IAI重力四子棋实验报告

刘蕴皓 2021012884

算法思路

本实验采用了蒙特卡洛树搜索+信心上限算法，构建了UCT树进行了落子位置的收益统计，并在此之上针对四子棋特性进行了优化，以此决定每步的最终落子位置。

MCTS

蒙特卡洛树搜索（Monte Carlo tree search；MCTS），是一种用于求解具有巨大状态空间的博弈问题的搜索算法。它在模拟随机游戏对局的基础上，通过统计每个节点的胜利次数来指导搜索过程，从而找到最优的下一步行动。

步骤

1. 选择（Selection）阶段：从根节点开始，根据一定策略选择子节点，直到达到一个未完全展开的节点。
2. 扩展（Expansion）阶段：对于未完全展开的节点，根据一定策略选择一个未访问的子节点，将其加入树中。
3. 模拟（Simulation）阶段：从扩展的子节点开始，随机模拟一局游戏对局，直到游戏结束。
4. 回溯（Backpropagation）阶段：根据模拟的结果，将得分反向传播到所有经过的节点上，更新它们的胜利次数和访问次数。

具体实现

定义蒙特卡洛树节点结构体MonteCarloTreeNode，记录该点的落子位置，访问次数，得分，父子节点及尚未扩展的可落子位置。

1. 选择阶段：根据当前树的情况，选择一个需要向下扩展或进行模拟的节点。从根节点开始，对于任意一个不存在尚未扩展的子节点的非叶节点，选择其子代中的最优节点（由UCB判断），直至该节点需要扩展或找到叶节点。
2. 扩展阶段：对于已选择的节点，若其不是叶节点，则从此节点中的可扩展位置中挑选一个，更新棋盘并建立新的节点，加入当前节点的子代列表中。
3. 模拟阶段：从已更新的棋盘状态开始随机落子，直至任意一方获得胜利或平局。
4. 回溯阶段：更新当前节点及其所有祖辈的访问次数，并根据模拟结果更新分数：若节点对应的玩家失败则-1，胜利则+1，平局不变动。

UCT

UCB1公式

$$ UCB1 = score_i/visit_i + c\sqrt{\frac{2\ln{visit_p}}{visit_i}} $$

- $score_i$ 表示当前节点 i 的得分。
- $visit_i$ 为当前节点i被访问的次数。
- $visit_p$ 为当前节点的父节点被访问的次数。

- c 为探索参数。c 越大，就会越照顾访问次数相对较少的子节点。

具体实现

在上述过程中的选择阶段，根据UCB1公式计算各节点分数，并选出最优节点。经多次实验，设定探索参数为0.6，既不至于在已有的高分点位上固步自封，也不会太过瞻前顾后而错失良机。

其他优化方案

一步定胜负

对于只需一步便可有玩家胜利的情况进行了特殊判断。在每次模拟时，若监测到某个可行的落子点是任意一方三连子的两端或是四连子中间缺失一个的情况，则取消随机方法，立即落子于此进行决胜或防守，由此来降低由于某些其它点位胜率较高而未能直接进行决胜或防守的情况，我认为这也更为符合人类在下棋时的直觉。对于当前步骤亟需一步定胜负的情况下，则直接跳过MCTS过程，也在一定程度上降低了时间成本。

位置权重

根据围棋“金角银边草肚皮”的布局思路，我尝试在随机落子策略中增加了靠近中间位置的权重，不过对于最终结果似乎并无显著影响，甚至于略有下降。根据对局的过程监控，我发现无论是否加入这一策略，AI的布局落子大多是靠近中间的。由此可以猜想在MCTS过程中AI已经根据大量模拟得出了落子靠中胜率更高的规律，遂放弃这一优化策略。

总结

最好版本在默认的批量测试集中 96胜 4负 0平。 本次试验中，我学习并实践了MCTS+UCB方法，并针对实际情况进行了适当优化。经过本次试验，我对人工智能的博弈策略等有了更为深入的理解。