四子棋作业实验报告

计 52 周京汉 2015011245

2017.05.16

1 实验任务简介

在 M 行 N 列的棋盘中，棋手每次只能在每一列当前的最底部落子，如果某一列已经落满，则不能在该列中落子，目标是在横向、纵向、两个斜向共四个方向中的任意一个方向上，使自己的棋子连成四个（或四个以上），并阻止对方达到同样的企图。先形成四连子的一方获胜，如果直到棋盘落满双方都没能达到目标，则为平局。

棋盘的大小是随机的，宽度和高度的范围均为 [9,12]，每次棋盘生成之后，会同时在棋盘上随机生成一个不可以落子的位置。因此任何情况下都存在必胜策略是不可能的，程序的目标是给出在任何情况下都可行的 AI 算法。

2 实验算法说明

2.1 核心算法

在本实验中，我使用了蒙特卡洛信心上限搜索树（UCT）的算法。其基本功能分为：在树中进行搜索，选择进行结点扩展或者寻找出最好的子节点，扩展树节点策略，选择最佳子节点，计算默认的收益值和回溯更新 N 和 Q 值这几个功能。

算法的基本思路和材料中给出的伪代码相近，我只做出了一些修改。

在扩展结点的判断上，每次在扩展结点的时候，我选择将所有的可能的结点一下子全部构造出来，并且赋值为默认的参数。并用一个标记来记录时候进行过来结点扩展。在判断进行过结点扩展的时候，就直接开始进行选择最好的子节点。

在信心上限搜索树的结点方面，我没有选择应用指针，而使用一批数组来储存他们。其中包括记录子节点的区间的 l，r 数组，记录父节点位置 fa 数组，记录其获胜因数的 win 数组，计算总数 tot 数组，这一个结点要下的位置 xx，yy 数组，判断记录当前局面的情况的 re 数组和记录当前一步为那一方 user 数组。数组的上限为 8000000。

在选择子节点的部分，对于位置为 i 的子节点，选择的公式为:

$$uct = \frac{win[i]}{tot[i] + 0.001} + c * \sqrt{\frac{2ln(tot[fa[i]] + 1.001)}{tot[i] + 0.001}}, user[i] = 0 \tag{1}$$

$$uct = -(\frac{win[i]}{tot[i] + 0.001} + c * \sqrt{\frac{2ln(tot[fa[i]] + 1.001)}{tot[i] + 0.001}}), user[i] = 1 \tag{2}$$

2.2 参数方法

计算的时限设为 2.3 秒，搜索的次数上限为 600000 次。在程序达到这两个上限任意一个时候，程序停止搜索，开始计算输出正确答案。

对于选择结点部分的 c 的取值，我取的是 0.707，即 $\frac{1}{\sqrt{2}}$，的时候，我认为其能力是最强的。

在对局面进行判断的时候，如果当前局面为没有结束，我会返回-2，如果当前局面为自己赢了，就返回 1，对面赢了返回-1，如果为和棋则返回 0。因为我认为，一胜一负的策略和没有搜索到过，或者两胜两负的策略其信心是一样的；并且由于我下棋是要追求胜利，因此和棋毫无意义，因此设为 0，即不影响对于棋局下法的信心判断。

3 实验结果评测

我应用给出的 compete 程序进行了与所有测试用的 AI 的对战。在我的电脑上没有出现超时的情况。

在每轮模拟中，大约可以进行 20 万次的模拟，其结果一般可以得到一个突出的最好的结果。

	2.dylib	4.dylib	6.dylib	8.dylib	10.dylib	12.dylib	14.dylib	16.dylib
测试轮数	2	2	2	2	2	2	2	2
先手胜率	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
后手胜率	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	18.dylib	20.dylib	22.dylib	24.dylib	26.dylib	28.dylib	30.dylib	32.dylib
测试轮数	2	2	2	2	2	2	2	2
先手胜率	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
后手胜率	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	34.dylib	36.dylib	38.dylib	40.dylib	42.dylib	44.dylib	46.dylib	48.dylib
测试轮数	2	2	2	2	3	3	3	3
先手胜率	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
后手胜率	1.00	1.00	1.00	1.00	1.00	0.67	1.00	1.00
	50.dylib	52.dylib	54.dylib	56.dylib	58.dylib	60.dylib	62.dylib	64.dylib
测试轮数	3	3	3	3	3	3	3	3
先手胜率	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
后手胜率	0.67	0.33	1.00	1.00	1.00	0.67	1.00	0.67
	66.dylib	68.dylib	70.dylib	72.dylib	74.dylib	76.dylib	78.dylib	80.dylib
测试轮数	3	3	3	3	3	3	3	3
先手胜率	1.00	1.00	1.00	1.00	1.00	0.67	1.00	1.00
后手胜率	0.67	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	82.dylib	84.dylib	86.dylib	88.dylib	90.dylib	92.dylib	94.dylib	96.dylib
测试轮数	5	5	5	5	5	5	5	5
先手胜率	1.00	1.00	0.80	1.00	1.00	1.00	1.00	0.80
后手胜率	0.80	0.80	0.80	1.00	1.00	1.00	1.00	0.80
	98.dylib	100.dylib						
测试轮数	5	5						
先手胜率	0.80	1.00						
后手胜率	1.00	1.00						

80 至 90 的先手胜率为 96%，后手为 88%；而 90 以上先手为 93.33%，后手为 96.67%，由于存在概率问题，因此可以认为在对阵 90 及以上的对手的时候可以拥有 95 左右的胜率。因此，程序很好的完成了对战任务。