下棋策略及防守策略。

第二個需要克服的問題，就是 Game-Tree 的記錄，因爲一個盤面平均有 50 個以上的對應盤面(最少 13 個，最多 221 個)，所以我們在設計 Game-Tree 時應該把 Child node 指向 Parent node。以 L-Game 的啓始盤面爲例，共有 65 個對應盤面，所以這 65 個盤面都指向啓始盤面。我們把啓始盤面的編號設成(Node 0)其盤面記錄爲[(1,0,3),(2,3,2),(0,6),0]，它沒有 Parent，所以指向-1，而它之下的 65 個 Node，分別編號爲(Node 1)～(Node 65)，而他們的指標就指向(Node 0)。當展開的過程中發現該 Node(例如圖 5 的 Node 210)經程式判斷爲不能再動的盤面，就記錄不能動的一方爲輸。

在 Game-Tree 展開的過程中，會有重覆的盤面出現，對於這樣的盤面，我們就不再展

開它。除此之外，我們還需要一個陣列把所有輸的盤面(就是找不到下一步的盤面，以下叫這些輸的 Node 是 LoseNode)記下來。

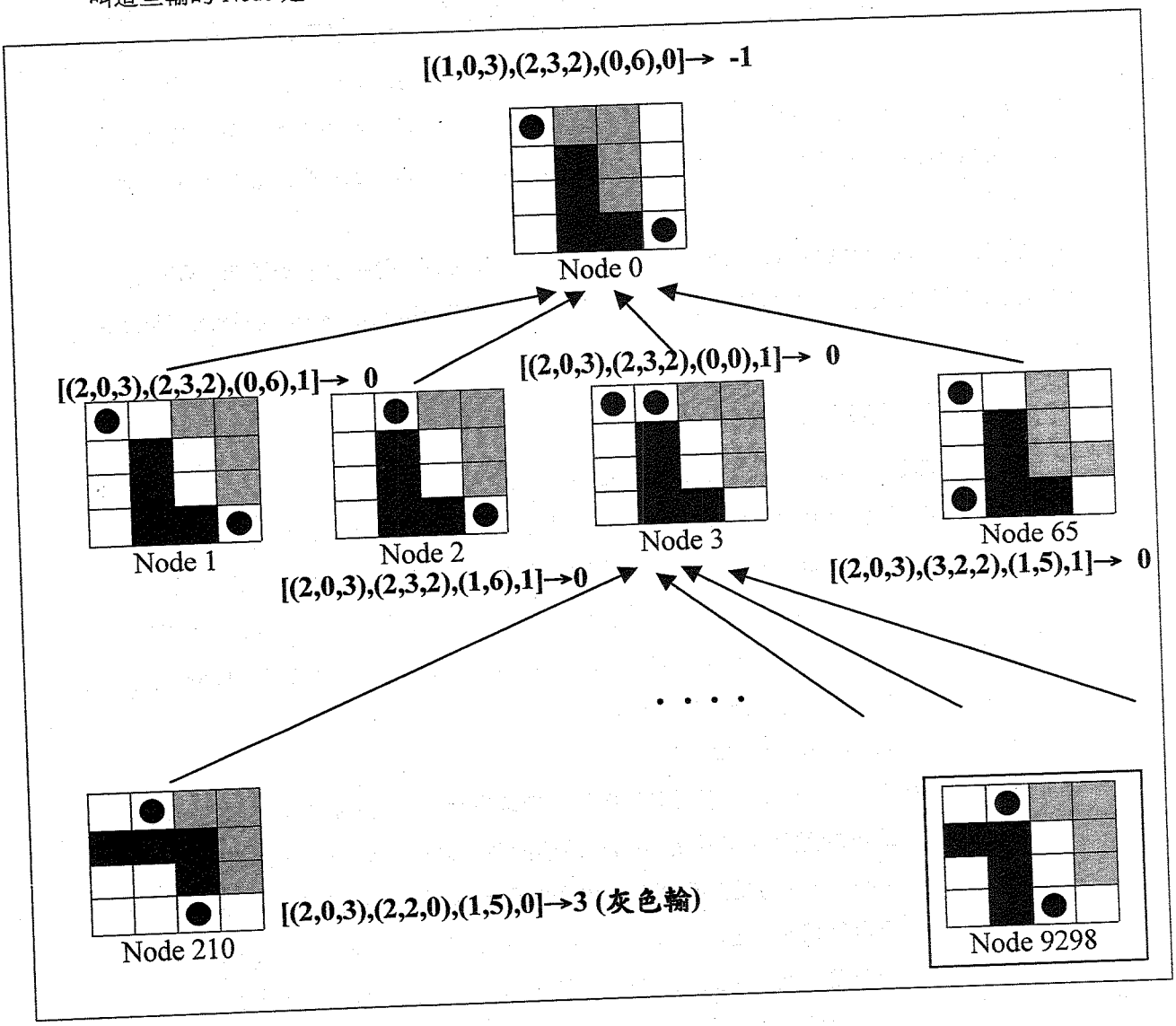


圖 5. Game-Tree 的架構圖

接下來，我們要用倒推方式找出 LoseNode 所有可能的前一步盤面，前一步的盤面不一定是上圖中 LoseNode 的 Parent node，舉例而言，上圖中 Node 210 是灰色方的一個 LoseNode，而 Node 3 是它的 Parent Node，但 Node 210 倒推回去的前一步有 24 個之多，上圖 Node 9298 就是一個可能是 Node 210 的上一步。這 24 個 Node(以下我們叫這樣的 Node 是 WinNode)，表示在下棋的過程中如果有機會到達這一盤面，就應該下讓對方輸的盤面來

讓自己贏。而這樣的 WinNode 的前一步可能的 Node(以下我們叫這樣的 Node 是 GeneralNode)，GeneralNode 在選擇最佳的的下一步時，一定不能考慮走到這些 WinNode，才不會輸。所有的 Node，除了 LoseNode 及 WinNode 之外，都是屬於 GeneralNode，這些 GeneralNode，可以走的下一盤面可能不只一個，如何從這些可以走的盤面找一個對自己最佳的盤面，我們就必須分析可以走的盤面之致勝機率來下決定，致勝機率的算法待會兒再介紹。依這樣的方法，我們可以把原來 Game-Tree 上的每一個 Node 找到最好的下一步，加以記錄。

請注意，這邊所說的 LoseNode、WinNode、GeneralNode，還必須分辨是屬於哪一方。三個之間的在 Game-Tree 中的關係是 LoseNode(A 方)指向 WinNode(B 方)指向 GeneralNode(A 方)。

三、 程式執行結果與分析

我們程式的執行環境，是在 Pentium MMX-233 的主機，64MB SDRAM，Win98 作業系統下，程式語言使用 Borland C++ Builder 3.0。程式第一部份是架構 Game-Tree，展開所有的盤面並找出所有的 LoseNode。第二部份將 LoseNode 以倒推方式標出所有的 WinNode。第三部份找出特別的 GeneralNode(下面文中會介紹爲何特別)，並將 Game-Tree 重新整理。第四部份將所有 Node 的最佳下一步找出，並輸出至檔案，整個程式執行時間大概兩分鐘。以下用問答的方式說明程式執行結果與分析。

1. L 棋有多少個不重覆不對稱的盤面?

我們以程式窮舉了 Game-Tree 下所有的盤面，發現總共有 36736 個 Node，雙方各擁有 18368 種可能但不重覆的盤面，其中有 6144 種盤面是必贏(WinNode)，120 種盤面必輸(LoseNode)，先下手及後下手的雙方都擁有一樣多的對應盤面，這點證實了黎波諾教授所說的 L 棋的特性：勝負無限制。也就是先下手或後下手並無差別。而雙方加起來所有的盤面只有 36736(接近 215)種可能，遠比 219 還小，這表示之前我們對於盤面的記錄方式仍有改進的空間。而這些盤面具有對稱性，所以如果我們把對稱的情況扣除(每個盤面有 8 個對稱情況)，則 L 棋的雙方有 2296 種不重覆不對稱的盤面，其中有 768 個 WinNode、15 個 LoseNode。

2. Game-Tree 的每一個 Node，擁有多少個 Child Node?

每一個盤面，如果可以找到一個 L 形棋子可以放的位子，其它兩個圓形棋子就有 13 種可能的擺法(兩顆都不動 1 種，動第一顆圓形棋子的有 6 種，動第二顆的有 6 種)，也就是說，每一個盤面之下的對應盤面必定是 13 的倍數。我們統計的結果發現一個 Node 最多

可以擁有 221 個 Child Node。

Child Node 個數統計	0 個	13 個	26 個	39 個	52 個	65 個	78 個	91 個	104 個
	240	1440	2400	2880	4880	3456	3920	3072	2016
	117 個	130 個	143 個	156 個	169 個	182 個	195 個	208 個	221 個
	3200	3696	1536	1248	896	480	512	0	864
	總計 36736 個盤面，平均 1 個 Node 有 88.89 個 Child Node								
Child Node 中 WinNode 個數統計	0 個	13 個	26 個	39 個	52 個	65 個	78 個	91 個	104 個
	0	0	32	96	464	848	800	928	1040
	117 個	130 個	143 個	156 個	169 個	182 個	195 個	208 個	221 個
	1216	2048	1312	784	880	464	512	0	864
	總計 12288 個 (雙方各 6144 個)								

如果我們再加以分析這些 Node，其中擁有 169 個以上 Child Node 的盤面，他們的幾乎都是 WinNode(195 個以上的就必定是)，因爲他們的 Child Node 之中必定擁有讓對方輸的 LoseNode。所以當我們在與對手下 L 棋時，如果發現自己的 L 形棋子有 13 種以上的擺法時(13*13=169)，表示我們要贏了，千萬別放過對方唷。

3. 下 L 棋有沒有必勝的策略？

L-Game 並沒有必勝的策略，但是只要依循下面的幾個規則，便可以與對手繼續纏鬥下去，直到有一方露出破綻，不小心走到對方的 WinNode，而輸了遊戲。我們從 Game-Tree 分析出四個玩 L 棋的策略：

(1) L 形棋子盡量不要放在棋盤的角落

(2) 棋子盡量不要放在對稱的位子

(3) 圓形棋子要和自己的 L 形棋子接觸

(4) 要把自己的 L 形棋子放在對方的直角部份內側

但是這四個下棋策略，還需要彼此配合，才能做好防守的工作，例如：不把自己 L 形棋子放在棋盤的角落之外，還要注意自己 L 形棋子的直角內側不要讓對方有機會卡死自己，才不會反而輸了。換句話說，不要輕視圓形棋子的位子，它很可能是救你一命的大功臣唷，以下我們舉個例子來說明：

（圖 7）是比賽中一個盤面，輪到灰色下手，（圖 7.B）、（圖 7.C）、（圖 7.D）是其中三個對應盤面，其中（圖 7.C）是把灰色 L 形棋子放在角落，而（圖 7.B）、（圖 7.D）是把灰色 L 形棋子遠離角落，但是其中（圖 7.D）是最差的方法，因爲雖然它沒放在角落，但是他讓別人可以放到他的直角內側，造成（圖 7.E），所以輸了，而其他兩種還不會馬上輸。

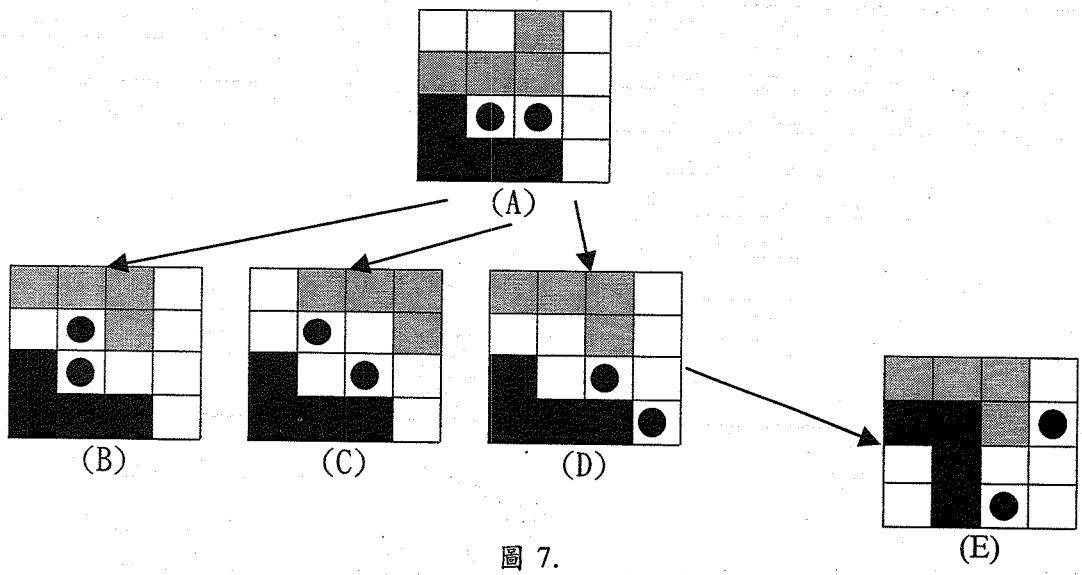
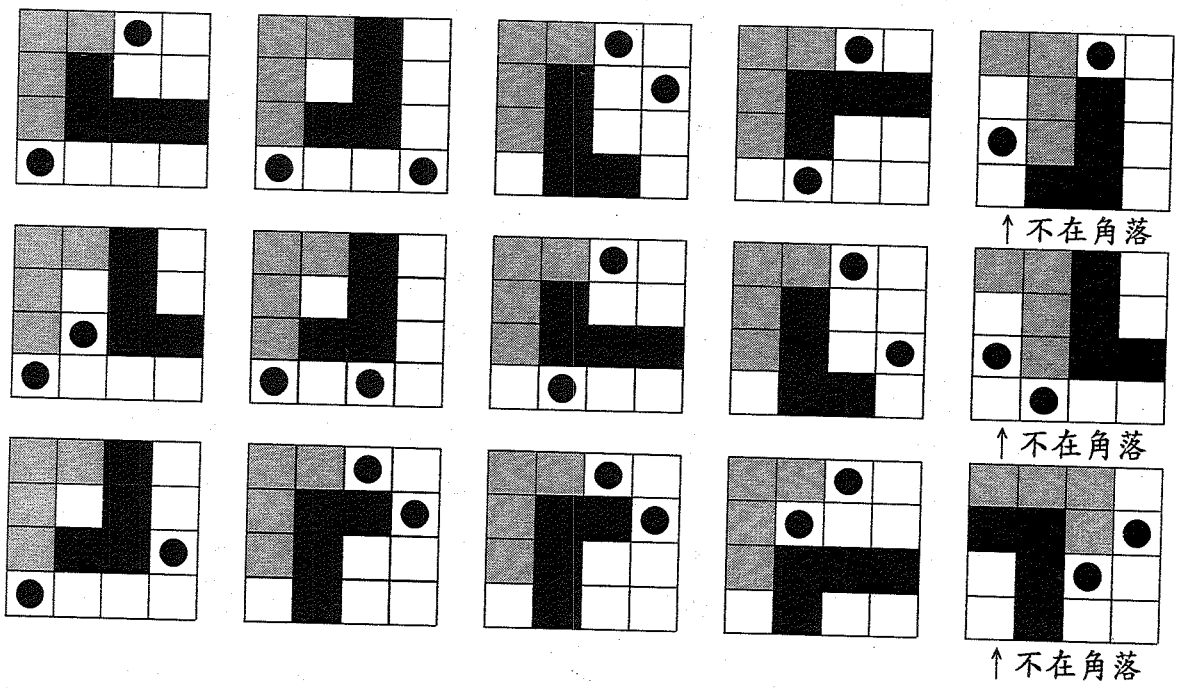


圖 7.

4. LoseNode 擁有什麼樣的特性？有沒有辦法不走到對方的 WinNode？

當我們把盤面的對稱性扣除，發現 15 個(120÷ 8=15)必輸的盤面(LoseNode)之中有 12 個盤面，輸的一方把它的 L 形棋子放在棋盤的角落，但是有三個盤面並非如此，所以「L 形棋子盡量不要放在棋盤的角落」只是勝算較大，仍要移動圓形棋子來配合保護自己。

以下將這 15 種必輸盤面列出(持左上角灰色 L 形棋子的一方輸)



5. GeneralNode 應該怎麼計算「致勝機率」才容易贏？

由上述的 LoseNode(A 方)->WinNode(B 方)->GeneralNode(A 方)的關係中我們知道，在遊戲中任一個 GeneralNode(A 方)的下一步，必定包含一些 WinNode(B 方)和一些 GeneralNode(B 方)，我們當然不能選擇 WinNode(B 方)來當自己(A 方)的下一步，所以我們要分析比較其它剩下的 GeneralNode(B 方)的優缺點來選擇。而這些 GeneralNode(B 方)，他們的下一步的選擇中也必定含有一些 WinNode(A 方)和一些 GeneralNode(A 方)，如果 WinNode(A 方)的個數比 GeneralNode(A 方)的個數多，表示如果剛才我們(A 方)選擇了這個 GeneralNode(B 方)，B 方的下一步如果不小心，就很容易選錯步，選到 WinNode(A 方)，我們(A 方)就贏了

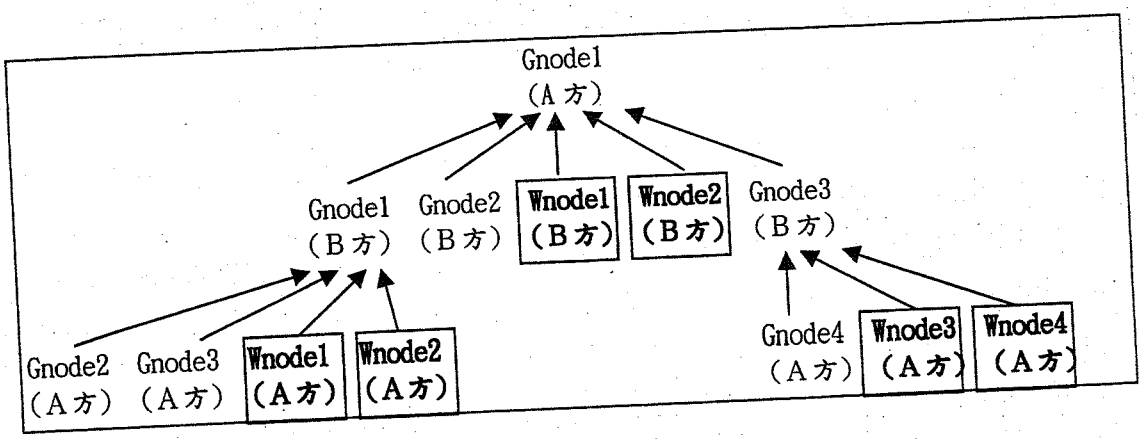


圖 8. 以樹狀圖來表示(Gnode=GeneralNode、Wnode=WinNode)

如果我們是 A 方，處在 Gnode1(A 方)，我們有 Gnode1(B 方)、Gnode2(B 方、Gnode3(B 方) 、Wnode1(B 方)、Wnode2(B 方)五條路，我們一定不會去選 Wnode1(B 方)和 Wnode2(B 方)讓自己輸，所以我們要從其它三條路來比較優劣，其中 Gnode1(B 方)之下有四條路，而 Gnode3(B 方)之下有三條路，比較起來，我們(A 方)選擇 Gnode3(B 方)會比較有利，因爲我們選了 Gnode3(B 方)，對方有三分之二的機會走進 Wnode3(A 方)或 Wnode4(A 方)，所以 Gnode1(B 方)的致勝機率是 50%，而 Gnode3(B 方)有 66%。依這樣的方法我們可以爲所有的 GeneralNode 找到最好的一條路。

你可能有這樣的疑問，在所有的盤面中有沒有可能有一個 GeneralNode 的下一步，全部都是對方的 WinNode？事實上是有的，而且還對我們的研究產生某些影響，因爲這個原因，原來的 LoseNode 及 WinNode 個數就增加了。我們將在下一段中詳細介紹。

6. 所有的盤面中有沒有可能有一個 GeneralNode 的下一步，全部都是對方的 WinNode？

我們在爲每一個 GeneralNode 找最佳下一步時，的確發現 36736 個盤面中，有 48 個 GeneralNode 的 Child Node 全部都是對方的 WinNode，所以這 48 個特殊盤面的價值就跟 LoseNode 是一樣的，只要遇到這樣的盤面，只要我們下任何一步，只要對方不放水，我們就輸定了。48 個盤面，去除對稱性，只取一方的話，就是有三個盤面，以下把這三個盤面列出。

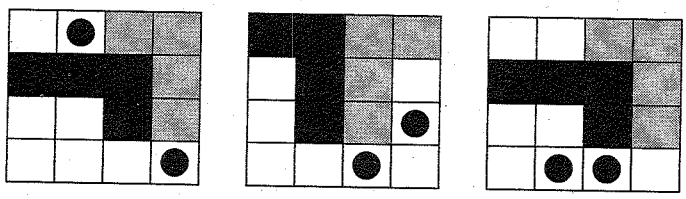
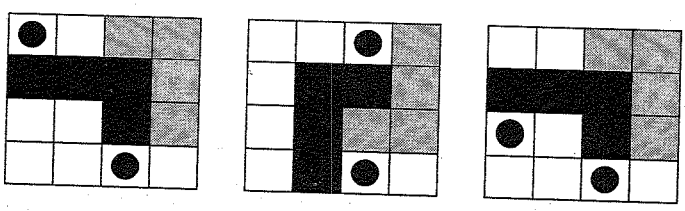


圖 9.　三種特殊的 GeneralNode，輪到灰色的一方要下手，但不管怎麼走，都輸定了。

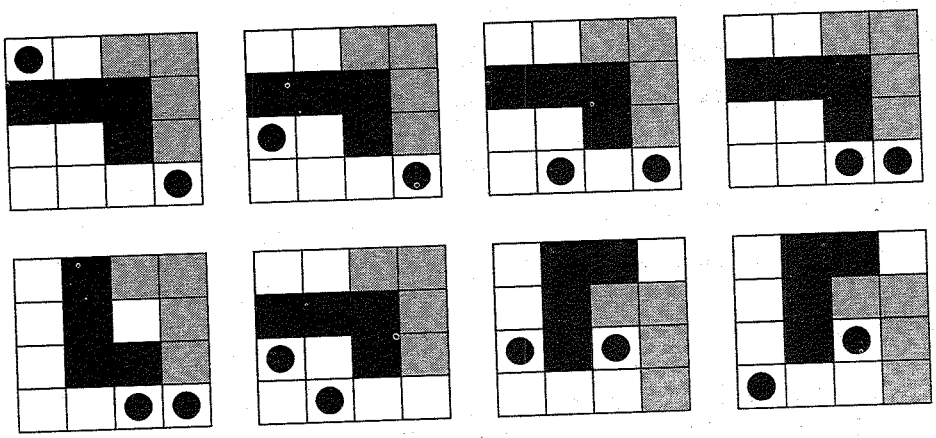
這三個盤面還有一個相同的地方，就是灰色的一方只有一個位置可以放，配合上圓形棋子，總共會有 13 種可能的下棋法，13 個方法都是必輸走法。

如果我們把雙方的這三個盤面及其對稱也當成 LoseNode(等於現在有 240+48=288 個)，再將原來的 Game-Tree 重新整理(因爲這 48 個特殊 Node 原來被當成 GeneralNode，現在被當成是 LostNode，所以 WinNode 的個數會改變，Game-Tree 的分析也會改變)。整理後又發現 48 個原來被當成 GeneralNode，現在因爲 Game-Tree 的分析改變，所以他們的 Child Node 變成全部都是對方的 WinNode。以下再把這三個盤面(去掉對稱並只取灰色的一方)列出。



這三個盤面跟剛才的三個盤面有一樣的特性，都是會有 13 種可能的下棋法，13 個方法都是必輸走法。我們再把這三個盤面及其對稱也當成 LoseNode(等於現在有 240+96=336 個)，再送進 Game-Tree 做分析，發現又出現了 80 個 Node 擁有之前發現的特性，不禁讓我們猜想，會不會就這樣一直把新發現的 Node 加入 LoseNode 中，就可以找到所有盤面的最佳解了呢？很可惜的，答案並非如此。我們總共用這樣的方法找到了 224(48+48+80+48)個新的 LoseNode，也因爲如此，WinNode 的個數也從雙方各 6144 個，變成 8048 個。多出來的 LoseNode 表示雖然不會馬上輸，但是一定會輸；相同的，多出來的 WinNode 表示是一

個陷阱，對方只要踏進來了就遲早會輸。以下將除了剛才列出的六個特殊盤面(6*2*8=96=48+48)以外的八個擁有一樣特性的盤面：(八個是因爲 224 種去掉 96 個之後去掉對稱性，並只取灰色的一方)



從這 14 種特殊的盤面，我們不難看出他們的相似性，就是大部份輸的一方的 L 形棋子都放到棋盤的角落，不然就是因爲圓形棋子擺得不好所以產生的破綻，又再次呼應了之前所歸納出來的下棋策略。

四、 結論

1. L-Game 沒有必贏的策略，但是如果位置下錯，勝負將立即分曉。L-Game 對於人類而言是一個極需思考周全的的遊戲，因爲既使能記下所有的失敗盤面，在實際對奕時，常常因爲盤面的對稱，而不小心走到錯誤的盤面。
2. L-Game 先下手的一方與後下手的一方，擁有一樣多的對應盤面，所以先後並沒有差別。
3. 盡量不要把自己 L 形棋子放在棋盤的角落，80%的失敗盤面都是因爲 L 形棋子卡死。圓形棋子要和自己的 L 形棋子接觸，讓直角內側不被對方攻擊。相同的道理，要把自己的 L 形棋子放在對方的直角部份內側，會有較大致勝的機會。這是從所有的輸棋的盤面(包括遲早會輸的特殊盤面)分析出來的結果。
4. 我們成功的以 Game-tree 的方式，以電腦找出 L-Game 所有的盤面，並加以分析。我們將我們的研究成果設計成 CGI 程式放在網頁上，提供大家上網挑戰我們的 L-Game 程式，不過我們的程式是不會輸的唷，不信你可以試試看。(請看附錄一中 CGI 程式部份執行畫面。(http://www.ice.ntnu.edu.tw/~u84340/L-game/)
5. 發現雙方各擁有 18368 種可能的盤面，其中有 6144 個盤面是必贏，120 個盤面必輸。加上一些特殊的 Node(遲早會輸的盤面，每一方有 112 個)，所以只要雙方都沒有走進對方

的 WinNode，就會持續纏鬥下去，直到有一方走錯。

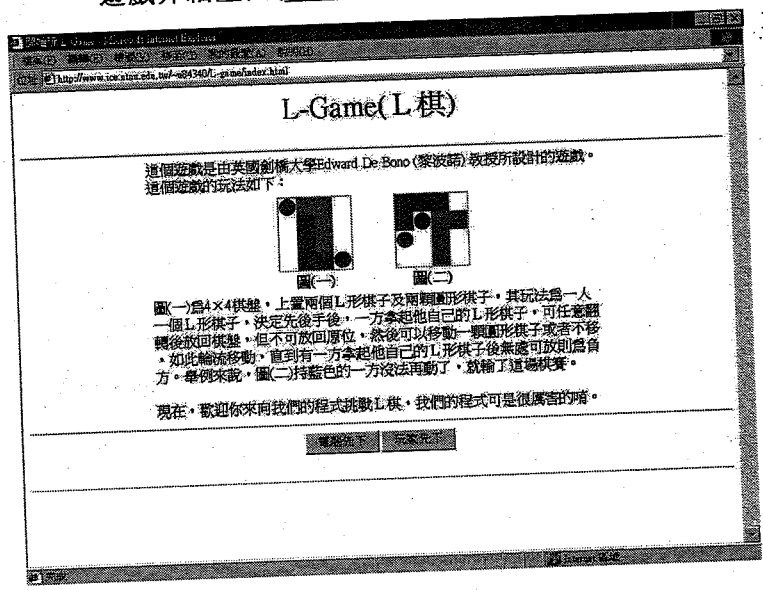
6. 對於盤面記錄的方式，未來期望能找到更好的資料結構，或許也可用於其它棋類遊戲的研究。因爲雖然目前的電腦速度及容量都不錯，但是面對棋類遊戲的盤面記錄，仍需要有較好的資料結構與演算法，讓執行的效率更好。目前有一種想法是針對兩顆中立的圓形棋子的紀錄方式做改變：我們知道，當兩個 L 型的棋子位置確定後，兩個圓形的棋子就只有 28 種可能的位置，目前的紀錄方式是用兩個 0 到 6 的數字(n1,n2)來記錄，事實上由於 0≦n1≦n2≦6，所以如果我們把(n1,n2)的紀錄方式改成一個變數 round 來記錄(0≦round≦27)，就可以節省接近一半的記憶體空間(原來用兩個變數，需要 49 單位記憶體，現在只要 28 單位就可以了)。

五、參考文獻

1. Edword De Bono (1988) 余阿勳 譯. 水平思考五日訓練法. 水牛出版社.
2. Grimaldi Ralph P. (1994). Discrete and combinatorial mathematics : an applied introduction. 3rd ed.
3. Ronald E. Walpole & Raymond H. Myers (1993). Probability and statistics for engineers and scientists, 5th Edition, by Prentice-Hall, Inc.
4. Wackerly & Mendenhall & Scheaffer (1996). Mathematical statistics with applications, 5th Edition, by Wadsworth Publishing Company.
5. David Kincaid & Ward Cheney (1996). Numeriacl analysis : mathematics of scientific computing, 2nd ed.
6. Harry R. Lewis & Christos H. Papadimitriou (1998). Elements of the theory of computation, 2nd ed.
7. Neapolitan & Naimipour (1996). Foundations of algorithms, by D. C. Heath and Company.
8. Horowitz & Sahni (1994). Fundamentals of data structures in Pascal, 4th Edition, by Computer Science Press.
9. Gilbert Strang (1988). Linear algebra and its applications, 3rd Edition, by Harcourt Brace Jovanovich, Inc.
10. 冼鏡光 (1990). 名題精選百則 技巧篇 - 使用 C 語言。格致圖書公司。

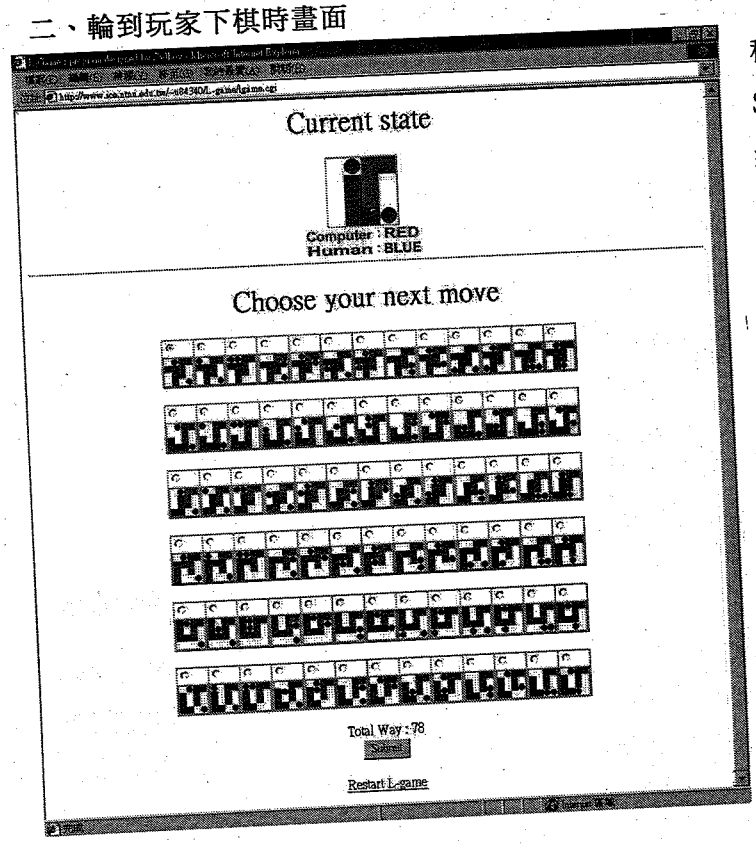
附錄一　L-Game 的 CGI 程式執行畫面

一、遊戲介紹畫面(ttp://www.ice.ntnu.edu.tw/~u84340/L-game/index.html)



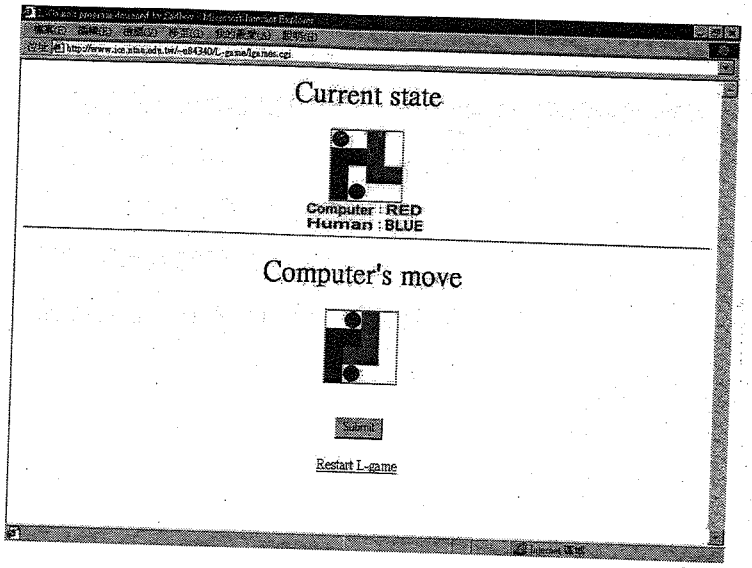
本畫面簡單對 L-Game 做介紹，並提供使用者選擇要先下還是讓電腦先下。選擇之後會將不同的參數傳入我們所設計的 CGI 程式 lgame.cgi 之中，並開始玩遊戲。

二、輪到玩家下棋時畫面



程式列出目前的盤面（Current State），以及所有可能的下一步盤面供玩家選擇，選擇完後按下 Submit 鍵，會將玩家的選擇傳出，並且輪到電腦下。電腦會針對使用者所選擇的盤面尋找最佳的對應盤面，並秀出電腦的下一步（畫面詳見下一頁）。

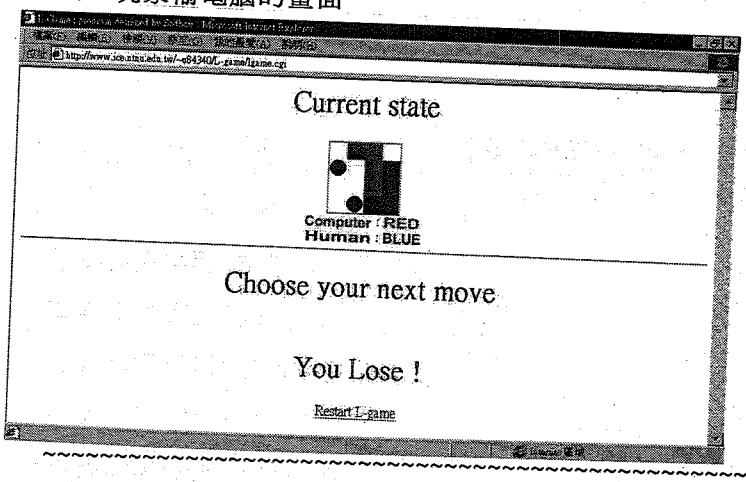
三、輪到電腦下棋時畫面



程式列出目前的盤面（Current State），以及電腦所選擇的下一步盤面，玩家可以從電腦的琪法中看出電腦的對應方法，玩家看完後按下 Submit 鍵就輪到玩家選擇下一步。

如果電腦發現自己沒有辦法走下一步了，就會印出 You Win!，告訴玩家已經勝利，但是我們的程式，玩家是不可能玩到讓電腦沒路走的☺。

四、玩家輸電腦的畫面



輪到玩家時，如果電腦發現玩家沒有下一步可以選擇時，就秀出 You Lose！，告訴玩家已經輸了，玩家可以選 Restart L-game 重新再玩一次。

申請 89 會計年度中小學科學教育計劃專案補助相關資訊報導

編輯室

配合新的會計年度的計算期間（89 年 1 月 ~ 89 年 12 月），89 會計年度涵蓋 88 年 7 月－89 年 12 月，長達 18 個月，係權宜措施。教育部預定本年六月初研修 89 會計年度（88 年 7 月－89 年 12 月）中小學科學教育計畫專案補助申請實施要點，修訂後即會擇期公告申請，敬請有意申請之中小學科學教育專案之教師等待並預先做好申請準備。