四子棋实验报告

计 64 侯林洋 2016011336

houly16@mails.tsinghua.edu.cn

1 算法思路

本次实验我采用了“信心上限树算法”，算法基本思路如下：

每次以当前局面为根节点建立一颗 UCT 树，从根出发，若当前节点存在可以扩展的子节点，则扩展并选中新扩展出的节点，若当前节点无法扩展，则依据 UCB 算法选出此节点信心上界索引最大的子节点尝试进行扩展，上限信心索引计算公式为：

$$\frac{Q(v)}{N(v)}+c\sqrt{\frac{2\ln(N(u))}{N(v)}}$$

其中 u 为当前节点，v 为 u 的子节点，N(u)为 u 被访问的次数，Q(v)为 v 的收益，N(v)为 v 被访问的次数，c 为一常数。

重复上述操作直到扩展出新节点或者选中终止节点（可区分胜负的局面状态），从此节点出发模拟对弈双方随机落子直至棋盘终了，将本次对局收益（胜利为 1，失败为-1，平局为 0）反馈给此节点的所有祖先。

在规定时限内不断重复上述操作，最终选取根节点收益最大的子节点作为最佳落子点返回。

2 具体实现

为实现上述算法，我新增了 Node 类和 uct 类，分别代表节点和信心上限树。

Node 类：

int x,y;	相较于父节点改变的节点
int who;	落子方，1 为用户，2 为策略
int profit;	此节点收益
int visitednum;	此节点被访问次数
int self;	此节点索引
int father;	父节点索引
int childrennum;	子节点数量
int children[MAX_SIZE];	子节点索引
int next;	下一个可扩展子节点所在的列
void change(…);	重置节点信息
int expand(…);	返回扩展节点索引，无效值-1
int bestchild(…);	返回最佳子节点索引，无效值-1

uct 类：

Node* pool;	内存池
int num;	可用节点索引最小值
int** board;	系统传入的棋局
const int* top;	系统传入的顶端状态
int noX,noY;	不可落子点
int** boardtmp;	模拟时使用到的棋局
int* toptmp;	模拟时使用到的顶端状态
int M,N;	棋盘规模
double c;	参数 c，用于信心上界索引计算

uct(…);	构造函数
~uct();	析构函数
void uctsearch(…);	搜索最佳落子点
int treepolicy(…);	扩展节点，返回节点索引
int defultpolicy(…);	模拟函数，返回收益
void backup(…);	回溯函数

3 算法效果

参数 c 的选择：

测试 c=1.0 对 c=0.0，2.0，3.0 的胜率分别为 100%，80%，90%，最终选择 c=1.0。

对部分测试样例的胜率：

测试样例	胜率
90.dll	85%
92.dll	100%
94.dll	85%
96.dll	90%
98.dll	90%
100.dll	90%

4 实验总结

通过本次实验我加深了对蒙特卡洛规划和信心上限树算法的理解，经过反复的调试，我最终实现了较好的计算效率，算法的实际效果令人吃惊。程序编写者不需要任何的对弈经验，程序依靠大量的随机模拟来确定胜率最大的落子点，对

于较少的棋局状态来说其效果已经优于大多数人的经验,信心上限算法的效果也是显而易见的，大大加速了模拟结果的收敛，事实证明，在绝大多数情况下，收益最大的节点恰恰是模拟次数最多的节点，体现了算法的有效性。