**数据结构课程设计实验报告**

**1 分工与合作**

**1.1具体分工完成的模块：**

1.移动生成（getValidMoves）：  
(1)功能描述：

①完整扫描7×7棋盘，计算当前棋手的所有合法落子点。

②输出结构化的Move对象数组，包含起点/终点坐标。

(2)技术特点：

①方向覆盖：利用预定义的24方向delta数组，覆盖全部移动可能性（8邻域+16跳跃）。

②性能优化：采用快速边界检查（inMap函数）优化性能。

2.Zobrist哈希优化：

①使用mt19937\_64生成高质量随机数填充ZobristTable。

②设计updateHash函数实现增量式哈希更新，哈希值覆盖棋盘所有可能状态（空/黑/白）。

③将棋盘状态转换为64位唯一标识，使置换表查询效率显著提升。

3.评估函数（evaluate）

根据棋局的不同阶段（早期、中期、后期），动态调整评估权重。

评估指标包括：

棋子数量：己方棋子数与对方棋子数的差值。

行动力：合法落子点的数量。

中心控制：棋子靠近棋盘中心的程度。

潜在翻转：己方棋子周围敌方棋子的数量。

边角控制：占据棋盘四角的棋子数量。

孤立子惩罚：没有邻居的己方棋子数量。

稳定子奖励：无法被翻转的己方棋子数量。

前沿子惩罚：容易被翻转的己方棋子数量。  
4.剪枝函数（Alpha-beta剪枝优化的Minmax搜索）  
功能描述：

实现Alpha-Beta剪枝算法，结合置换表、杀手启发和历史启发，优化搜索效率。

核心优化：

置换表缓存：存储已搜索节点的评估值，避免重复计算。

启发式搜索：杀手启发和历史启发（killerMoves & historyTable）：

记录在特定深度下表现优异的移动，优先尝试；根据移动的历史表现调整搜索顺序，优先尝试高价值移动。

迭代加深搜索：

逐步增加搜索深度（MAX\_DEPTH = 8），确保在时间限制内找到最佳解。

关键技术：

1)哈希标记处理：精确处理三种哈希标记（精确值/下界/上界）。

2)移动排序启发式：先验高分移动优先搜索。

3)状态保存/恢复机制：支持回溯操作，确保搜索正确性。

5.搜索最佳移动（findBestMove）

功能描述：

在保证不超时的前提下结合Alpha-beta剪枝优化的Minmax搜索，在所有合法步骤里寻找所能找到的最佳步骤。同时，加入Zobrist哈希防止重复局面，返回所能找到的最佳移动数组。

**1.2协同工作亮点**

接口设计：

移动生成模块输出适配的搜索框架，Move对象结构统一。

性能配合：

Zobrist哈希与置换表协同提升搜索效率，减少重复计算。

评估集成：

动态评估函数无缝嵌入Alpha-Beta剪枝流程，支持不同阶段权重调整。

联合调试：

共同优化杀手启发与历史启发的参数配合，提升搜索速度与质量。

**2 算法思想**

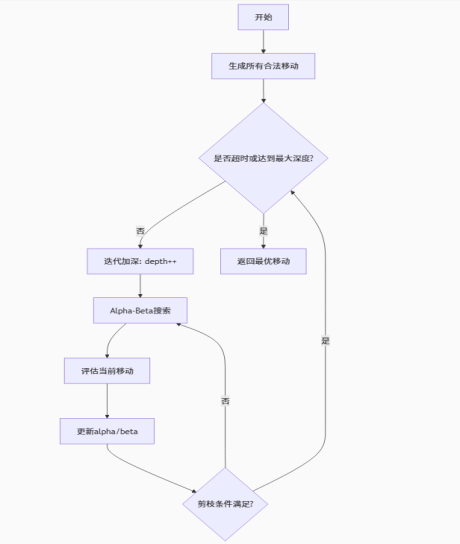
**2.1总体思路**

核心算法：  
 核心算法基于Alpha-Beta剪枝的博弈树搜索，结合迭代加深和时间控制。使用7×7二维数组表示棋盘状态，通过置换表缓存历史局面结果，并采用Zobrist哈希快速比对棋盘状态。策略上设计多阶段评估函数，根据棋局阶段动态调整权重，并应用杀手启发和历史启发加速搜索。

游戏逻辑与状态表示：  
 棋盘表示使用gridInfo[7][7]存储棋盘状态（0=空，1=黑棋，-1=白棋）。全局变量blackPieceCount和whitePieceCount实时统计棋子数。移动规则通过delta[24][2]定义24种方向（8邻域+16跳跃）。翻转敌方棋子在落子后检查8邻域，翻转相邻敌方棋子（ProcStep函数实现）。合法移动生成由getValidMoves遍历所有己方棋子，生成可达的空位列表。

搜索算法：  
 Alpha-Beta剪枝通过递归搜索从当前局面出发，模拟双方落子，交替最大化/最小化评分。剪枝条件为alpha（当前玩家能保证的最高分）和beta（对手能接受的最低分），当alpha >= beta时停止搜索后续分支。迭代加深逐步增加搜索深度（depth=1 → MAX\_DEPTH），每次迭代调用alphaBeta函数。超时终止在每次深度增加前检查时间，确保在980ms内返回结果。

演示图如下：



评估函数设计：  
 分阶段权重：开局（EARLY）棋子数阈值≤15，侧重控制角落、避免孤立子；中局（MID）棋子数阈值≤35，侧重提升中心控制、增加行动力；残局（LATE）棋子数阈值>35，侧重最大化棋子数、稳定子奖励。评分指标包括物料差、行动力差、中心控制、孤立子惩罚和稳定子奖励。

**2.2优化技术：**

置换表使用currentHash（Zobrist哈希值）作为哈希键，存储内容为{depth, value, flag}，避免重复计算相同局面。杀手启发记录每层历史最佳移动，优先搜索这些移动（killerMoves[depth][2]）。历史启发通过historyTable[x0][y0][x1][y1]统计移动历史得分，优先高分移动。

哈希与回溯：  
 Zobrist哈希初始化时为每个棋盘位置和状态生成随机64位数（initZobrist）。动态更新在每次落子或翻转时，异或对应位置的哈希值（updateHash）。状态保存与恢复在搜索前保存棋盘、棋子数、哈希值到临时变量，回溯时还原。

杀手移动和历史启发式（Killer Moves & History Heuristic）：

记录在特定深度下表现优异的移动，优先尝试；根据移动的历史表现调整搜索顺序，优先尝试高价值移动。

迭代加深（Iterative Deepening）：

逐步增加搜索深度，确保在时间限制内找到最佳解。

数据结构：

除了STL里的很多库之外，还有：

7×7二维数组表示棋盘状态；

置换表（Transposition Table）缓存历史局面结果；

Zobrist哈希快速比对棋盘状态等等

策略：

多阶段评估函数：根据棋局阶段（开局/中局/残局）动态调整权重。

启发式优化：杀手启发（Killer Moves）、历史启发（History Heuristic）加速搜索。

所用方法的特别、新颖或创新之处：

动态评估权重：针对不同阶段（棋子数量≤15/≤35/其他）设计差异化评分策略，提升决策适应性。

混合剪枝技术：结合置换表、杀手启发和历史启发，减少冗余搜索节点。

Zobrist哈希优化：通过位运算高效更新哈希值，支持快速状态回溯。

**3.总结**

**3.1性能表现：**

搜索效率：通过置换表、杀手移动与历史启发式，搜索效率显著提升，能够在980ms内完成深度为8的搜索。

棋力表现：AI在多数情况下能够选择较优落子，尤其在中期与后期阶段，能够通过控制中心与边角区域取得优势。

得意之处：动态评估函数和混合剪枝策略显著提升胜率

**3.2问题瓶颈**

·性能瓶颈：

问题：初始版本搜索深度不足

解决：引入置换表和杀手启发，深度提升且不超时

·评估函数偏差：

问题；早期过度重视角落导致被动，后期的决策可能过于保守

解决：分阶段调整权重，根据不同阶段的局面调整自身的进攻策略

·重复走子

问题：有时候决策失误导致重复走子

解决：引入Zobrist哈希和简单的位棋盘(BitBoard)

**3.3优化方向**：

1)搜索算法优化：尝试更高级的剪枝策略（如Null Move Pruning）或并行搜索；加入主要变量搜索（PVS）加速搜索，减少冗余计算，提高决策准确率；已经观察到剪枝在后期可能会出现决策失误的情况但前期的决策较准确，而MCTS算法及其延伸算法在前期由于模拟数据不够导致决策较差，可以采用混合树搜索（Hybrid Tree Search），结合UCT算法，兼容两者的优点进行决策。

2)动态对手模型识别：根据观察结果，大多数的决策模型集中在暴力贪心或二步贪心、Alpha-beta剪枝优化的Minmax搜索或者MCTS算法及其延伸，可以考虑在前期下棋时只记录对手的棋盘，根据三种决策模型各自的特点尝试猜测对手的决策模型，据此调整自身的决策和评估参数，以适应对手的进攻节奏。

3)结合简单的机器学习做出决策引导：引入3层左右的小型MLP模型，在PyTorch里离线训练，作为自身决策的前向引导，可以在极短的时间内辅助自身做出更好的决策。

4)引入更复杂的位棋盘和权重表(PST)：位棋盘负责“怎么快”，PST负责“评得准”，两者结合可以给AI提供更好的决策建议，在加速的同时节省更多时间来进行决策，同时提高决策的准确率

**4.参考文献**

(1)黑白棋AI设计探究\_黄海同

(2)计算机博弈算法在黑白棋中的应用\_彭之军

(3)蒙特卡洛树搜索实验设计——以黑白棋为例\_张宇辉

(4)全局求解Minimax线性规划的分支定界算法（英文）\_王慧满

(5)面向蒙特卡洛树搜索算法的深度神经网络的设计与实现\_明红兵

(6)Accelerating Deep Neural Net work guided MCTS using Adaptive Parallelism

(7)几种智能算法在黑白棋程序中的应用\_柏爱俊0511@USTC

(8)[https://blog.csdn.net/caozixuan98724/article/details/10321375](https://blog.csdn.net/caozixuan98724/article/details/103213795)

(9)<https://blog.csdn.net/weixin_42165981/article/details/103263211>

(10)<https://oi-wiki.org/search/alpha-beta/>