LAB3 四子棋 实验报告

Qianyi Fan 2020013250

May 27, 2024

采用策略

四子棋实验通常可以通过 alpha-beta 剪枝算法或者蒙特卡洛树搜索（MCTS）方法实现。由于而 alpha-beta 剪枝的性能在很大程度上依赖于启发式评估函数的质量，而蒙特卡洛树搜索的算法只通过不断随机的模拟来改进策略，因此选择了蒙特卡洛树进行实现。蒙特卡洛树不需要专家知识的特点也是未来优化博弈问题的发展方向。

本实验主要实现了一个基础的 MCTS 算法，包括四个步骤，选择、扩展、模拟和回溯。在选择步骤中，算法从根节点开始，依次计算子节点的 UCB1（Upper Confidence Bound for Trees）值，兼顾过去表现优秀的节点和未被探索的节点平衡，选择子节点；在扩展步骤中，若选择的节点为叶节点且未终结，则为其生成所有可能的合法动作的子节点。每个子节点代表一个新的游戏状态。在模拟步骤中，从当前节点开始，通过模拟游戏的多次随机走子直到游戏结束，观察该节点双方的获胜情况，以便评价走到该节点的游戏策略的优劣。最后通过回溯步骤，算法从模拟结束的节点自下而上更新经过路径上所有节点的访问次数和胜利次数，从而记录该次模拟信息并优化整棵搜索树。通过在固定的时间和模拟次数的范围内不断迭代更新搜索树，从而得到类似于未来发生的棋局变化的 “理智的先知”，以便最后做出下一步走子的决策。

MCTS 在处理具有高度不确定性和随机性的环境中表现尤为出色。即使没有实现必胜或者必败的判断，我们也发现蒙特卡洛树对于下一步走子或者下两三步走子对方或者我方可能存在的必胜策略存在较充分的感知能力，能够通过当前局面以及接下来每一个状态的 UCB 值进行有效评估。

实现方法

`Strategy.cpp` 调用了 `MonteCarloTreeSearch` 类的接口，通过其构造函数将当前棋局状态传入类中，最后可通过 `get_best_action` 接口获取单步棋局的模拟结果。在每一步模拟中，

循环调用四个方法进行模拟，直到最后达到 2 秒后停止模拟，以免发生超出 3 秒决策时间的情况。

蒙特卡洛树搜索的算法主要封装在 `MonteCarloTreeSearch` 类，包含四个主要方法。

`Selection(Node node)`：从根节点开始，计算所有子节点的UCB值，选择UCB值最大的子节点，直到到达一个叶节点。
`expansion(Node node)`：如果节点不是终端节点，生成并返回一个子节点。
`simulation(Node node)`：从选择好的节点开始进行随机模拟，直到达到棋局的终状态，返回模拟结果。
`backpropagation(Node node, int winner)`：将模拟的结果反向传播回根节点，更新每个经过节点的访问次数和胜利次数。

在每一次模拟之后，还需要调用一次 `MonteCarloTreeSearch` 类的析构函数，递归地释放蒙特卡洛树中节点的内存。(当没有释放时，大约 5 次下子之后在测试平台上就会出现 MLE 的 bug)

对抗结果的统计数据

总胜率：80%

AI 编号	胜	负	平
2	10	0	0
4	10	0	0
6	10	0	0
8	10	0	0
10	10	0	0
12	10	0	0
14	10	0	0
16	10	0	0
18	10	0	0
20	10	0	0
22	9	1	0
24	10	0	0
26	10	0	0
28	10	0	0

30	10	0	0
32	10	0	0
34	10	0	0
36	10	0	0
38	9	1	0
40	10	0	0
42	10	0	0
44	10	0	0
46	7	3	0
48	10	0	0
50	10	0	0
52	8	2	0
54	8	2	0
56	8	2	0
58	10	0	0
60	9	1	0
62	10	0	0
64	8	2	0
66	4	6	0
68	9	1	0
70	8	2	0
72	9	1	0
74	7	3	0
76	9	1	0
78	7	3	0
80	8	2	0
82	7	3	0
84	4	6	0
86	7	3	0
88	5	5	0
90	3	7	0
92	3	7	0
94	0	10	0
96	1	9	0

98	3	7	0
100	1	9	0