國立交通大學

資訊學院資訊科技（IT）產業研發碩士班

碩 士 論 文

麻將人工智慧之研究

The Study of Mahjong Artificial Intelligence

研 究 生：林典餘

指導教授：吳毅成　教授

中 華 民 國 九 十 七 年 八 月

麻將人工智慧之研究

The Study of Mahjong Artificial Intelligence

研 究 生：林典餘　　Student：Tien-Yu Lin

指導教授：吳毅成　　Advisor：I-Chen Wu

國 立 交 通 大 學

資訊學院資訊科技（IT）產業研發碩士班

碩 士 論 文

A Thesis

Submitted to College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Industrial Technology R & D Master Program on

Computer Science and Engineering

Aug 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年八月

麻將人工智慧之研究

學生：林典餘　　指導教授：吳毅成

國立交通大學資訊學院產業研發碩士班

摘要

麻將為資訊不明確、多玩家類型的遊戲，除了無法得知另外三家對手的手牌以外，隨著牌局的進行，玩家可能會根據摸牌而轉牌，更增加了遊戲的變化性。

在本篇論文之中，我們根據麻將遊戲的特性來設計 AI，除了根據我方的手牌，以減少上聽數為首要準則，決定如何捨牌；另外，隨著牌局進行到中、後期，在無法確定安全捨牌時，我們試圖使用 Monte-Carlo 方法模擬遊戲進行，避開有可能導致輸牌的捨牌，以追求最終贏牌為目標。

The Study of Mahjong Artificial Intelligence

Student: Tien-Yu Lin Advisor: Dr. I-Chen Wu

Industrial Technology R & D Master Program of
Computer Science College
National Chiao Tung University

ABSTRACT

Mohjong is an imperfect information game, and multiplayer game. Each player couldn't get other players' tile information. During game processing, each player might change tiles to make better combinations. And it makes game more unpredictable.

In this thesis,we analyze Mohjong's features, and try to design a smart computer program. We try to reduce count-to-listen number to decide which tile be discarded is better. When we detect other player might in listen condition, would start to use Monte-Carlo method to simulate the game to the end. We want to use the method to make a better decision to avoid to lose game.

誌謝

首先要感謝我的指導教授，吳毅成博士，由於他細心的指導，這篇論文才能順利完成。

接著，要感謝群想公司的 CYC 麻將子系統，這套系統提供了一個平台，讓我在撰寫程式時，專注於遊戲的特性及 AI 設計，省去許多非研究相關的程式開發。特別感謝同學高偉傑幫忙修改這套平台，讓麻將 AI 能夠在此平台順利測試。

此外，感謝學長林秉宏、同學曾汶傑提供很多好的程式設計方式，讓我學到很多程式撰寫技巧。而且要感謝同學李榮欽、學弟林宏軒，自己本身對麻將這遊戲並不擅長，但每次與他們討論時，都能學到許多寶貴的經驗及技巧，原本看書只能一知半解，經過討論後更能想出適合的方法實作出來。

還要感謝實驗室的同學林哲毅、吳秉儒、周鼎量、陳威霖及所有學長學弟們，不論是研究上或生活中，每當我遇到挫折，大家總會給我關懷和鼓勵，真的非常感謝，讓我能有這段寶貴的求學經歷。

最後，我要感謝我的父母親、哥哥和姊姊，無時無刻都給我最大的支持與鼓勵，家也是最讓我感到溫馨的避風港。謹以此論文，獻給我摯愛的家人。

目錄

圖表目錄

第一章、緒論

本章首先會在 1.1 節簡介麻將相關的背景知識，介紹一般十六張麻將的玩法及規則。1.2 節會介紹人工智慧將不同遊戲的分類方法，並探討麻將是屬於哪種分類及其特性。1.3 節中為蒐集麻將過去相關的研究，以及同類型遊戲 AI 相關的設計方法。在 1.4 節，為簡介 Monte-Carlo 的運作原理，及應用於 AI 領域的成功案例。1.5 節為問題及研究目標。最後在 1.6 節簡介本論文的內容大綱。

1.1 麻將介紹

麻將可以稱的上是我國的國粹，逢年過節家家戶戶幾乎都會摸上幾圈，甚至曾有看過報導，中國也正積極的爭取想把麻將變成奧運的比賽項目。

追溯麻將的起源[3]，也可以找到許多不同版本的趣事，像是相傳元末明初萬秉迢先生因推崇水滸英雄而發明麻將，將萬索筒共計 108 張牌對應到 108 條好漢；另有此一說，麻將源於江蘇太倉，古時設立糧倉儲糧，而麻雀是害鳥，因此設立了一個捕雀制度，麻將就是由此演變而來。

台灣十六張麻將，據說是民國初年由廣東麻將演變而來，起手每位玩家拿進十六張牌，並擁有一些不同的台數規定，而不同版本的麻將規則，玩家考量的思考決策也大不相同。

1.1.1 台灣十六張麻將介紹

麻將的牌類共分為萬子、索子、筒子、字牌及花牌，共計 144 張牌（如圖 1-1）。

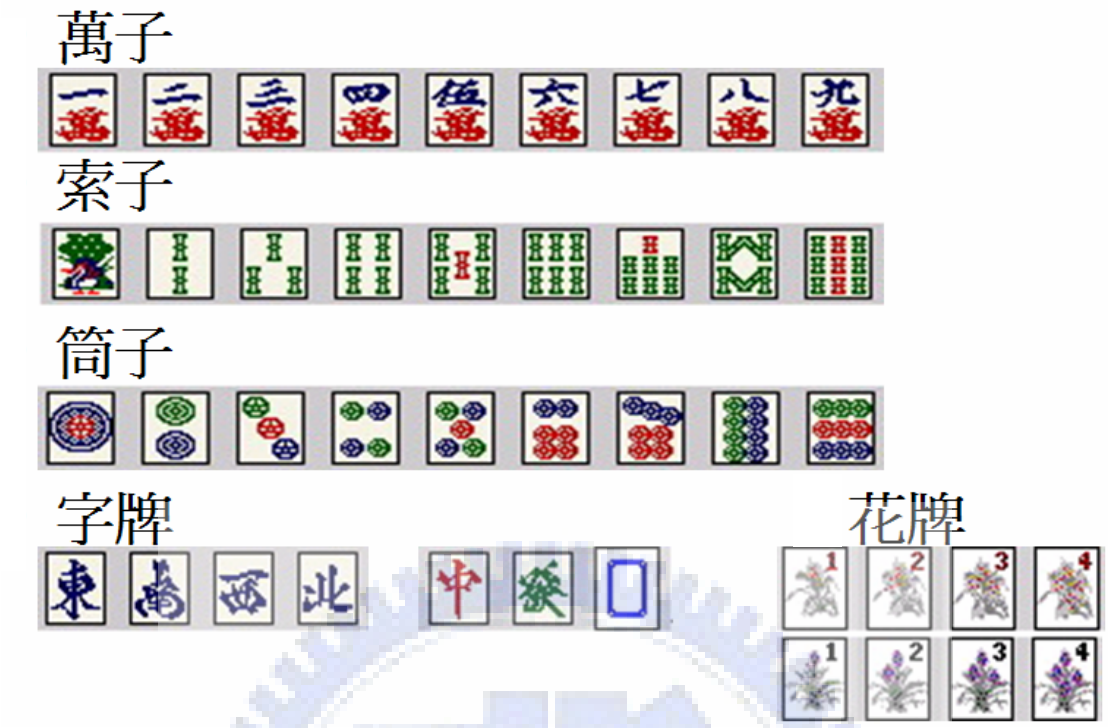

圖 1-1 麻將所有的牌

1.1.2 麻將進行方式

麻將規則簡介如下[2][3]，遊戲是由四位玩家一同來同樂，一開始會先取出東風、南風、西風及北風四張牌翻成背面，再由玩家隨機摸取決定座位，摸到東風的暫定為假東，之後遊戲也依照此座位逆時針的方向進行。

每場遊戲開始前，玩家先把牌蓋起來並兩個一疊，共疊成十八堆推至手前。第一場開始時，由假東先擲骰子決定真正的莊家，之後若由莊家獲勝，將繼續連莊直到其他玩家獲勝為止，莊家若獲勝會有額外台數的加分，相對的輸牌也會有額外的台數扣分（我們設計的 AI 排除額外台數的加減分），莊家替換時也是依照逆時針的順序。

之後每場都由莊家擲骰子，決定從何處開始摸牌，大家依照著順時針拿進四張牌，共拿四輪取得十六張手牌。遊戲進行中，若摸進花牌，將由牌堆的尾部補牌。

莊家會開門(再摸進一張牌)並決定捨牌，玩家依照逆時針的順序摸牌，並決定丟出一張牌，遊戲一直進行到有人達成贏牌條件稱為胡牌，或者摸牌至末端只剩下 16 張牌稱為流局。

然而，遊戲中除了摸牌以外，還有一些特殊的打牌規則，分別介

紹如下：

1. 吃牌

我方手中有兩張鄰近的牌(相連或間隔一張)，若上家打出可湊成三張連續的牌，我們可以選擇吃牌，並推至手前攤開。如圖 1-2，我方手裡有二索、三索，這時若上家打出一索或四索，我們可以選擇吃牌。

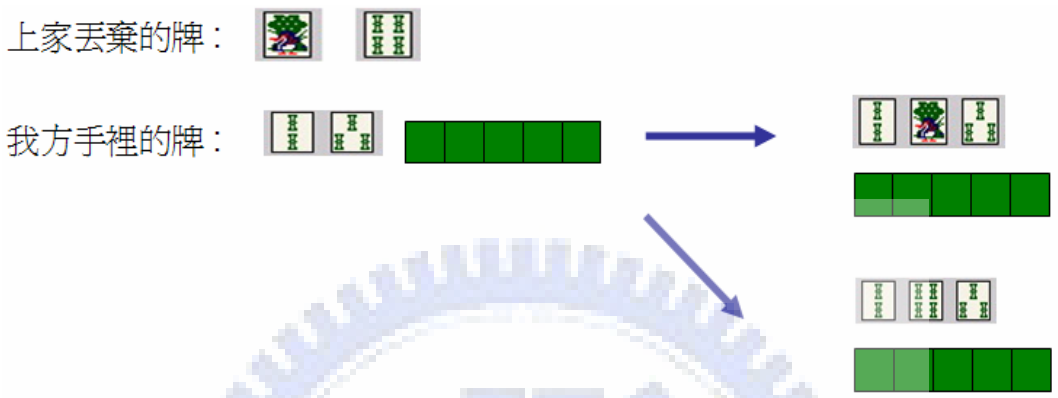

圖 1-2 上家若是打出可湊成連續的牌，玩家可選擇吃牌

2. 碰牌

若我們手中有兩張相同的牌，任意一家打出同樣的牌，我們可以選擇碰牌，並推至手前攤開。

圖 1-3 任意一家打出我方手中擁有兩張的牌，玩家可選擇碰牌

3. 槓牌

槓牌又分為明槓、暗槓和補槓，分別介紹如下：

a. 明槓 – 若我們手中有三張相同的牌，其他家(不包含上家)打出同樣的牌，我們可以選擇槓牌，並推至手前攤開，從牌尾摸進新牌，如圖 1-4(a)。

b. 暗槓 – 若我們手中有四張相同的牌，我們可以選擇槓牌，並

推至手前不須將此組攤開，從牌尾摸進新牌，如圖 1-4(b)。

c. 補槓 – 若是我們摸進已碰過的牌，我們可以選擇槓牌，並將此牌放進碰的牌堆，從牌尾摸進新牌，如圖 1-4(c)。

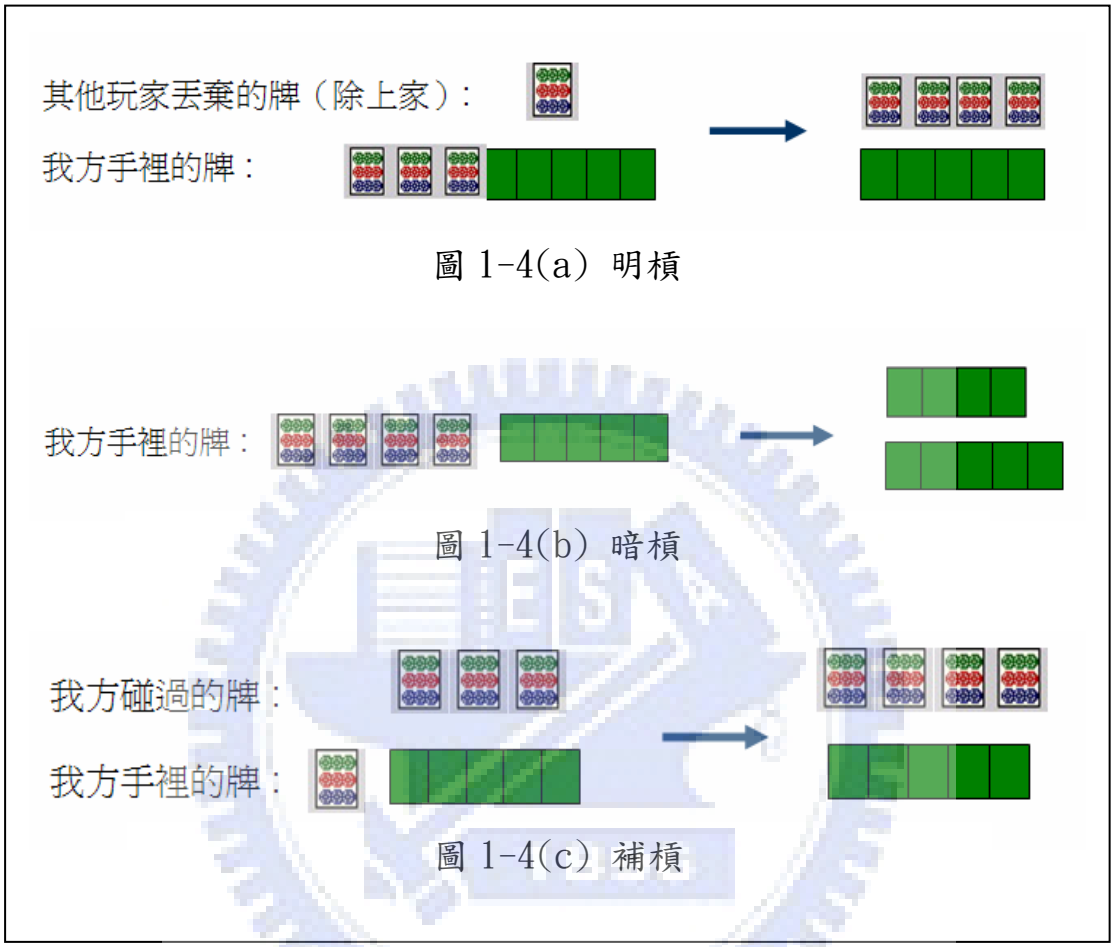

圖 1-4(a) 明槓

圖 1-4(b) 暗槓

圖 1-4(c) 補槓

1.1.3 贏牌條件

若是三張相同的牌，我們稱其為刻，若是三張同花色連續的牌，我們稱為順，兩者都視為完整的一組；另外，由兩張相同的牌我們稱其為眼，麻將的贏牌條件為，第一個湊成五個完成組和一對眼的玩家。

若是玩家捨出的牌，造成其他對手達成贏牌條件，捨出此張牌的玩家稱為放槍，我們會在之後的實驗去統計玩家的放槍次數，並希望能使用 Monte-Carlo 去避免放槍。

1.2 遊戲分類

人工智慧是試圖使用電腦去模擬人們的思考模式，因此針對不同的應用，設計上也會有所差異，在遊戲 AI 這塊領域會依據遊戲的特性去分類，不同類型的遊戲，設計上也會不逕相同，我們分析麻將的遊戲特性，並試圖將其歸類於下面的遊戲分類。

1.2.1 明確與不明確資訊遊戲

我方可以完全掌握遊戲所有的資訊，稱為明確資訊遊戲(Perfect Information Game)[11][12][19]，如：象棋、西洋棋，我方可以知道對方所移動的位置，清楚的掌握整個盤局狀況來下決定；相對的，如：橋牌、麻將，我方只擁有自己手牌的資訊，而對手的手牌及部份牌局資訊是無法知道的，這類遊戲屬於不明確資訊遊戲（Imperfect Information Game）。

1.2.2 雙玩家與多玩家遊戲

遊戲進行時，只有兩位玩家參與的遊戲為雙玩家遊戲(Two-player Game)，如象棋、西洋棋，兩位玩家彼此一爭勝負；相反地，如：橋牌、麻將，玩家人數大於兩位，即為多玩家遊戲(Multi-player Game）。

1.2.3 麻將屬於不明確資訊、多玩家遊戲類型

根據上面的分類方式，由於我方只能掌握自己手牌的資訊，而無

法得知其他人的手牌，因此，屬於不明確資訊遊戲；且麻將必須由四位玩家才能進行，屬於多玩家類型的遊戲，不明確資訊遊戲包含了未知的訊息讓 AI 難以去審局，四位玩家不同的決策都會對牌局造成影響，讓遊戲更難以預測。

1.3 麻將過去之研究

市面上有許多麻將教學的書籍[1][2]，整理出打牌的技巧，就像象棋的棋譜，讓玩家能夠遵循這些原則，增進打牌的技巧，這些打牌技巧是前人的心得，也都有他們的道理。

然而，有關麻將的學術論文，卻是非常稀少的，找到的論文也比較偏向麻將遊戲推廣為主，對於 AI 的設計方式比較少描述[18]。因此，我們去尋找類似麻將的遊戲橋牌[6] [13][14][20]，兩者同為不明確資訊、多人遊戲類型，橋牌可分為兩個階段，第一階段為叫牌階段，而第二階段為打牌階段。Ian Frank 對橋牌有深入的研究及整理[9][10]，叫牌階段會依照高手玩家叫牌方式來統計，設計出來已有專家的水準表現，由於不明確資訊讓牌局很難預估，使用策略的方式去評估，然而打牌階段階段的 AI 尚未有適合的解法，目前還只有業餘玩家程度的能力。

國內師大的林順喜教授和學生莊凱閔和陳玥汝，對於麻將 AI 也有研究[21]，透過計算勝率的方式來判斷捨不同牌的好壞，論文中有提到雖然很難評估用這種機率計算方式來當審局函數的準確性，不過，跟麻將高手下過，也有接近高手的打牌能力。

1.4 Monte-Carlo應用於人工智慧

本節首先在 1.4.1 節簡介 Monte-Carlo Method 的概念，如何應用於 AI 這個領域。在 1.4.2 節介紹兩種 Monte-Carlo 成功的應用，分別為 Go 和 Phantom Go。

1.4.1 Monte-Carlo的概念

拜電腦快速發展之賜，CPU 運算速度越來越快，對於 Monte-Carlo 這種需要大量模擬才會準確的演算法，近年來開始越來越受到重視。Monte-Carlo[4][5]是使用亂數產生許多下法，再透過大數法則，來決定最有可能的最佳解，如圖 1-5。

共分為四個階段，反覆執行到時間結束為止，分別簡述如下：

1. Selection :為選擇要下的節點。依照不同的應用可以使用不同的選擇方式。
2. Expansion :根據第一步驟選擇出來的點，開始擴展此一節點。
3. Simulation :模擬遊戲進行到最後分出勝負或和局。
4. Back propagation :將每次模擬出來的結果，回傳更新上面的節點。

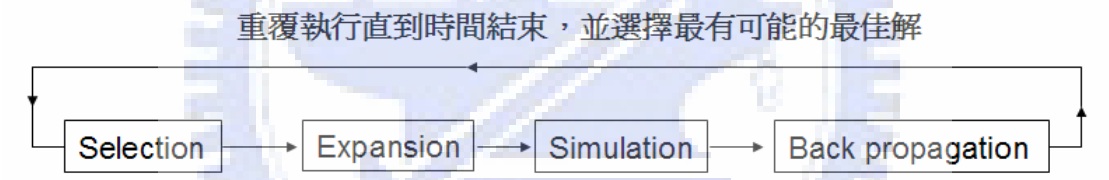

圖 1-5 Monte-Carlo 的運作方式。

1.4.2 Monte-Carlo的應用

由於圍棋的決策除了考慮吃子、佔地及種種不同的策略，很難找到一個適合的審局函數[7]，MoGo[15][16][17]就是一個成功使用 Monte-Carlo 應用的 AI，使用 Monte-Carlo 模擬棋局的進行，再加上一部分的 Domain Knowledge，使 MoGo 在下 9 路圍棋具有段位的水準。

另一個圍棋的變形遊戲為 Phantom Go[8]，我們先簡介這遊戲的規則，這是一個不明確資訊遊戲，，如圖 1-6，黑、白子雙方都只能看到自己本身棋子的位置，只有裁判看得到另一個包含黑、白子雙方的棋盤，輪到黑方要下子時，若他決定要下的位置為空，這時可直接

下，但是，若這個位置已經有白色下過，這時裁判會提醒黑方無法下這個位置，並要求黑方下別的位置(這時黑方棋牌可在該位置用白子標示)。

Monte-Carlo 應用於這個遊戲的方式，就是去模擬對方可能下的位置，接著再模擬遊戲進行到最後，經由大數法則，於時間耗盡時，決定最有可能的最佳解。

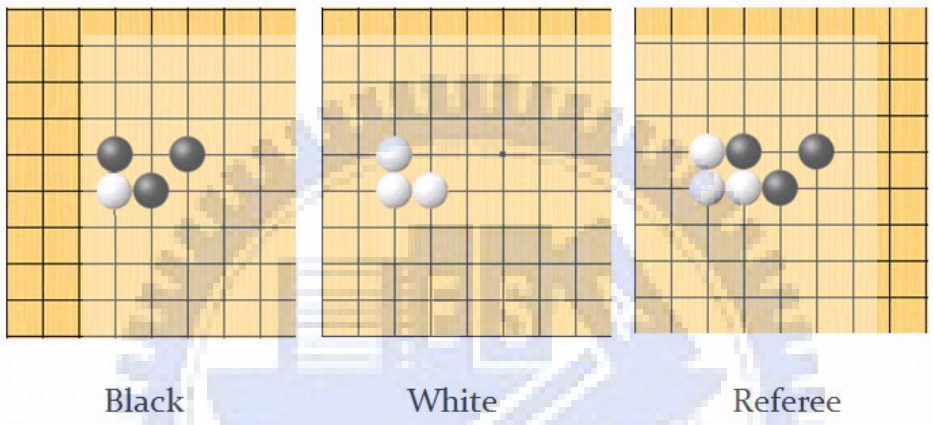

圖 1-6 Phantom Go

1.5 研究目標

基於以上的背景，我們想要根據麻將的特性設計出聰明的 AI，針對這種不明確資訊的遊戲，試圖能找到一個合適的方法，來模擬人類的打牌策略。

我們的 AI 有以下兩個研究目標：

1. 忽略麻將的台數規則，追求最多的胡牌次數。

麻將複雜的台數規則增加了遊戲的趣味性，不過，對於這個全新未知的領域，我們希望能先專注麻將整體打牌的架構，不考慮特殊的胡牌方式，以及莊家或連莊的額外分數加成，單純以追求最多的胡牌次數為目標。

2. 無法找到安全牌時，使用 Monte-Carlo 來模擬避開放槍。

遊戲中後期，判斷生死張是很重要的技巧[1][2]，新手玩家不會評估，以追求自己能聽牌、胡牌為目標，拿到沒用的牌，即使是生張也不猶豫地丟出，贏牌全憑運氣；而中手能夠判斷生死張，但有時為了避開放槍，判斷錯生死張而拆牌，只是讓自己損失贏牌的機會。高手的話，會保留可能的生死張，並試圖把手牌轉換成聽此張牌。

因此，當遊戲進行到中後期，我們希望透過 Monte-Carlo 來模擬對手的手牌，模擬遊戲進行到最後，根據大數法則的結果，選擇放槍機率最低、贏牌機率高的為捨牌，同時兼顧不放槍，且仍朝著最終贏牌的目標前進。

1.6 本文大綱

本論文第一章介紹麻將相關的背景知識，並提出這篇論文的目標，研究設計麻將 AI 的方法，並避免放槍。第二章將針對麻將 AI 程式設計的方法做詳細的說明。第三章則深入介紹本論文如何利用 Monte-Carlo 來模擬對方手牌，並避開放槍的捨牌。第四章對本論文所作的實驗結果做分析與討論。最後第五章針對本論文的結果作總結，並進一步討論未來的發展方向。

第二章、麻將人工智慧設計

由於本論文會使用到一些麻將的專有名詞，因此本章一開始將於 2.1 節介紹麻將的上聽數計算方式，在 2.2 節介紹有效牌，及詳細介紹設計時所注意到的特性。在 2.3 節描述如何將手牌依照花色分類，各自搜尋及結合。最後，在 2.4 節為了之後使用 Monte-Carlo Method 模擬牌局進行，我們使用了一些方式加速遊戲 AI 的決策。

2.1 上聽數介紹

本節首先在 2.1.1 節簡介上聽數的計算方式。接著在 2.1.2 節說明本論文如何使用麻將上聽數來決策，選擇捨牌的方式。

2.1.1 上聽數

由於麻將是由五組完整的牌組（刻或順），再加上一組眼為胡牌，因此，若手牌再進一張牌，即可滿足胡牌條件時，我們稱為聽牌階段。

上聽數就是計算距離聽牌階段還差幾張牌，如圖 2-1(a)的例子，我方手牌已經有四組完整的組及一對眼，這時已經是聽牌階段，所以為 0 上聽。

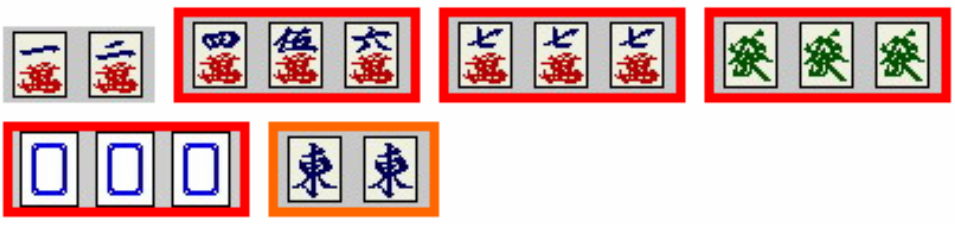

圖 2-1(a)　0 上聽

圖 2-2(b)的例子中，我方再進一張三萬或六萬，即可前進到聽牌階段，因此，此為 1 上聽。

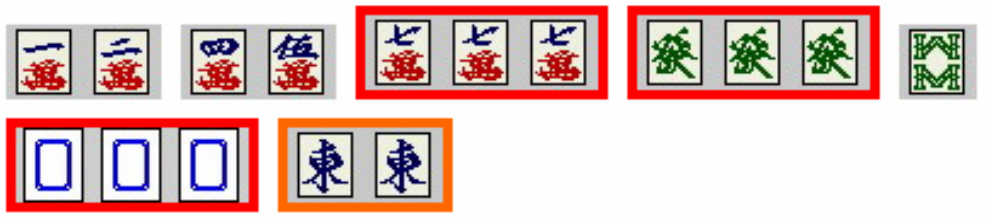

圖 2-1(b)　1 上聽

2.1.2 減少上聽數

麻將有句術語：「早聽不一定早胡，晚聽一定不會早胡。」這句話的含意是，如果能夠早點達到聽牌階段，大多是以追求能早點聽牌為目標，即使聽的不好，也能隨著之後的進牌來轉牌。

我們設計的麻將 AI 也是遵循這個道理，在不考慮安全牌的狀況下(通常為遊戲前期，其他對手尚未聽牌階段)，我方決定捨牌時，會將手牌逐一丟出，並去計算剩餘牌的上聽數，以追求最小上聽數為主要方針，越快達到聽牌階段，贏牌的機率自然也增加。

然而，有時丟出不同的兩張牌，能夠取得相同的上聽數，這時我們會參考有效牌來決定捨牌，將會在下節詳細介紹有效牌的計算方式。

2.2 有效牌介紹

本節首先在 2.2.1 節定義何謂有效牌。接著在 2.2.2 節說明有效牌之間的聯集關係。2.2.3 節說明牌堆裡剩餘牌的張數對有效牌實際效應的影響。2.2.4 節說明吃、碰兩種類型有效牌，具有不同的比重。最後在 2.2.5 節描述有效牌實際效用的評估方式。

有效牌

兩張同花色有關聯的牌(可能為鄰近或相同)，稱為一搭。新進一張牌，能讓這搭變成完整的一組，稱為有效牌。

如圖 2-2(a)，對於二、三萬這搭，一、四萬為有效牌。如圖 2-2(b)，對於二、四萬這搭，三萬為有效牌。如圖 2-2(c)，對於兩張三筒，三筒為有效牌。

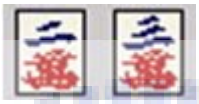

圖 2-2(a)有效牌為一、四萬

圖 2-2(b)有效牌為三萬

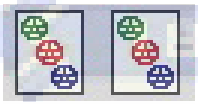

圖 2-2(c)有效牌為三筒

我們除了計算上聽數，也同時會分析各牌組的有效牌，並依照這兩種資訊決定最適合的捨牌。

有效牌之間的聯集關係

我們參考實際打麻將的經驗，對於同花色鄰近的牌，往往能夠透過不同的組合方式，產生不同的有效牌，而這些有效牌必須以聯集的關係來設計，才能正確的評估這副手牌實際的價值。

如圖 2-3，連續的二、三、四、五及六萬，可以視為前三張一組，有效牌為四、七萬，若是以另外一種方法來分割，將後三張視為一組，這時有效牌變成一、四萬，而實際上這些有效牌是聯集的關係，因此，真正有效牌為一萬、四萬及七萬。

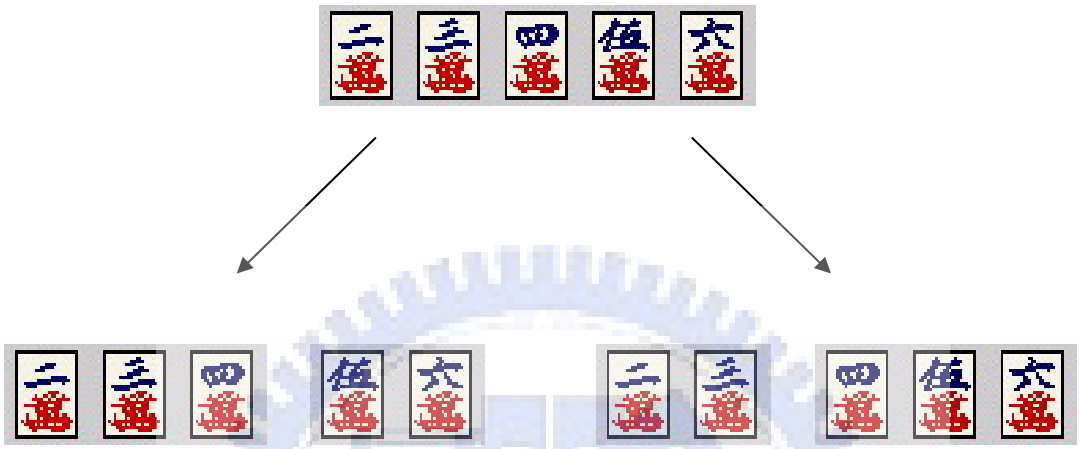

圖 2-3 不同分割方式會得到不同的有效牌，聯集後，真正的有效牌為一萬、四萬及七萬。

牌堆裡剩餘的有效牌張數

我們找到有效牌後，然而這些有效牌並無法直接顯現它實際的效用，如圖 2-4，若我們手牌裡同時有三、四萬及三、四索，我們可得知各自的有效牌分別為二、五萬和二、五索，但是，隨著牌局的進行，這些牌可能會被吃、碰或者丟到海底，如下面的例子中，對家吃了二萬，海底也出現過一張二萬和三張五萬，也就是之後我們還有機會拿到的二、五萬只剩三張；相較之下，二、五索都未出現過，所以，二、五索還可取得的張數為八張，後者機會自然比較大些。

因此，我們同時必須考慮牌堆裡剩餘有效牌的張數狀況，才能實際的顯現這些有效牌的效用。

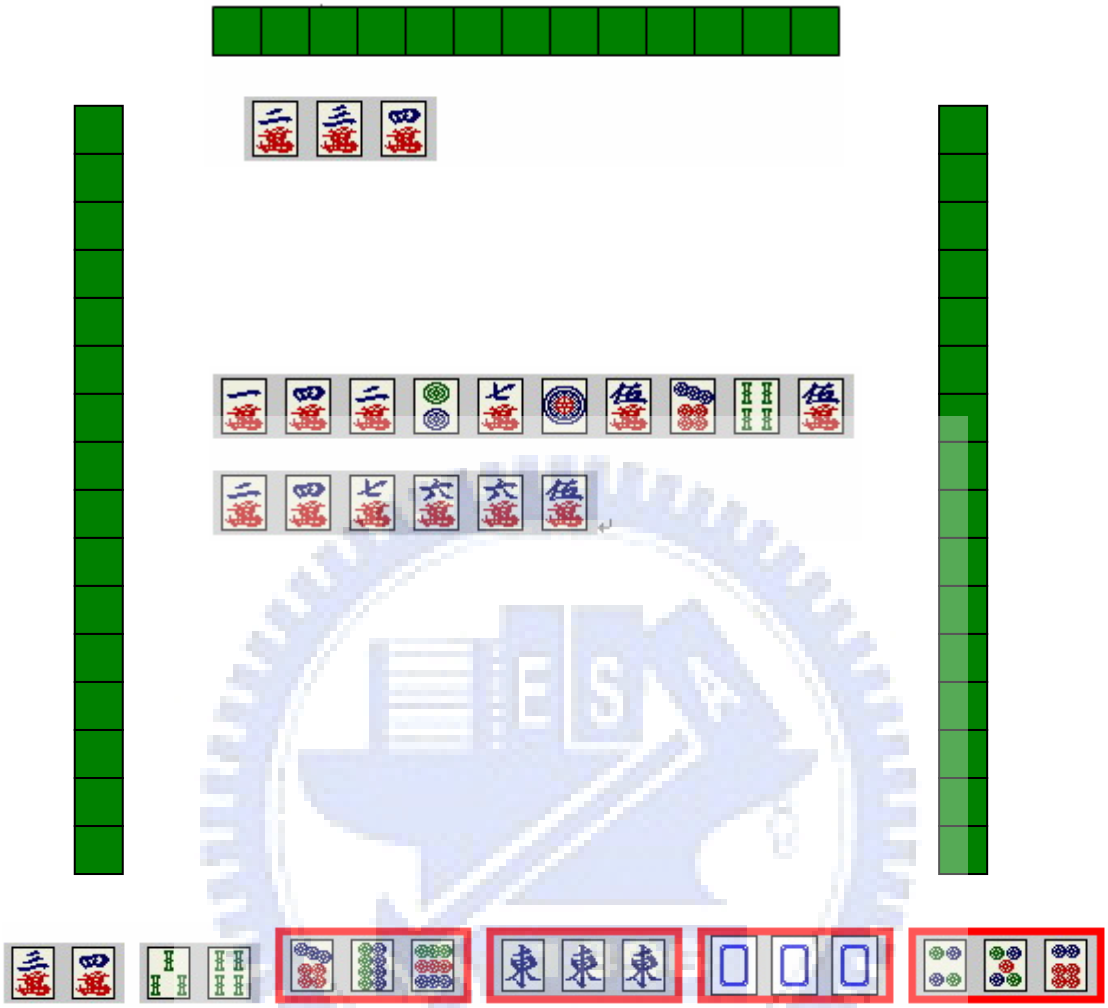

圖 2-4 考慮對手吃、碰牌及海底的牌，才能正確的評估有效牌實際的效用。

吃、碰比重

麻將除了一般的摸牌外，還可以使用吃牌或碰牌的方式進牌，對於同張有效牌，如果是可吃進或可碰進，由於不同的進牌方式，實際上的效用也有差異。

如圖 2-5(a)，我方手牌有一、三萬的搭子，有效牌是二萬，這時可以透過自己摸進二萬，或者由上家打出二萬，我方選擇吃進來完成組，來源有兩處。

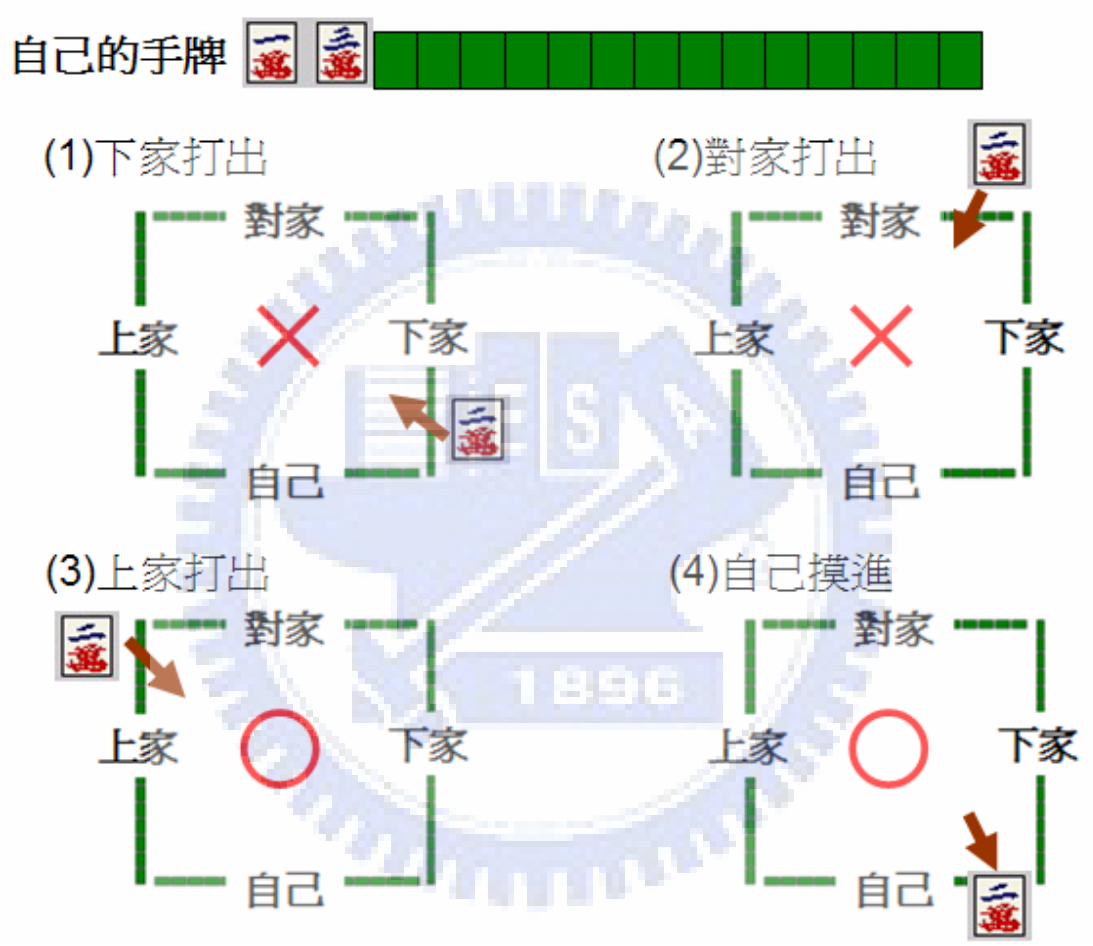

圖 2-5(a) 可吃進的有效牌

相較於 2-5(b)，我方手牌有兩張二萬的搭子，有效牌同樣為兩萬，我方除了可以自己摸進二萬，只要其他玩家打出二萬，我方也可選擇碰進這張牌完成組，因此，來源有四處。

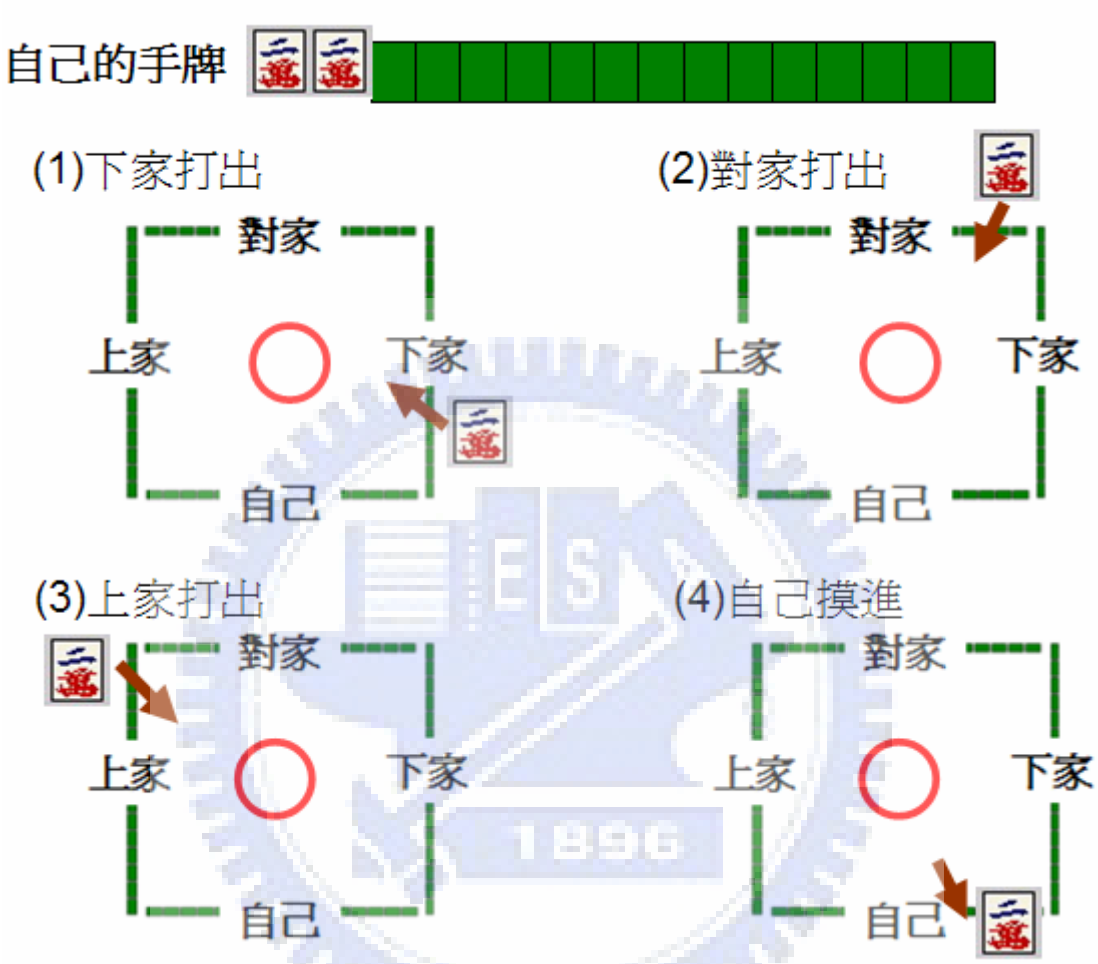

圖 2-5(b)可碰進的有效牌

然而，假設我方在聽牌的狀況下，不管是由下家、對家或上家的捨牌，只要這張牌能讓我方達成胡牌條件，我方都可以胡牌，因此，當聽牌的狀況下，不管是可吃進或可碰進的有效牌，都具有相同效益，如表 2-1 我們把吃、碰比重設為一個參數，可隨不同狀況做設定。

吃碰比重 / 時機	吃	碰
一般階段	2	4
聽牌階段	1	1

表 2-1 不同階段設定不同的吃、碰比重

總結這節所介紹的，我們取得有效牌後，除了考慮牌堆裡剩餘牌的張數，還必須判斷有效牌的類型，根據不同的階段乘上不同的比重，才是這張有效牌實際的影響力。

2.3 搜尋

本節首先在 2.3.1 節描述分析手牌時，遞迴搜尋的方式。在 2.3.2 節說明不同種類的牌可獨立搜尋。最後在 2.3.3 節介紹將不同種類牌組合在一起時，決定不同的眼牌會對上聽數造成的影響。

2.3.1 遞迴搜尋

當我們決定捨牌的時候，這時會依據捨棄哪張牌能有最小的上聽數為判斷準則，因此，為了計算不同牌組的上聽數，先去計算手牌裡的組數和搭數，我們使用遞迴的方式去搜尋，將同一副手牌做不同的分割，並依照表 2-2 的規則來做遞迴搜尋。

種類	例子
順	一二三萬
刻	一一一萬
中洞搭	二四萬
連續搭	二三萬
對搭	二二萬
孤張	一萬

表 2-2 彼此有關聯性的牌。

如圖 2 - 6，假設我方萬字牌為一萬兩張、二萬、三萬及四萬，由萬字牌最小的數字開始搜尋，將符合規則的取出綁在一起，第一層的所有組合包括一二三萬、一二萬、一三萬、一一萬及一萬孤張，剩餘未被框起來的為未處理的牌，接著再繼續往下搜尋，直到所有的牌都被分類，以這個例子來說，一一萬及一二三萬可完成一組一搭，為最好的組合方式。

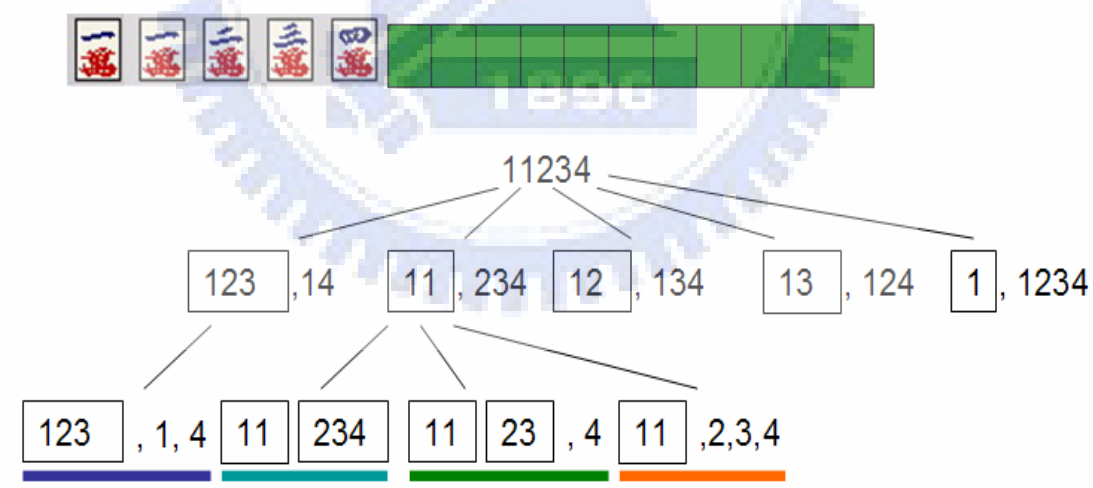

圖 2-6 遞迴搜尋得到最大的組數和搭數。

2.3.2 單一種類牌獨立搜尋

若是我們將手牌整個展開來，會建造出如圖 2-7(a)的樹，而其

中包含許多重複的計算，因此，我們依據玩家實際的打牌觀念，將不同種類的牌分開來考慮，各自計算出組數和搭數如圖 2-7(b)，再做組合取得最小的上聽數。

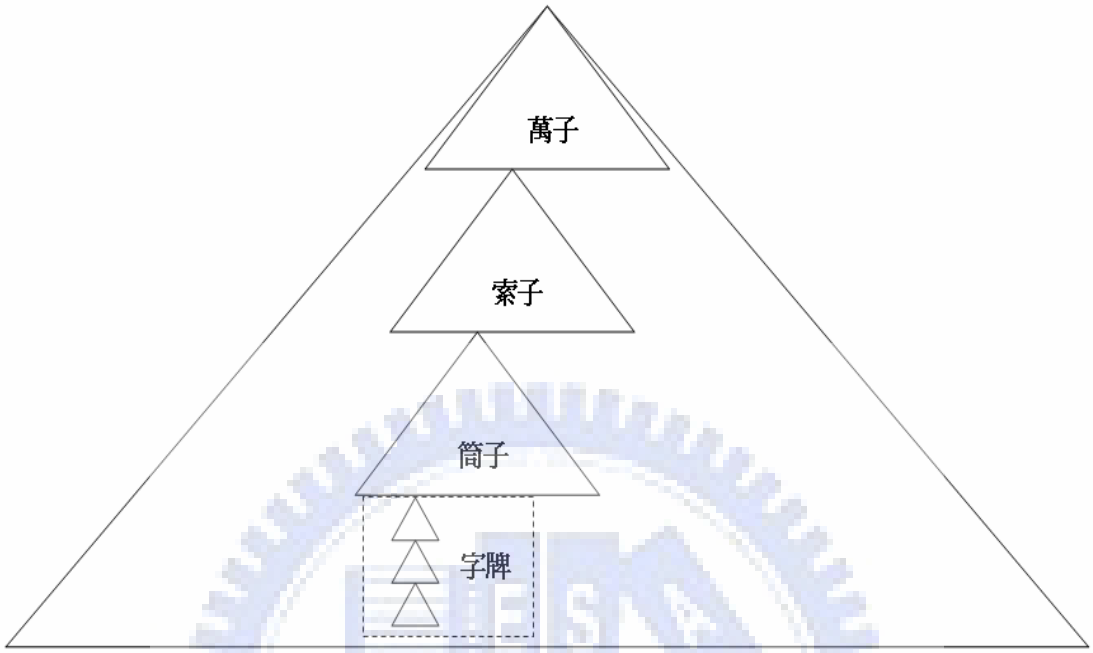

圖 2-7（a） 按照萬、索、筒、字牌順序搜尋下來，完全展開樹。

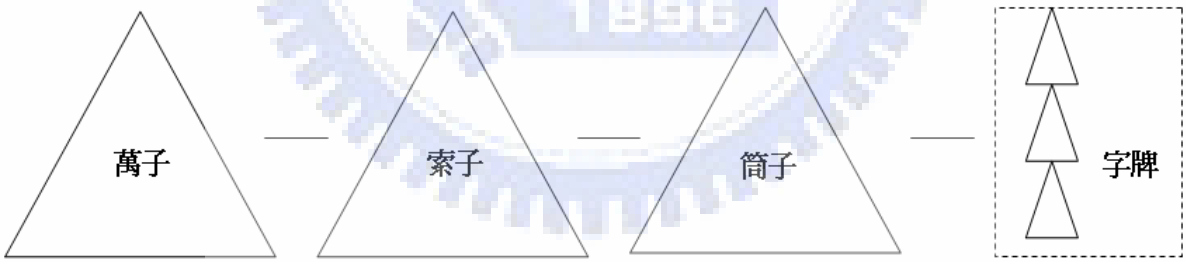

圖 2-7(b) 將各花色牌分開計算再做結合。

我們將各種類牌獨自計算的組數和搭數做結合，加總已完成的組數及有機會成為組的搭數，推算上聽數。如圖 2-8 萬子牌可組成 1 組、1 搭，索子牌可組成 2 搭，筒子牌可組成 2 搭，字牌可組成 1 組，可推算上聽數為 2 上聽。

圖 2-8 結合各種類的牌可得上聽數為 2。

2.3.3 不同種類牌組合與眼牌的關係

我們在實作 AI 時，實際發現一些較複雜的牌型會無法正確計算出最小上聽數，如圖 2-9(a)若我們把眼定在三萬會得到 1 上聽數，但將相同的牌另外分割，如圖 2.9(b)把眼定在西風牌，可得更小的 0 上聽，因此，不同的牌當眼會得到不同的上聽數，我們針對此問題設計出一個解決方法。

圖 2.9(a) 眼定在三萬為 1 上聽。

圖 2.9(b) 眼定在西風為 0 上聽。

我們會獨立計算各種類牌的組數、搭數和有效牌，除此之外，再檢查此種類的牌是否有兩張以上相同的牌可當眼，若是有多張可當眼的牌，比較哪張當眼有最多的組數、搭數，並將此牌設定為此類牌若包含眼的眼牌。

當結合各種類的牌時，我們會分別計算眼落在不同種類的牌，可得到的上聽數和有效牌分數，如圖 2-10，從中選擇最小的上聽數視為最好的組合。

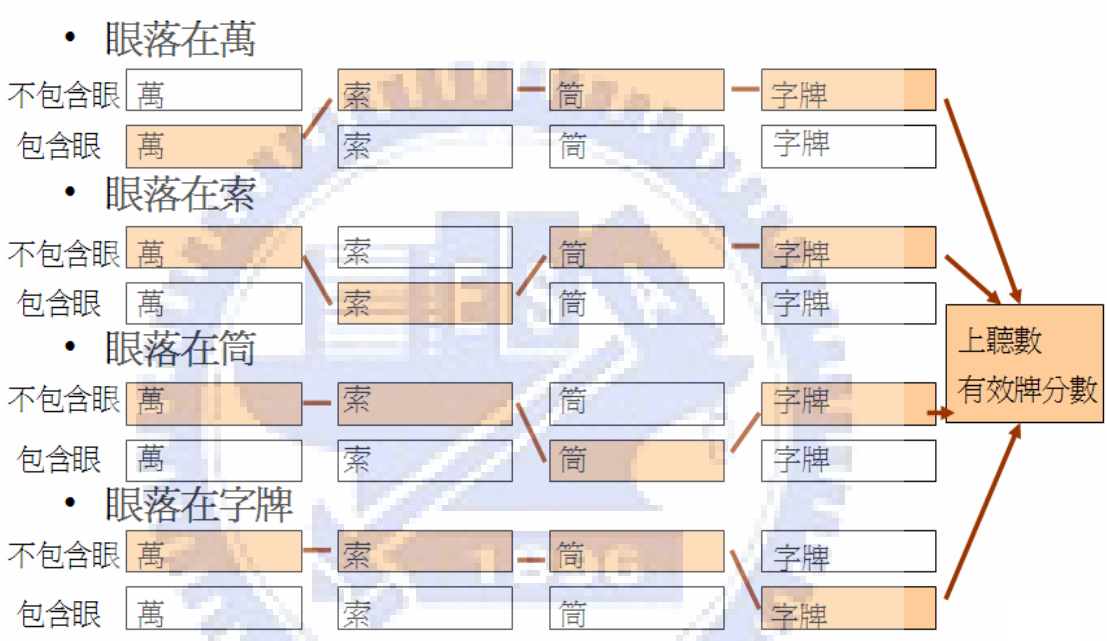

圖 2-10 計算眼落在不同花色的上聽數和有效牌分數。

若擁有相同的上聽數時，代表兩種當眼方式距離聽牌有相同的距離，而這時有效牌的分數可累加，如圖 2-11，若由東風當眼，我方的手牌已經為聽牌狀況，可聽三、六萬；另外，這時也得考慮由六萬當眼，還可以聽東風，因此，這三張都可達成胡牌條件，必須設為聯集關係。

如圖 2-11 不同眼有相同上聽數，有效牌為聯集關係。

2.4 加速搜尋方式

本節首先在 2.4.1 節描述我們想要加速搜尋的原因，並使用預先建表的方式，快速取得萬、索、筒牌的資訊。在 2.4.2 節說明在決定捨牌時，我們可根據不同種類的牌分等級去決策。

2.4.1 預先建立牌數對應表

在上節我們介紹過計算上聽數時，我們必須同時擁有「此種類牌包含眼牌的資訊」及「此種類牌不包含眼牌的資訊」，經由多種組合方式，才能取得實際的上聽數。

我們在遊戲後期，找不到安全牌的狀況下，選擇 Monte-Carlo 來模擬遊戲的進行，而四家玩家的捨牌策略仍是依照原來的 AI，因此，若能增快捨牌的決定速度，就能模擬出更多種的牌局狀況，讓模擬的結果更加準確。

在搜尋萬、索、筒類的有效牌資訊時，我們是使用遞迴的方式去搜尋，然而，實際上我們能把這些搜尋動作，預先計算出所有狀況儲存起來，隨著不同的手牌，經由查表的方式取得有效牌資訊。

麻將每張牌都有四張，因此，我們同時最多取得 4 張，可將擁有的牌數分為 0、1、2、3 及 4 張牌五種狀況，而萬、索、筒類都是由數字 1~數字 9 構成，總共有 5^9 = 1,953,125 個，且每份資料為

11bytes，因此，建出的表約為 23MB。

由於所見的表並不大，因此，我們可在程式啟動時，就將這份表讀到記憶體中，之後，每一次需要取得有效牌資訊時，只需將手牌中該類牌轉成對應的 Index，即可快速取得結果，相較於原本每次使用遞迴搜尋，預先建表可大幅提昇搜尋速度，我們會在第四章節列出實驗數據。

如圖 2-12，我們先計算手牌中的萬字牌，包含一萬 *2、二萬、三萬及四萬，這時我們將手牌轉成對應的 Index，即可取得有效牌資訊，同理而言，索、筒牌也可以使用相同的方式取得。

字牌只可能組成刻子，因此，不需透過查表，根據手牌狀況直接計算即可取得有效牌資訊，最後，我們將四種種類的牌做組合，計算出上聽數和有效牌分數。

圖 2-12 將手牌該類牌轉換成對應的 Index，即可快速取得有效牌資訊。

2.4.2 捨牌分階層搜尋

決定捨牌是麻將遊戲很重要的一環，不論是摸牌、吃牌、碰牌或槓牌，最終都需要選擇一張牌來捨牌，適宜地捨牌不但能快速的減少上聽數，尤其到了遊戲中後期，如何能避開放槍更是重要的決策。

不考慮安全牌的狀況下，我們想知道哪張捨牌可以得到最小的上聽數，我們只需每次丟出一張手牌，然後把剩餘的手牌結果計算並紀

錄，這樣就可以確切的知道丟哪張牌最有利；然而，實際上我們並不需要測試所有的牌，對於已經完成組的三張牌，拆這三張牌一定會增加上聽數，因此，我們可以跳過檢查這些已完成組的牌。

參考實際玩家打牌的經驗，依據牌的有效性分層級，層級越低的越優先捨棄。孤張字牌只能與本身相同的牌才能湊成一組，因此，未來湊成完整一組的機會較小，屬於第一層級；接著，孤張萬、索、筒若在之後拿到鄰近的牌可湊成搭子，相較孤張字牌有較大的機會湊成一組，屬於第二層級；最後，在找不到孤張牌的狀況下，才考慮去拆搭子，拆不同搭會得到不同的有效牌分數，選擇有效牌分數最高的方式捨牌，因此，我們可以把需要檢查的牌分為下面三類：

1. 孤張字牌
2. 孤張萬、索、筒
3. 搭子

第三章、Monte-Carlo模擬避免放槍之設計

在麻將中後期，往往最怕的就是放槍，因此，選擇安全牌是很重要的步驟，我們會依據手牌的狀況，決定所使用的策略，若是到了中後期是二上聽以上，這時就放棄胡牌，以能夠不放槍打到流局為目標；或是我方仍有胡牌機會，則選擇且戰且走，除了避免放槍，同時以保持或減少上聽數為目標。

在 3.1 節我們先簡介我們如何模擬牌局，產生對方可能的手牌，在 3.2 節說明在模擬手牌時，實際遇上的問題，以及我們的解決方式。在 3.3 節介紹如何將模擬牌局到最後的資訊更新，作為決策時的依據。在 3.4 節描述當決策時間用盡，我們如何選擇捨牌。

3.1 模擬其他玩家手牌

遊戲進行到中後期，我們開始使用 Monte-Carlo 來解放槍問題，然而這時若是完全隨機模擬其他玩家的手牌，產生的手牌可能都是二上聽以上，這時怎麼打都不會放槍，模擬出來的結果也不具參考性，因此，我們會先依據統計的結果，產生對方可能的上聽數，在實際去模擬對方的手牌。

一巡為我方兩次捨牌之間，或四家行牌一周[3]。我們使用四家相同的 AI，實驗 1100 場，統計出第八巡至第十二巡各玩家實際上聽數的分佈，如圖 3-1。

因此，我們實際模擬另外三家手牌的流程如下：

1. 依據不同巡的機率分佈，隨機產生三家的上聽數。
2. 根據玩家目前現有的吃、碰狀況，檢查此上聽數是否合理，決定是否重新產生上聽數。
3. 三家都取得合理的上聽數後，再開始產生可能的手牌。

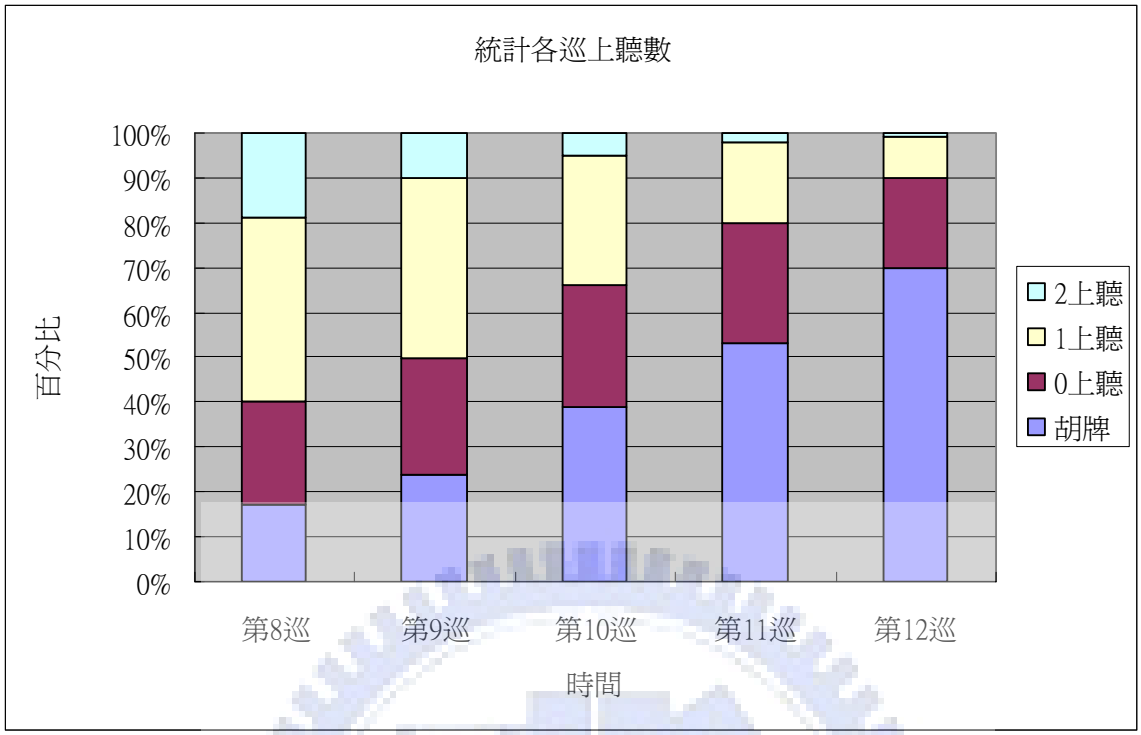

圖 3-1 各巡玩家的上聽數分佈。

3.2 模擬所遭遇的問題

本節首先在 3.2.1 節簡介模擬對方手牌時，由於牌堆所剩牌數不多，常會產生不合理的手牌。接著在 3.2.2 節說明使用建表的方式，來解決上述的問題。

模擬玩家手牌所遭遇的問題

同上聽數的狀況下，仍有許多種牌型，以 0 上聽為例，我們將其分為三類：

1. 五組完整的三張牌組，再加上一張孤張牌。
2. 四組完整的牌組，加上兩組對子。
3. 四組完整的牌組及一組眼，再加上一組搭子(中洞搭或連續搭)。

我們隨機決定為何種牌型，並開始產生玩家的手牌，首先，模擬

對方已完成組的部份。

直觀而言，依照萬、索、筒的特性，可產生完整組的方式有順或刻，單一種牌類可產生七種順和九種刻，再加上字牌有七種刻，因此，所有的種類為

(7 + 9) * 3 + 7 = 55

我們可隨機產生 1~55 的數字，再依照定義的對照關係，取得實際所對應的三張牌，並根據剩餘未知的牌檢查是否合理，若合理則繼續產生之後的牌。

然而，實際設計時，由於剩餘未知的牌數並不多，且每場牌局分佈的狀況不同，很常會產生不合理的牌組，舉例來說，遊戲進行到第十巡時，假設四家共吃、碰三組，這時剩餘未知的牌數為

136 - (4 * 10 + 3*3 + 16) = 71

我們可紀錄所有捨牌的資訊，得到剩餘未知牌的數目，以表 3-1 為例，表中紀錄萬、索、筒及字牌剩餘的張數，我們發現隨機產生的 55 種完整組，有高達 23 種不存在，換言之，有高達 23/55 = 42%機率會產生不合理的完整牌組，且會隨著每產生一個新合理牌組而減少未知牌數，更讓產生不合理組的機率逐漸升高。

萬	1	2	3	4	5	6	7	8	9
張數	3	0	2	3	0	3	3	0	3

索	1	2	3	4	5	6	7	8	9
張數	0	2	3	0	4	3	3	2	3

筒	1	2	3	4	5	6	7	8	9
張數	3	2	0	3	2	0	3	3	4

字牌	東	南	西	北	中	發	白		
張數	3	2	2	3	2	4	3		

表 3-1 假設為十巡後，扣除已知牌，剩餘 71 張未知牌分佈的例子。

改良模擬方式

為了防止在產生牌組時，常產生一些不合理的牌組，因此，我們改變模擬方式，預先建表紀錄不同剩餘牌的狀況下，可產生合理牌組的組數及是何種順或刻。

因此，實際要產生完整組時，我們再根據各花色牌的分佈獨自去讀表（字牌使用另外的計算方式），並加總四種類可能產生組的總數 Max，再隨機產生 1~Max 的數字，將此數字去取得對應的組，這樣的產生方式可保證每次產生的組皆為合理的牌組。

表 3-2 為與 3-1 相同的例子，右方的數字為我們紀錄完整組的數目，四種種類可產生的合理組共計 22 組，因此，我們只需隨機產生 1~22 的數字，並取得對應的組，可保證每次取出的皆為合理的組。

萬	1	2	3	4	5	6	7	8	9
張數	3	0	2	3	0	3	3	0	3

順	刻
0	5

索	1	2	3	4	5	6	7	8	9
張數	0	2	3	0	4	3	3	2	3

順	刻
3	5

筒	1	2	3	4	5	6	7	8	9
張數	3	2	0	3	2	0	3	3	4

順	刻
1	4

字牌	東	南	西	北	中	發	白		
張數	3	2	2	3	2	4	3		

順	刻
0	4

表 3-2 將剩餘未知的牌數查表取得合理的組數，之後產生的皆為合理的完整組。

3.3 分數回傳更新

我們模擬出一個合理的牌局後（含三家玩家的手牌和之後牌堆摸牌的順序），接著輪流丟出我方需要檢驗的手牌，模擬遊戲進行到最後的結果，再把最後胡牌或流局的資訊更新各節點。

如圖 3-2，我們會去檢查未成搭的牌，包含一萬、二索、三索、六索、七索、二筒及三筒，模擬遊戲進行到最後，並更新對應節點的資訊。

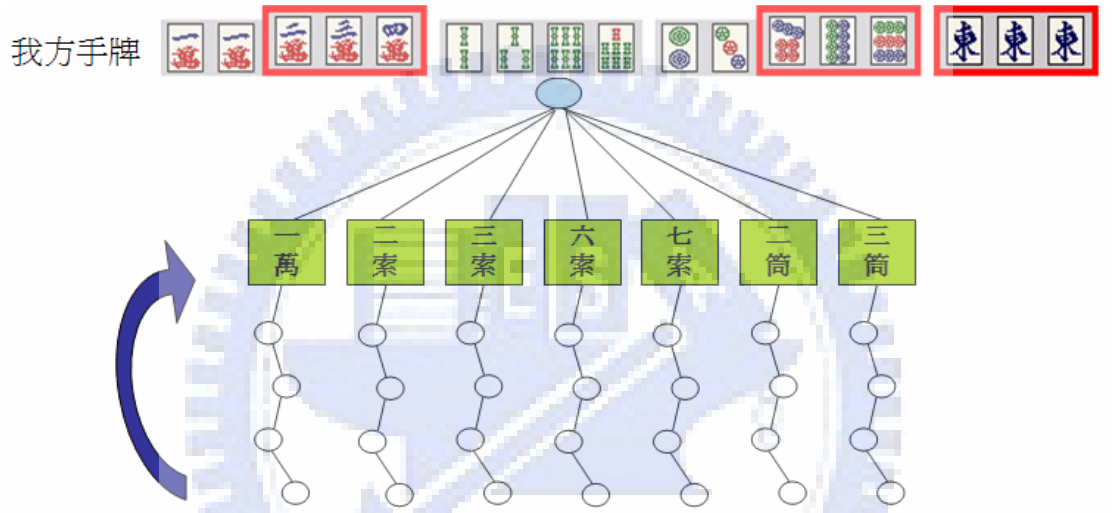

圖 3-2 模擬遊戲進行到最後，並更新胡牌或流局的資訊。

3.4 決定捨牌

本節首先在 3.4.1 節簡介我們的計分方式，除了統計獲勝的得分積分，同時也考慮放槍的扣分。接著在 3.4.2 節說明如何決定最後的捨牌。

3.4.1 設定積分方式

若玩家 A 使用自摸的方式達成胡牌，這時，玩家 A 的積分為+3 分，其他玩家都－1 分；若是玩家 A 放槍給玩家 B，玩家 A 的積分為-3 分，玩家 B+3 分，我們使用這種計分方式累加來得到最後的積分。

3.4.2 決定捨牌

我們設定為一秒內要做出捨牌的決策，因此，在 Monte-Carlo 啟動後，我們開始模擬牌局，並不斷地更新各節點的積分，待決策時間用盡，便依據目前積分最高的設為捨牌。

如圖 3-3，我們選擇最左邊積分最高的為捨牌，此為使用 Monte-Carlo 模擬出最有可能避免放槍的牌。

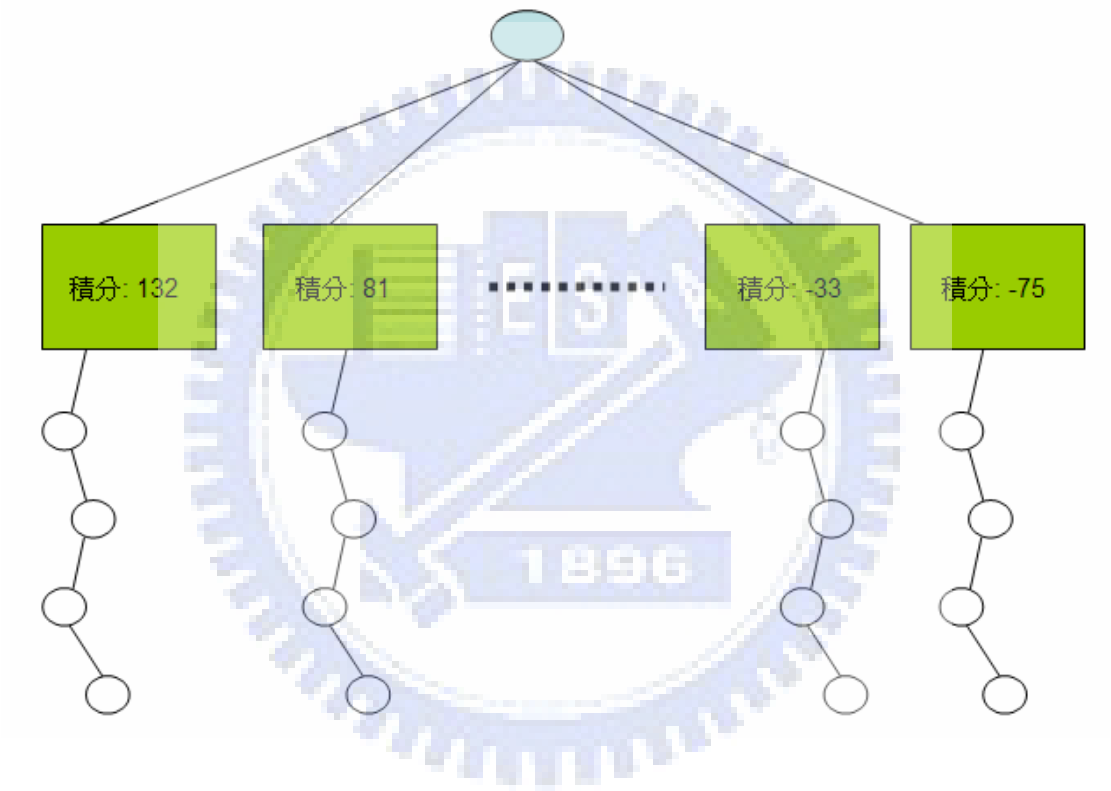

圖 3-3 選擇積分最高的牌來捨棄。

第四章、實驗數據分析與討論

本章將於 4.1 節說明本論文中使用的實驗環境，並在 4.2 節分析數據與討論。

4.1 實驗環境

感謝群想公司提供的 CYC 系統平台，這是由實驗室學長開發設計的，讓我們可以直接應用麻將子系統。CYC 系統中的麻將子系統，提供了完整的 GUI 介面，有關網路傳輸指令的部份都已設計好，我們只需依照預先定義好的指令，專注於設計 AI 即可。

由於 CYC 系統是用 Java 所撰寫出來的，當程式執行時，會透過 Java Applet 的方式，讓本機的 AI 與 Server 連上線並開始比賽，我們執行 AI 所使用的電腦配備如表 4-1。

CPU	AMD Athlon™ 64 Processor (1.8GHz)
RAM	1.5 GB DDR400
Platform	Windos XP SP2 Java Platform,Standard Edition 6

表 4-1 AI 程式執行環境。

4.2 實驗數據分析

本節在 4.2.1 節描述使用不同方式分析手牌，計算速度上的改良。接著在 4.2.2 節說明不同版本的 AI，勝率上的差異。

4.2.1 分析手牌速度的提昇

我們分了四種版本做比較，結果如圖 4-1。由於決定捨牌是最花費時間的計算，因此，各版本各測試 100 場比賽，約有 2000 個捨牌指令，各版本詳述如下：

1. 遞迴搜尋 – 每一巡決定捨牌即時遞迴搜尋計算。
2. 查表_全部搜尋 – 預先將萬、索、筒單一花色建表，將手牌中的每張牌輪流捨出，一次捨出一張，再計算上聽數和有效分數來比較，選擇可得到最佳結果的定為最後捨牌。
3. 查表_二層搜尋 – 同樣使用預先建表的方式，將捨牌分為兩種層級，第一層級為孤張字牌，優先檢查孤張字牌，比較不同捨牌後的分數來決定出牌。若是沒有孤張字牌，才去選擇未成組的牌做捨牌比較。
4. 查表_三層搜尋 – 同樣使用預先建表的方式，並將查表 Ver2 再多加一層分類，第一層為孤張字牌，第二層為孤張非字牌，第三層為未成組的搭子，使用分層方式對於決定捨牌，速度有大幅的提昇。

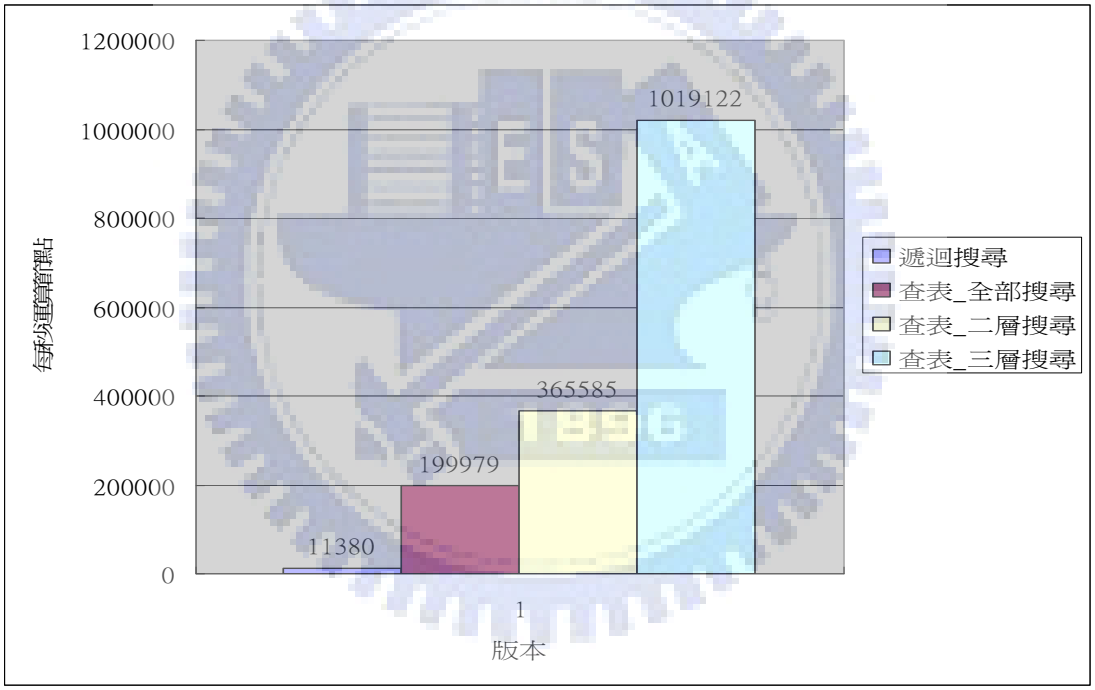

圖 4-1 不同版本的計算速度比較。

由圖中可知，使用建表的方式，讓原本每秒只能運算 1 萬個 Nodes 的速度，提升到每秒能運算近 20 萬 nodes，提昇的幅度非常明顯。另外，針對不同的牌局，有些牌具有被捨出的優先權，若能適當的分層，對於運算提昇的速度也是很顯著，決定分層的方式是非常重要的一環，除了要考慮到提昇的效能，同時也還需注意合理性。我們實際觀察比賽的進行，目前的分類方式對於大部分的手牌都能做出合理的

捨牌，但仍有些特例需要另外處理。

4.2.2 不同版本勝率的變化

我們計算的積分方式為，若玩家 A 放槍讓玩家 B 胡牌，則玩家 A -3 分，玩家 B +3 分，若是玩家 A 自摸胡牌，則自己+3 分，其他玩家 -1 分，使用這樣的計分方式，能夠加大放槍對總積分的影響。

另外，比較的四個版本詳述如下：

1. 基本搜尋 - 早期的版本，程式進行中以遞迴方式搜尋，不包含眼牌比較和聯集概念。
2. 搜尋+眼牌 - 萬、索、筒的分數由預先建的表取出，再加上字牌，比較不同眼牌的影響，選擇出可得到最佳結果的定為最後捨牌。
3. MC_10 - 程式進行到第十巡，找不到絕對安全牌時，啟動 Monte-Carlo。
4. MC_終局 - 從第八巡開始啟動檢查機制，若上聽數低(目前設定為 3 上聽)，則保持原本的丟牌決策方式，反之，才啟用 Monte-Carlo 決定捨牌。

共分三方向去比較，分別為積分、勝場數及放槍數。圖 4-2 為勝場數比較，勝場數越高越好，搜尋+眼牌相較於基本搜尋，在加入了眼牌比較及有效牌的聯集概念後，勝率明顯的提昇了，比較特別的是，MC_10 的勝場是下降，MC_終局有最高的勝場數。圖 4-3 為放槍數比較，放槍數越低越好，值得注意的是，MC_10 並沒有明顯的減少上聽數，然而，MC_終局則有顯著的改善。圖 4-4 為積分比較，我們得到結論為在終局啟用 Monte-Carlo 會有最佳的結果。

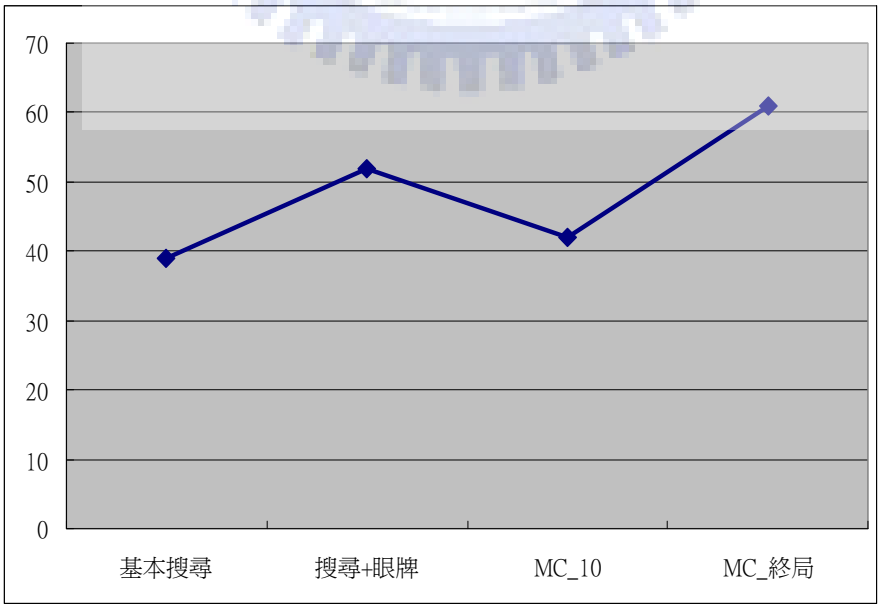

圖 4-2 勝場數比

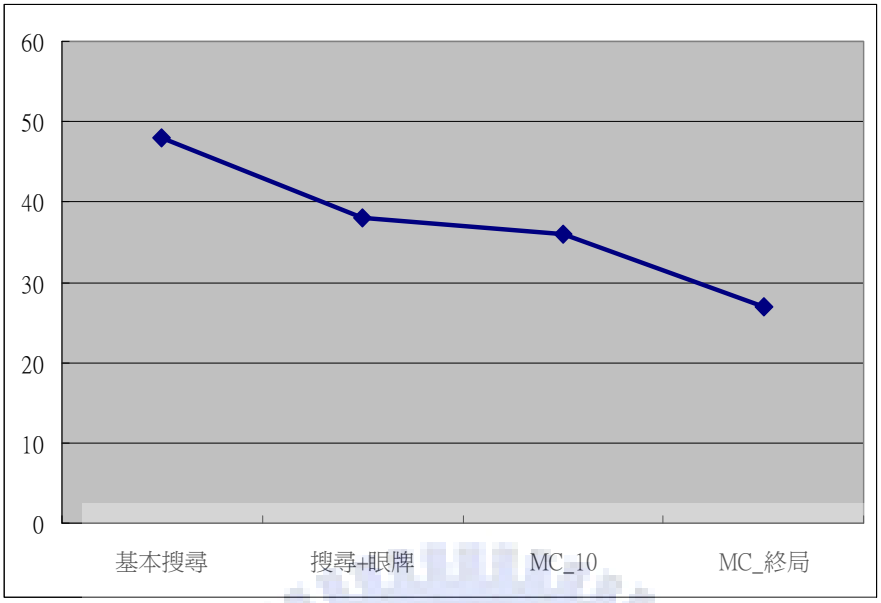

圖 4-3 放槍數比較

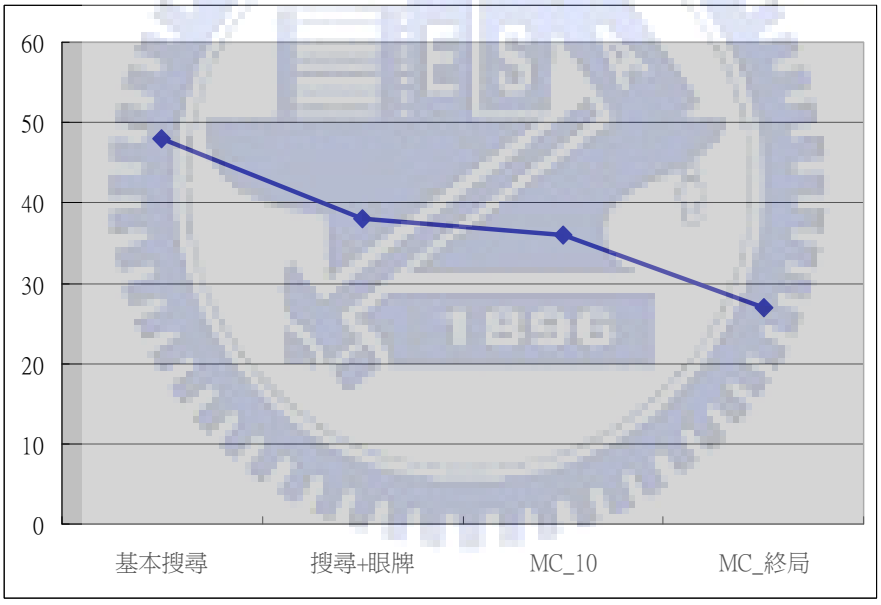

圖 4-4 積分比較

我們去分析實驗結果，發現 MC_10 的結果並不理想，放槍數沒有明顯降低，勝場數還下降，討論其原因是因為在遊戲中局就去模擬，很難正確的模擬出對方的手牌，因此，使用 Monte-Carlo 反而讓捨牌決策造成偏差；相較於 MC_終局，若是在終局才啟動 Monte-Carlo，由於所剩的牌更少，模擬的結果較符合實際狀況，因此，對於減少放槍數有明顯的改善。

第五章、結論與未來展望

本論文是根據麻將的遊戲特性，希望能設計出聰明的 AI。麻將相關的論文研究較少，因此，在分析與實作時，除了根據真實中玩家打牌的方式來構思，實際設計時，發現一些遊戲的特性，必須要特別注意，我們在論文中整理出一些遭遇到的問題及解決方式。

我們使用預先建表的方式，考慮萬、索、筒這類牌可能產生的所有組合，這對於計算每副手牌上聽數，具有很顯著的加速效果，越能快速的計算出各種手牌的資訊，使用 Monte-Carlo 來模擬的場次就能越多，預期也能有更準確的效果。

我們使用 Monte-Carlo 來模擬牌局，並加上一般玩家打牌的概念，到了中後期，若是發現離胡牌還很遠，就以追求安全下車為唯一目標，若是還有機會胡牌則且戰且走，希望能同時兼顧不放槍並胡牌。

使用此模擬方法，所得到的效果顯著，終局使用 Monte-Carlo 確實能降低放槍數，勝率也有跟著提高，與我們的預期的效果相同，但特別值得注意，若中局就啟動 Monte-Carlo，我們發現放槍數並沒有變好的偏向，甚至連勝場數也跟著下降，我們討論其原因，認為由於中局所剩牌過多，較難正確預估對方手牌，反而會讓捨牌決策偏差，造成反效果。

麻將是一個策略多變的遊戲，因此，若是能夠加入更多麻將的專家知識，更貼近現實中高手玩家的打牌方式，麻將 AI 相信會有更好的表現；另外，我們在未來也希望能嘗試不同方式去模擬對手手牌，比較不同的 Monte-Carlo 模擬手牌方式，以期能增加勝場數、減少放槍數，讓麻將 AI 能有更聰明的表現。

參考文獻

[1] 張晉愼,麻將大富豪,婦女與生活社文化事業有限公司,Feb 2005

[2] 麻將尊者,台灣麻將高手,2006,知青頻道出版

[3] 中華麻將競技協會,available from http://atawmj.org.tw/

[4] Bouzy, B. (2005). Associating Domain-Dependent Knowledge and Monte-Carlo Approaches within a Go program, Information Sciences, Heuristic Search and Computer Game Playing IV 175, 4, pp. 247–257.

[5] Bouzy, B. and Helmstetter, B. (2003). Monte-Carlo Go Developments, in H. J.van den Herik, H. Iida and E. A. Heinz (eds.), Proceedings of the 10th Advances in Computer Games Conference (ACG-10) (Kluwer Academic), pp. 159–174.

[6] E.R. Berlekamp, Program for double-dummy bridge problems – a new strategy for mechanical game playing,Journal of the ACM,10(4):357-364,1963

[7] Jay Burmeister and Janet Wiles,An introduction to the computer Go field and associated Internet resources,Technical Report CS-TR-339,Department of Computer Science,University of Queensland,1995

[8] Tristan Cazenave, A Phantom Go Program, Universit'e Paris 8, St-Denis, France,2007

[9] Ian Frank,1997,Computer Bridge Survey

[10] Ian Frank,David Basin,1998,Search in games with incomplete information: a case study using Bridge card play,Artificial Intelligence 100:87-123

[11] Fudenberg, D. , Tirole, J. ,Game Theory, MIT Press, Chapter 3, sect 2.2 , 1993

[12] Gibbons, R. ,A primer in game theory, Harvester-Wheatsheaf, Chapter 2, 1992

[13] M. Ginsberg, How computers will play bridge,The Bridge World,1995

[14] M. Ginsberg,Steps Toward an Expert-Level Bridge-Playing Program, IJCAL,1999,available from http://www.cirl.uoregon.edu/research/gib.html

[15] Sylvain Gelly,Yzzao Wang,Remi Munos,and Olivier Teytaud.Modification of UCT with patterns in Monte-Carlo Go.Technical Report 6062,INRIA,France,November 2006

[16] Sylvain Gelly and Yizao Wang.Exploration exploitation in Go:UCT for Monte-Carlo Go,December 2006

[17] Sylvain Gelly and David Silver.Combining online and offline knowledge in UCT.In international Conference on Machine Learning,ICML 2007,2007.

[18] Daniel Hellsson,A Mah Jong-Playing Program, 2000

[19] Luce, R.D. ,Raiffa, H., Games and Decisions: Introduction and Critical Survey, Wiley & Sons ,Chapter 3, section 2 , 1957

[20] Stephen J. J. Smith,Dana Nau,Tom Throop,1998,Computer Bridge: A Big Win for AI Planning,AI Magazine, 19(2):93-105

[21] 莊凱閔,陳玥汝,林順喜,2007,電腦麻將演算及相關議題之研究,TAAI2007第十二屆人工智慧與應用研討會