亚马逊棋博弈技术研究

乔 治 黄 鸿
北京理工大学 100081
qiaozhi@bit.edu.cn

摘要：亚马逊棋作为一种博弈类游戏，由于其本身特殊的规则，使其适合于博弈中人工智能算法的研究。文章从两个方面讨论亚马逊棋的博弈技术，先是提出了一种对亚马逊棋局面进行评估的方法，为亚马逊棋的初学者在局面评估方面提供了一定的思路。然后介绍 MC/UCT 搜索算法在亚马逊棋搜索中的运用，UCT 是机器博弈领域最新的搜索算法，它很好的解决了传统搜索算法无法在亚马逊棋搜索中遍历更深层节点的问题。

关键词：亚马逊棋 机器博弈 局面评估 UCT 算法

1. 引言

亚马逊棋是欧美国家的一个棋种，在亚洲并不常见，它由国际象棋中后的走法衍生而来，与著名的 8 皇后问题貌似神和。

亚马逊棋的棋盘由 10*10 的格子组成，每方拥有 4 个皇后，它们的走法与国际象棋中后的走法相同而且每行一步还要放置一个障碍，这种特殊的走法规则导致了亚马逊棋每步的走法非常多，一般在 1500 左右，而且对大多数走法的评估也具有很大的不确定性。由于这几种特点，导致了亚马逊棋并不适合人与人之间对弈，而确非常合适用于人工智能，机器博弈方面的研究。

2. 亚马逊棋基本评估方法

在刚接触亚马逊棋的时候，对于这种棋的评估会让许多人觉得无所适从，什么的走法在这样的棋盘上才算一步好棋呢，亚马逊棋最终的目的是用自身和障碍将对方的棋子堵死，使其不能移动，而对于可以在八个方向任意行走的 Queen 来说，将其堵死又谈何容易，所以还有一种占地盘的思想，就是利用自身和障碍为自己圈得足够大的一块地盘，当然这块地盘别人是不能够进来的，这样当自身获得足够的地盘时，对方会因为没有地方可走而自己将自己堵死。现在的主流方法是第二种，即为自己圈出足够大的地盘。

然而即使有了大体的思路，在棋盘中走出具体的一步还是显得十分困难，特别是在对局初期，当棋盘比较空旷时，通过人的观察和判断很难确定一步棋的好坏。这时的亚马逊棋和围棋很相近，初期的许多走法更像是布局而非具体的争夺。而到了对局的中后期，当棋盘上的障碍逐渐增多，每块地盘的形状已经明朗时，亚马逊棋的行棋思路则又和象棋相近，需要通过更深层的搜索来确定具体某块区域的争夺和归属情况。

Jens Lieberum 在他自己的关于亚马逊棋评估的文章中[1]，对亚马逊棋的评估方法做了详细的讨论，此篇文章也是现在亚马逊棋评估方法的主要来源，本文在这一小节评估方法方面的讨论主要是基于这篇文章的。

（1）基于 QueenMove 和 KingMove 的评估

QueenMove 和 KingMove 思想：

如图 2-1 所示：

QueenMove 表示对于某一个空格（即没有障碍和棋子）。某方的四个棋子通过 Queen 的走法走到此空格最少需要走多少步；

KingMove 表示对于某一个空格（即没有障碍和棋子）。某方的四个棋子通过 King 的走法走到此空格最少需要走多少步。

如：第一副图左上方的空格数为 2 和 4，表示白方通过 Queen 走法最少需要 2 步可以到达此处，而黑方则需要 4 步，同样对于第二副图中的 2 和 7 表示 King 走法时的情况。

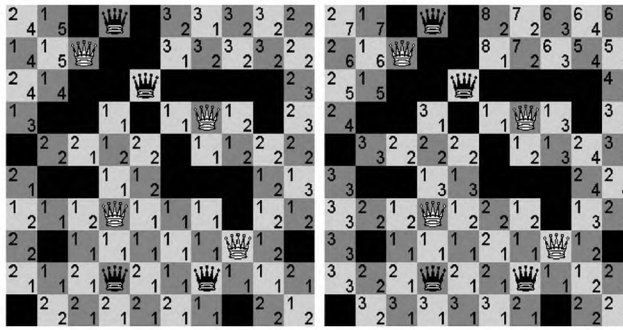

图 2–1　QueenMove 和 KingMove 分布图

如上文所述，通过对每一空格 QueenMove 和 KingMove 的计算，可以大体看出棋局的优劣，当白方棋子可以比黑方棋子用更少的步数到达某一空格时，我们认为白方实际已经控制了这个空格（地盘），这样通过对全局每一个空格的计算，可以算出每方控制的空格数目，从而大体确定局势的优劣。

下面给出计算公式：

$$t_i = \sum_a \Delta(D_i^1(a), D_i^2(a)) \quad (i = 1 or 2) \qquad \text{公式 2–1}$$

其中

$\Delta(n,m) = 0$ 当 $n = m = \infty$；

$\Delta(n,m) = k$ 当 $n = m < \infty$； $-1 < k < 1$

$\Delta(n,m) = 1$ 当 $n < m$；

$\Delta(n,m) = -1$ 当 $n > m$；

对公式的解释：

公式 $D_x^y(a)$ 中，a 表示棋盘中任意空格（没有棋子或障碍的格子），下标 x 表示 QueenMove 还是 KingMove 的选择，上标 y 表示白方和黑方的选择。如：$D_i^2(a)$ 表示对于黑方到达某个标号为 a 的空格采用 QueenMove 的走法最少需要走几步。计算结果 t_i 则表示采用 QueenMove 或 KingMove 时，白方和黑方对所有空格占有情况的综合评估得分。

（2）棋子灵活度的计算

当棋盘上障碍较多时，即处于对局中后期时，对于棋盘的 QueenMove 和 KingMove 的评估是一种相对准确的评估方法，而当棋盘上的障碍较少，棋局处于前期布局阶段时这种评估方式就显得并不合适，因为这个时候双方对大多数空格的控制都在两步以内，而此时一个棋子可能控制很大一片区域（实际上一个棋子是没有机会得到这么大一片地方的）。

由于上述情况的存在，所以还需在开局阶段对每个棋子的灵活度做出评估，使得某个棋子不会在一开始就被对方堵死而其它的棋子却控制了过大的地盘。

棋子灵活度的计算如图 2–2 所示

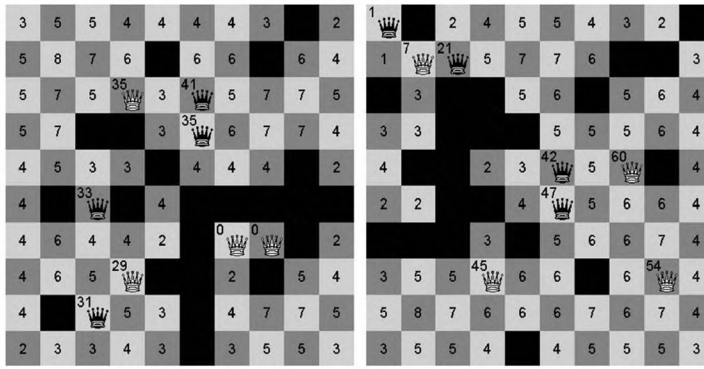

图 2–2　灵活度分布图

图中棋子左上角的数字即为该棋子的在此局面下的灵活度数值。

下面是某个灵活度的计算方法：

1. 计算出当前棋盘中每个空格的相邻空格个数（八个方向），作为该空格的灵魂度；

2. 对于某个棋子记录下此棋子采用 QueenMove 走法时一步之内能到达的空格；

3. 对于这些空格采用 KingMove 走法到达时步数作为分母，将所有空格的灵活度相加，即得到当前棋子的灵活度。

例如对于图 3–2 中左上角的白色棋子，它的灵活度计算式为：

$$a = 7+6+5+3+3+(5+4+7+4)/2+5/4 = 35.25$$

此时我们就可以对棋盘的灵活度做出评估：

$$m = \sum_{a} F(a) - \sum_{b} F(b) \qquad \text{公式 2–2}$$

其中 a 表示某方棋子，$F(a)$ 表示此棋子的灵活度计算式。

（3）综合计算

上述在讨论亚马逊棋局面评估方法时，多次提及对局的进度问题，很容易看出，当对局处于不同的进度时，这几种评估所占的份量是不同的。

$$value = F(w,t_1) + F(w,t_2) + F(w,m) \qquad \text{公式 2–3}$$

公式 2–3 给出了一种综合评估的方法，式中 t_1，t_2 和 m 分别表示棋盘的 QueenMove，KingMove 和灵活度评估值，w 表示当前对局的进度，可见综合的评估值是上面 3 个值与对局进度的函数。

关于对局进度的计算，有许多不同的方法，大体可以从已走的招法数和当前双方棋子灵活度数值综合考虑得出，Jens Lieberum 在他的文章中也给出他的计算方法，读者可以参考。

对于 F 函数的形式问题，则属于参数优化的范畴。

3. MC 搜索算法

在文章的开头已经提到，由于亚马逊棋特殊的规则，导致了亚马逊棋平均每步的合法走法多达 1000 多种（第一步有 2176 种），如果仅向下搜索一层，则所需要评估的所有节点数已经超过百万，即使加入较好的剪枝算法，亚马逊棋庞大的节点数仍然让传统的基于 minmax

的搜索方法望而却步，据统计一般基于 minmax 的搜索算法在开局仅能搜索到 3–4 层，在中后期可以到 5 层左右。

MC（蒙特卡洛）搜索算法最早是在围棋博弈领域提出，根据大数定理，MC 对求解问题本身就具有概率和统计性的情况，例如中子在介质中的传播，核衰变过程等，我们可以使用直接蒙特卡洛模拟方法。该方法是按照实际问题所遵循的概率统计规律，用电子计算机进行直接的抽样试验，然后计算其统计参数。直接蒙特卡洛模拟法最充分体现出蒙特卡洛方法无可比拟的特殊性和优越性，因而在物理学的各种各样问题中得到广泛的应用。该方法也就是通常所说的“计算机实验”。

对于机器博弈中的搜索，假设下了一步想走在这里，那么如何知道这步棋好不好呢？有个科学家就提出了，假设你这步棋已经下到这了，找两个傻子，让他们随便下，把棋下完，看看是黑棋胜还是白棋胜。只有两个傻子，这结局太有偶然性了，那么我们找 10000 对傻子来下一万盘棋，然后统计一下是黑棋赢的多还是白棋赢的多，这样就得到了这一手棋的获胜概率，然后把所有可能下法的获胜概率都这样算出来，选择获胜概率最大的下法。这就是 MC 思想在机器博弈搜索算法中的运用。

4. 从 MC 到 UCT 算法

当人们把 MC 算法直接用在计算机围棋等领域时，发现这种算法存在很多缺点，首先是浪费了模拟次数，因为在有限的时间内只能模拟出一定的棋局数。其次，它没有考虑对方的下一手，如果我们把 MC 当做估值函数，那么对比传统的人机博弈程序，简单的 MC 相当于只搜索了一步。

博弈搜索可以理解成一个树形结构上搜索，在树形搜索上结合 UCB，就是 UCT 算法了（UCB for Tree）。

假设已经有了一棵树，每一个叶子节点上有一个评估值，怎么用 UCB 算法来计算这些评估值呢？方法一是把所有的叶子节点都列出来，然后按公式计算每一个叶子节点的 UCB 值，从中取最大值的那个叶子节点。这个方法有点笨，既然这样，何必要用树呢？

方法二，由每一个非叶节点汇总它下面的节点的评估值和模拟过的次数，如果下面节点也不是叶节点，那么也向下汇总。这样计算 UCB 值时，只需要计算直接子节点中的最大值，然后进入这个子节点，再去选择它的下一级节点的最大值，以此类推，最终选到的那个叶节点。

方法二与方法一选出来的最大值不是同一个，因为，方法二多出一个假定，要选的最大值是在最大分支里面。而实际上一个最有钱的城市里不一定住着最有钱的人，因此，这个假定不一定合理，不过无所谓了，就算找出来的不是最有钱的，也不会太平庸。

把 UCT 用于棋类博弈，我们会面临两个问题：一，UCT 的树是已经建立好的，而对于围棋，初始情况下是没有树的，树如何建立。二，棋是黑白双方轮流下子，这个又当如何考虑呢。

首先，如果假设事先建立起了一棵庞大无比的博弈树，它的节点已经包含了全部的 n 层对局，就是任意下 n 手，得到的局面都已经出现在了这棵树的节点上。这个时候，就可以利用 UCT 进行搜索了，不错，但是这个时候最多也只能搜索 n 层。而且，大部分节点根本来不及去使用。

因此，一开始建立根节点，再为根节点建立它的第一层子节点，除此之外不再预先建立子节点，而仅在需要的时候再建造。这个时候开始寻找这棵树的最大 UCB 值的叶节点了，就用上面的 UCT 算法，遇到叶节点后，就用 MC 方法开始模拟对局一次，得出了一个胜负结果，然后更新它的祖先节点，直到此时，算法的进行与 UCT 一模一样，现在又开始下一轮寻找最大 UCB 了，假设找到的还是上一轮的那个节点，为它生成它的下一层子节点，然后继续用

UCT 的方法寻找最大的 UCB 值，然后继续。

现在的这种方法，其实每一个节点所代表的局面，最多只进行一次模拟对局，因为下次再选中它时，就会继续深入到它的子节点了。这再一次印证了 UCT 的思想，不要最强的节点，只要最强的分支。因此单个节点由于运气表现不好时，只要整个分支表现好，还是有机会的。

对于双方轮流下子的问题，很简单，每一层子节点所属的是黑子还是白子这是可以知道的，所以在更新胜负结果数时，只更新对应颜色的胜局数。按照这样的方式汇总胜负数，在寻找 UCB 最大值时，就刚好是符合双方都走最强应对的原则。因为每方只记录自己这方的胜利数，选择的时候，都是选使得自己这方胜率最大的节点。

UCT 方法是 MC 方法的发展，在 MC 方法中，我们试图在某个节点通过一定次数的模拟对局反映出这个节点最终的性质。但是由于我们能够模拟对局数是有限的，一般在几万次左右，而整体对局数十分庞大（远远大于模拟数），所以模拟的结果往往不能真实反映这个节点的性质（偏差较大）。UCT 方法的实质是运用了这样一种思想，就是尽量在有价值的节点上作较多的模拟对局，什么是有价值的节点，就是敌我对局时大家都愿意选择的节点，这些节点的层层确定是通过 minmax 方法，而这些节点的得分的获得则是通过 MC 方法，所以 UCT 方法实际上是一种 minmax 和 MC 的混合，使用 MC 模拟对局，通过 minmax 确定路径（下一次对局选用的节点），从而使有限次模拟对局的结果更准确地反映出节点的性质。

5. UCT 搜索算法在亚马逊棋中的运用

UCT 算法源于围棋，但同样使用于亚马逊棋，只是在将围棋中的算法复制到亚马逊棋中时，会有一定的变化。

基本步骤：

由当前局面建立根节点，生成根节点的子节点

1. 从根节点开始
2. 利用 UCB 公式计算每个子节点的 UCB 值，选择最大值的子节点
3. 若此节点不是叶节点，则从此节点开始，重复 2
4. 直到遇到叶节点，如果叶节点曾经被模拟对局过 40 次，为这个叶节点生成子节点，从此节点开始，重复 2
5. 否则对这个叶节点进行模拟对局，得到胜负结果，将这个收益按对应颜色更新到该节点及它的每一级祖先节点上去
6. 回到 1，除非时间结束或者达到预设循环次数
7. 从根节点的子节点中挑选平均收益最高的，作为最佳点

UCB 公式：

$$UCB = X + k * \sqrt{\frac{\ln(parent\_count)}{node\_count}} \qquad \text{公式 5-1}$$

X：由模拟对局输赢次数决定，表现该节点基本性质

K：常数系数，K 取大表示希望模拟更多的兄弟节点，取小表示希望走入更深的层中；

Parent_count：父节点被模拟对局的次数；

Node_count：该结点被模拟对局的次数。

亚马逊棋中使用 UCT 算法需要额外确定的一些参数和变化：

1. 节点被模拟多少次后决定展开：当一个节点被模拟对局一定次数后，才能初步得到此节点的统计学信息，并展开节点。

2. 提前剪枝：亚马逊棋中每步的着法过于庞大，一般在 1000 步左右，而许多着法可以很容易的判定为垃圾，所以一般进行提前剪枝，保留 400 步左右的着法。

3. 决定展开节点后，子节点需要用提前剪枝后的顺序排列，并且并不生成所有的子节点而是按顺序先生产一定数量的子节点，在父节点的访问量突破一定数量时，再决定下一步生产的子节点。

4. 模拟对局层数的决定：一般选择 6 至 10 层。

5. 底层评估方法：一种为返回输赢，一种为返回输赢程度。

6. 结论

本文先给出了亚马逊几个主要评估方法，也是当今亚马逊棋评估的主流思想，通过对这种方法学习，可以给亚马逊棋的初学者提供一些思路，不再感到无所适从。文章后几段详细论述了现在机器博弈最新的 UCT 算法的原理和其在亚马逊中运用的流程以及需要注意的事项，为传统搜索算法无法在亚马逊棋环境中遍历更深层节点的问题找到了另一条解题的思路。

参 考 文 献

[1] Jens.Lieberum：An evaluation function for the game of amazons Theoretical .Computer Science 349（2005）230 – 244.

[2] J. Conway：On Numbers and Games，Academic Press，NewYork，1976.

[3] Richard J. Lorentz：Amazons Discover Monte Carlo. Department of Computer Science，California State University，Northridge CA 91330–8281，USA.

[4] Julien Kloetzer，Hiroyuki Iida，and Bruno Bouzy：The Monte-Carlo Approach in Amazons. Research Unit for Computers and Games Japan Advanced Institute of Science and Technology.