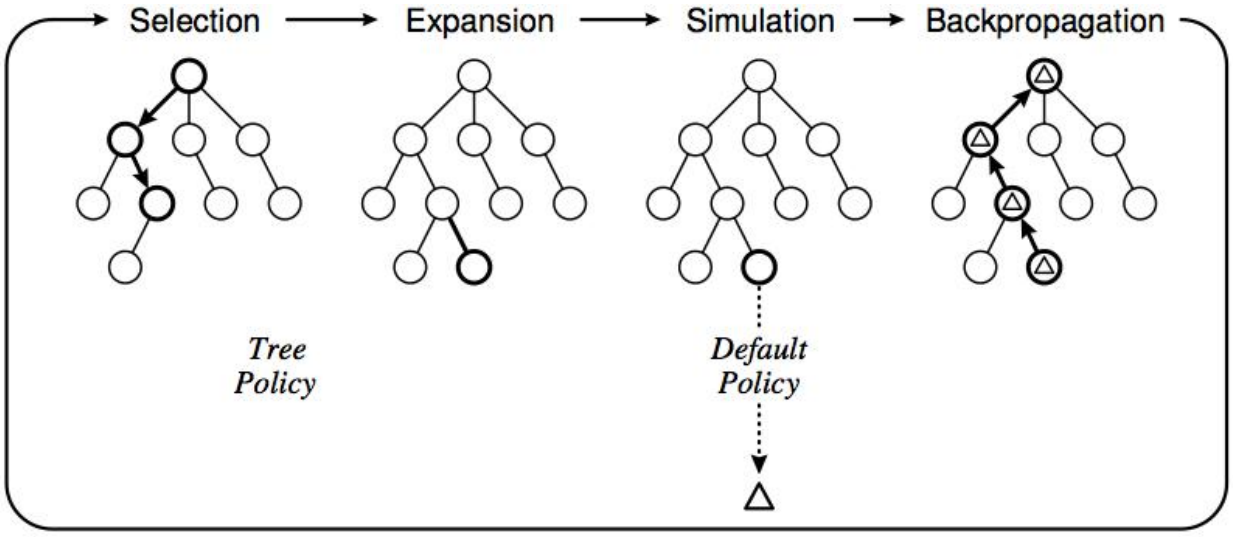
四子棋实验报告

计13 沈佳茗 2021010745

一、设计思路

1.蒙特卡洛树搜索

蒙特卡洛树包含四个步骤，分别为选择（Selection）、扩展（Expand）、模拟（Simulation）和回溯（Back propagation）。搜索树的每个节点对应一种下棋状态，每一个节点的子节点代表从这个节点出发，所有可能下出的棋盘状态。



(1) 选择

从根节点出发，沿搜索树从上至下寻找每一层的最优节点，知道寻找到一个最需要被扩展的节点。可以将节点分为三类：该节点所有可行动作都已被扩展过；该节点还有可行动作未被扩展；该节点对应的棋局已结束。对于第一类节点，我们寻找到它的最优子节点（在实现中采用寻找 UCB 值最大的节点），并对这个子节点继续实行选择操作；对于第二类节点，随机选择它的一个未被扩展的节点，对这个子节点进行扩展；对于第三类节点，则直接进行回溯操作。

(2)扩展

对选择过程中选择的节点进行扩展操作，即建立一个子节点，这个子节点表示的是最迫切被扩展的节点的盘面在操作了某一个尚未拓展的动作之后的局面。

(3)模拟

从上述子节点对应的盘面开始，随机走子，进行蒙特卡洛模拟，直到分出胜负，获得收益。

(4)回溯

在模拟结束后，对该节点的祖先从下至上更新收益和选择次数。

2.信心上限树算法 UCT

信心上限（UCB）的计算公式为 $I_j = \bar{X}_j + c\sqrt{\frac{2ln(n)}{T_j(n)}}$ ，其中 $\bar{X}_j$ 是所有收益的均值，n是到当前这一时刻为止所访问父节点的总次数，$T_j(n)$ 是到当前这一时刻所访问当前结点的总次数。

信心上限树算法 UCT 将信心上限算法与蒙特卡洛搜索算法相结合，用于选择可落子点。在每次选择节点时，根据 UCB1 选择信心上限值最大的节点。

二、实现方法

使用蒙特卡洛信心上限树算法。

构造 Board 类，存储棋盘，由于对每一个节点创建棋盘会消耗大量空间，所以在每一步搜索中，可以根据这一步走子更当前棋盘信息，而不用再创建一个新棋盘。构造 Node 类存储节点的子节点，上一步走子的列，是不是极小节点，访问次数，收益。构建 MonteCarlo 类，作为蒙特卡洛搜索树。

在每一步开始前，首先判断是否有必胜、必负和对手必胜走法。对于必胜走法，则直接走子；对于必负走法，存储列的编号，并在最后决定走子时避开这些列；对于对手必胜走法，则堵住这个位置。

然后进入蒙特卡洛树搜索。从根节点（当前局面）出发，依次执行选择、扩展、模拟和回溯四个步骤，重复上述四个步骤，直到运算时间到达 2.6 秒。

最后，找到根节点访问次数最多的子节点作为落子的位置。

在实验过程中，最开始在模拟过程中，让其完全随机走子，但实际上，若下法中存在自己必胜或对手必胜的走法，则在一般下棋时，人们都会采用必胜走法，因此在面对这两种情况时，走法是固定的，即自己直接走子胜利，或者堵住对方即将要赢的位置。并且在观察对局中发现，有时候走子后，会让对方形成必胜局面，因此一开始针对这三种情况在进入蒙特卡洛搜索树前进行特殊判断，但这让胜率有了一些提高。所以又尝试将前两种判断加入模拟过程，胜率又得到提升。猜想原因可能是加入了这两种判断的模拟过程提高了模拟的有效性，在更新某个点的收益时，完全舍弃了没有用的局面，而只生成了应该下棋的局面。

三、实验结果

总局数	胜	负	平	胜率	采取策略
100	99	1	0	99%	在进入蒙特卡洛树之前增加对必胜、必负和对手必胜的判断，在模拟过程中增加对必胜和对手必胜的判断
100	98	2	0	98%	在进入蒙特卡洛树之前增加对必胜、必负和对手必胜的判断，在模拟过程中增加对必胜和对手必胜的判断
100	98	2	0	98%	在进入蒙特卡洛树之前增加对必胜、必负和对手必胜的判断，在模拟过程中增加对必胜和对手必胜的判断
100	94	6	0	94%	模拟过程完全随机
100	96	4	0	96%	模拟过程完全随机
100	93	7	0	93%	模拟过程完全随机

总局数	胜	负	平	胜率	采取策略
100	97	3	0	97%	在进入蒙特卡洛树之前增加对必胜、必负和对手必胜的判断，模拟过程完全随机
100	96	4	0	96%	在进入蒙特卡洛树之前增加对必胜、必负和对手必胜的判断，模拟过程完全随机
100	97	3	0	97%	在进入蒙特卡洛树之前增加对必胜、必负和对手必胜的判断，模拟过程完全随机