四子棋实验报告

蔡文静 计31 2013011425

1. 实验目的：

在人工智能领域，博弈问题一直是一类具有代表性的研究课题，博弈问题中一些新课题的提出，也不断的推动着人工智能学科的发展。

本实验以四子棋博弈游戏为背景，主要包括了剪枝算法，棋局评价，威胁检测与排除以及同学们自己的创新性算法等方面，主要锻炼了大家对人工智能算法的实际应用能力，从而使大家对人工智能有一个更为直观和真切的体会。

1. 实验内容：

1，游戏规则：

游戏双方分别持不同颜色的棋子，设A持白子，B持黑子，以某一方为先手依次落子。假设A为先手，落子规则如下：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，则不能在该列落子。在图形界面中，如果在某一列的任意一个按钮上点击，会自动在该列的最低端落子。

棋手的目标是在横向、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。先连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的，则为平局。

2，规则扩展：

首先，当两个策略进行对弈时，棋盘的大小是随机的，宽度和高度范围均在【9， 12】

其次，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

3，实验要求：

在提供的实验框架下完成四子游戏的人工智能决策部分，并将生成的策略文件（生成的dll文件）与提供的不同级别的策略文件对抗，并根据积分规则·计算得到的相应分数。

1. 实验设计：
2. 主要算法：alpha-beta剪枝，极大极小算法，递归
3. 数据结构：结构体、二维数组、指针
4. 设计思想：根据极大极小算法的“预估计”思想，将不同的双方落子可能列出来进行评估，在这过程中并根据评估值的上下界大小关系进行剪枝，提高运行效率，得出局部最优的一步。
5. 具体实现：
6. 评估函数：该函数所用参数为\*\*board, M, N, noX, noY,根据该状态的棋局状况进行计算，先将棋盘分成8种状态：死一（a），活一(b)，死二(c)，活二(d)，死三(e)，活三(f)，死四(g)，活四(h)，死一是指在某个方向上（横、竖或两个斜方向）一个棋子，该棋子的一端或两端被对方棋子堵住后不能落子，活一是指在某个方向上，有一个棋子，该棋子两端均可落子；死二、活二、死三······等类似于死一、活一。 一、二表示的是指棋子数目，死活是棋子所处状态描述。这八个状态的权值不同，死一、活一·······的权重依次增加(p1, p2, p3, p4, p5, p6, p7, p8)，计算某棋盘的评估值fp = max – min, max = p1\*a(我方，数量) + p2 \* b（我方） + p3 \* c（我方） + p4 \* d（我方） + p5 \* e （我方）+ p6 \* f （我方）+ p7 \* g（我方） + p8 \* h(我方)；min与之基本一样，只是将“我方”改成“对方”。Fp越大，越利于我方胜出，fp越小，越利于对方胜出。
7. Alpha-beta算法主题框架，int alphaBeta(node \*root, int M, int N, int noX, int noY),即为该函数定义，由一节点root生成其子节点，即下一步落子位置，用for循环遍历N中落子点，确定某一落子后，用该函数的递归算法进行深度优先遍历，当然，递归之前先进行判断，若该子节点不是父节点的第一个节点，，得到父节点的 则此时父节点已有alpha或beta值，则进行是否剪枝的判断；若该节点是第一个节点，则在递归之后，应及时向上回溯（其实就是父节点的递归），递归终止条件是到达叶节点。在到达最初的根节点时，其高度为既定递归深度，确定要走的一步（一列）。
8. 主要难点：指针、递归、评估函数、剪枝判定等
9. 实验结果：

由于局部评估函数不够好，或剪枝的算法写的不够完善等原因，目前的策略效果还不够好，也不够稳定，只能战胜一小部分测试策略，有待改善。

1. 实验结论：

虽然之前做过alpha-beta剪枝书面作业，但是在实现其具体算法时还是出现了一些混乱和错误，本次实验也确实让我深切的体会到了该算法的要点，而且该算法是局部优先，评估函数中各状态的权值也不好确定，只能不断地测试，改正。

最让我感受深刻的是指针的问题，本实验中我用了不少指针，刚开始都犯了一个错误，就是指针初始化问题，还是基础知识掌握不好的原因；

重温了递归算法。