人工智能导论
重力四子棋实验报告

高天宇 2016011348
gaotianyu1350@126.com

1 算法思路

在该实验中我采用的是“蒙特卡洛方法”和“信心上限树算法”，通过构建 UCT 树、随机模拟对战过程、计算收益，从而确定最佳落子点。

每次我以当前状态作为根节点，建立 UCT 树。根的子节点即为每一列可落子的状态。

从根出发，若当前点可以扩展新的子节点，则扩展并选中新扩展的点；若当前点无法扩展新的子节点，则选中以当前点为根的子树中得分最高的叶子节点。得分规则：

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

其中 u 为当前节点，v 为考察的子节点。$N(v)$ 为 v 被回溯的次数，$Q(v)$ 为 v 回溯的总得分。

之后从选中节点开始，随机模拟玩家落子的过程，直到得到结果。然后通过回溯，将收益（胜为 1，负为 -1，平为 0）传给其祖先。在时限内不断进行该随机过程，最终进行收益计算，选择根收益最大的一个子节点作为最佳落子点。

2 具体实现

除了原有的 Strategy 类外，我新增了两个类 Status 和 Uct。Status 类为节点类，存储棋局信息；Uct 类为信心上限树算法类。

Status 成员：

int st[MAX_SIZE][MAX_SIZE]	二维数组，存储棋局信息
int top[MAX_SIZE]	一维数组，存储每列最高点
int cnt, score	为上文中描述的 $N(v)$ 和 $Q(v)$
int cur_player	存储当前局面下一步谁执子
Status *pre	记录 Uct 树中该点的父亲
vector<Status*> child	该点的子节点
int get_winner()	获取当前局面的赢家
bool is_end()	当前局面是否表示游戏结束
bool able_to_expand()	当前节点是否可以扩展新的子节点
Status *best_child()	返回得分最高的子节点
Status *best_result()	返回最好的落子结果
Status *get_random_next()	随机落子并生成新节点
Status *expand()	扩展当前节点

Uct 成员：

Status *tree_policy(Status*)	选择最佳的要执行 default_policy 的节点
int default_policy(Status*)	从输入进去的状态开始随机落子并返回结果
void backup(Status*, int)	回溯更新选中节点的祖先的值
Point *uct_search(Status*)	在一定时间内，扩大 Uct 树并确定最优落子点

3 实验结果

这里只列出与 90、92、94、96、98、100 这 6 个 AI 的对战结果：

对战 AI 名称	我的 AI 胜率
90.dylib	90%
92.dylib	70%
94.dylib	80%
96.dylib	40%
98.dylib	80%
100.dylib	90%

4 总结

本次完成大作业的过程中，学习到了蒙特卡洛模拟方法、信心上限树算法，并将其应用到对战策略中，取得了较为理想的测试结果。

感谢马老师和助教的悉心指导！