四子棋实验报告

张益铭 2021010552 车13

基本思路

本次实验我选择的是蒙特卡洛树搜索算法。MCTS算法不依赖于特定的游戏规则或领域知识，具有较强的通用性。它可以应用于任何具有明确终局状态的决策问题，而 Alpha-Beta 剪枝通常需要特定的启发式评估函数来估计节点的价值，评估函数一般不容易设计，因此此次试验选择MCTS。

蒙特卡洛树搜索基本流程包括四个主要步骤：

- 选择（Selection）：从根节点*R*开始，连续向下选择子节点至叶子节点*L*。下文将给出一种选择子节点的方法，让游戏树向最优的方向扩展，这是蒙特卡洛树搜索的精要所在。
- 扩展（Expansion）：除非任意一方的输赢使得游戏在L结束，否则创建一个或多个子节点并选取其中一个节点*C*。
- 仿真（Simulation）：再从节点*C*开始，用随机策略进行游戏，又称为playout或者rollout。
- 反向传播（Backpropagation）：使用随机游戏的结果，更新从*C*到*R*的路径上的节点信息。

下面将从这四个部分来介绍我的算法。

方法

数据结构

在实现中，我们主要使用以下数据结构：

- **Node（节点）**：表示搜索树中的一个节点，包含位置、胜利次数、访问次数、所属队伍、子节点列表等信息。
- **MCTS 类**：包含了实现 MCTS 算法的主要方法，如选择、扩展、仿真、回溯等。

选择

在选择阶段，我们使用上置信界（Upper Confidence Bound, UCB1）策略来平衡探索和利用。UCB1 是一种在多臂赌博机问题中常用的策略，能够在保证探索新节点的同时，优先选择当前已知收益较高的节点。

信心上限我选择如下公式进行计算：

$$\text{Confidence} = \frac{\text{win}}{\text{total}} + c\sqrt{\frac{2\log(\text{parent total})}{\text{total}}}$$

其中total表示节点被访问的总次数，win表示节点被访问且获胜的次数，parent total表示节点的父节点被访问的总次数。在本次实验中，超参c被选取为1。

最终会返回信心最高的子结点，具体由`double MCTS::calcBelief(int nodeID);`和`int MCTS::getBestChild(int nodeID);`实现。

扩展

扩展阶段，我们在当前节点上生成所有可能的子节点，并初始化这些子节点的状态，之后会在棋盘上模拟走出这一部，更新棋盘。为了避免无意义的扩展，我们会跳过已经满列的列，具体由`int MCTS::expand(int nodeID);`实现。

仿真

在仿真阶段，我们从当前节点开始，随机选择动作直到游戏结束。仿真的目的是估计当前节点的胜率。我们假设双方都以随机策略进行游戏，这样可以快速评估不同决策的潜在效果，具体由`int MCTS::getWinTeam(int nodeID);`实现。

- **随机选择动作**：仿真过程中随机选择列进行落子，确保仿真的多样性。
- **判断终局状态**：在每一步落子后，检查是否形成四子连珠或棋盘已满，以终止仿真。

通过随仿真，能够估计当前节点的胜率。这种方法虽然简单，但在大量仿真中能够有效逼近真实的胜率分布。

反向传播

回溯阶段将仿真的结果从扩展的节点回溯到根节点。每个节点的访问次数和胜利次数根据仿真结果进行更新。这一步通过类似递归的方式实现，从叶节点逐步向上更新每个父节点的状态，由`int MCTS::UCT(clock_t start, const int *top, const int *_board);`的内部实现。

在 UCT 算法中，我们反复执行上述四个步骤，直到达到预设的计算时间(2.85s)或节点数。最终，选择胜率最高的子节点作为最佳决策。

评测结果

最终在saiblo平台测得100场测试中，胜90场，负10场，平0场，胜率为90%