2024 年春季学期

数据结构课程设计赛道 B 实验报告

张艺卓 [1] 武锦旗 [2] 吴仲博 [3] 陆学优 [1]

[1] 计算机科学与技术学院 2022 级 32 班

[2] 计算机科学与技术学院 2022 级 23 班

[3] 计算机科学与技术学院 2022 级 30 班

1 分工与合作

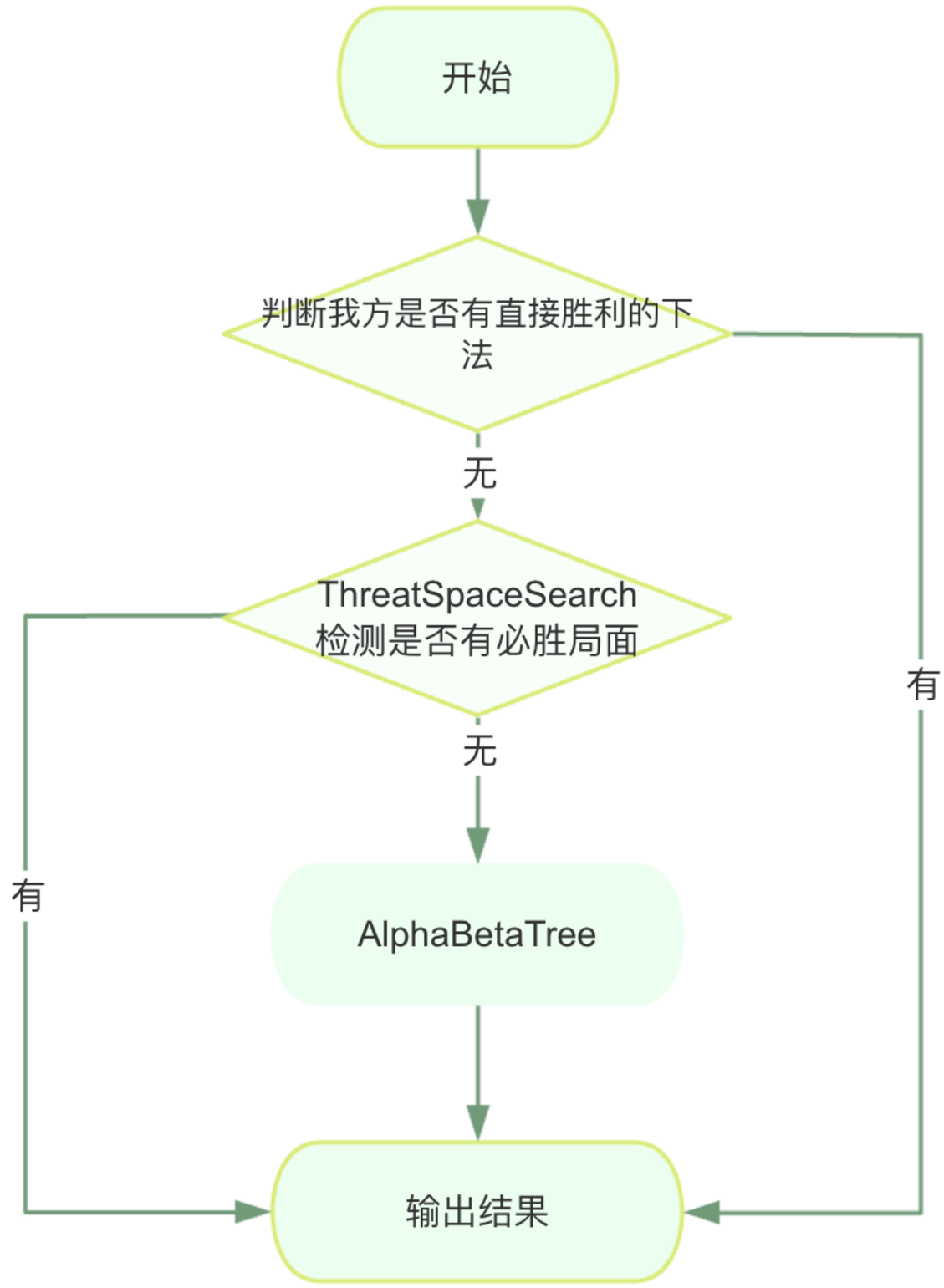
姓名	学号	分工	占比
张艺卓	21220122	实验报告的主要撰写者 博弈树代码的编写 博弈树代码的优化 博弈树参数的调整和最终确定	28%
武锦旗	21220716	代码的主要撰写者 TSS 代码的编写，路表代码的编写 TSS 代码的优化，路表代码的优化 路表和博弈树的结合以及路表和 TSS 的结合 代码中各个模块的包装	32%
吴仲博	21221429	评估函数与 TSS 相关论文的查找与分析 参考遗传算法优化参数选择 综合博弈树、局部评价与优化后的参数编写过渡 BOT 乜_w	20%
陆学优	21221020	路和博弈树相关论文的查找 基于全局评估和局部评估运用博弈树写出过渡 BOT nihao 参与模型的参数的调整	20%

2 算法思想

2.1 总体思路

2.1.1总体策略

在项目中，我们致力于通过算法的力量来模拟和优化棋局中的决策过程，以期在复杂的棋局中实现高效、精确的走法选择。为了达到这一目标，我们采用了博弈树 和 TSS 作为核心的搜索机制，结合基于路的评价策略来增强局面的评估能力。除此之外，我们在检索策略、模拟轮数等方面均做多次测试，不断优化调参，同时也从改进存储的数据结构入手，用路表代替查询，力求达到决策的最佳效果。



2.1.2 博弈树+Alpha-Beta 剪枝

博弈树是一种用于分析和解决博弈问题的工具，它以树状结构表示参与者的所有可能行动和决策。在博弈树中，每个节点代表一个特定的游戏状态，而每个分支代表一个参与者的一个决策或行动。我们可以根据通过博弈树模拟一些对局和落子情况来选择让我们收益最大的落子。因为我们还有 TSS 所以不能使博弈树的复杂度太高，但是我们不能遗漏可能的方案数，所以我们放弃了深度较深的树，选择了宽度大的树。不遗漏我们选择的可行点的任何组合。

Alpha-Beta 剪枝是一种用于优化博弈树搜索的算法。它通过剪除不必要的搜

索分支来减少搜索空间，从而提高搜索效率。算法中维护两个变量 alpha 和 beta，分别代表当前最好的最大值和最小值。当搜索到达某个节点时，如果该节点的值超出了 alpha 和 beta 之间的范围，则可以对其父节点进行剪枝，因为该父节点的值不会被选择。这样，在搜索过程中，可以跳过对一些不可能影响最终结果的节点的搜索，从而减少了搜索空间，提高了搜索效率。

在博弈树实现中，我们选择棋盘中已落点的周围两层点作为备用的可行点，然后计算各个点对的分数。对于第一层，我方落子应该优先选择对于我方更优的落子方案，这样才能使得 Alpha-Beta 剪枝可以剪掉更多，进一步优化我们的搜索速度。同理在第二层的时候我们也应该用更优于对手的子。所以我们预处理出两个vector<pair<pair<int,int>pair<int,int>>>类型的数据 作为我方和对方的落子选择顺序。

2.1.3评估函数

我们对于棋面的评估选择了基于路的评价策略。首先是路的定义 - - 在棋盘上连续 6 个点能够连成一条直线，则称为路。如果路上所有棋子都为黑子，则这条路的颜色是黑色；如果该路上所有棋子都为白色，则这条路的颜色为白色；如果该路上没有棋子或同时含有 2 种颜色的棋子，则这条路没有颜色。在有色路上已经含有棋子的个数称为路的长度，并成为这条有色路的“几路”

基于路的评价策略是将棋盘上每一条可能形成连珠的直线抽象成一条“路”，并对这些“路”进行评分，以评估棋局优劣。这种策略首先建立路模型，将棋盘的每一列、每一行以及两条对角线抽象成一条路。然后对每一条路进行评分，根据路的棋子分布情况，给每一条路打分。接着，将所有路的评分进行综合，得到

对整个棋局的评分。最终评分综合考虑己方和对方的评分差异，以指导落子。基于路的评价策略模型简单直观，便于实现，但难以充分反映棋局的复杂性，容易漏掉关键因素。我们将评价函数嵌入到棋盘的结构体中，每加入一步棋或者撤销一步棋，我们棋盘的分数是动态变化的。为了使得我们的复杂度不高，我们采用全局初始化 + 局部修改的动态评分方案。在初始化棋盘结构体时，我们会得到一个初始化的分数，然后在后续 add / sub 点中动态地修改分数。

2.1.4 威胁空间搜索（TSS）

Threat Space Search（TSS）是一种用于棋类游戏的搜索算法，特别是在解决无偶然因素的完全信息棋类游戏时非常有效。TSS 算法的主要思想是在搜索过程中利用游戏中的威胁来指导搜索，从而找到有效的走法或策略。

TSS 的实质就是产生威胁，使得对方一直处于被威胁状态，直到我们在某一步取得成功。在六子棋中，4 子和 5 子是对对方产生威胁的棋形。因而我们便需要找到落子之后可以对对方产生威胁的点，而要想让 TSS 的深度更深，更快一步地找到我们必胜的路我们必须在每一层快速地找到我们想要的点。因而我们引入了路表这一存储方式，可以让我们更快地寻找能造成威胁的点。我们采用了路表来存储棋盘中我方与对方的有效棋路，这一存储方式可以让我们很快地获得敌我的目前棋盘情况。这里我们定义了路的结构体，其中包含起始点，方向，活子数之类的与路相关的特征量。我们路表中实质就是根据活子数和敌我方来分类。而且我们采用全局初始化和局部修改的策略，使得我们路表的维护可以更快。

TSS 算法的优势在于它能够在搜索过程中利用棋类游戏的特定知识，这使得它比通用的搜索算法更有效。通过专注于威胁，TSS 能够更快地找到关键的走法，

从而提高搜索效率。但是 TSS 也有一定的劣势，如果我们在 TSS 中不能取得获胜，我们就会选择博弈树，这也进一步让我们的下棋策略更加均衡和完善。

2.2 所用方法的特别、新颖或创新之处

2.2.1路表的建立和维护---全局初始化和局部修改

对于一个棋盘的状态我们用了一个结构体路表来维护，维护的内容包括：棋盘各点的颜色，各路的状态；提供的操作包括：查询当前棋盘局面的分数，模拟下白棋或黑棋，撤销下棋，查询不同的棋形包括在哪些路中。

对于路的维护，如果每次都是扫描一遍，这会是很大的开销。经分析得知棋局状态的变化均由新增棋子引起，而新增的棋子只影响以它为基点的水平方向，垂直方向，左斜方向，右斜方向四个方向上的棋型，对其他位置并无影响。对于路来说新增棋子也只影响了这四个方向上敌我双方“ 路”的数量所以我们在评分中，采用全局初始化和局部修改的策略。有效地降低时间成本，并且减少了很多不必要的运算。

2.2.2博弈树优化搜索顺序

在我们的项目中，我们采用了 Alpha-Beta 剪枝，这一策略使得我们博弈树的时间可以更短，为我们剪掉很多不必要的枝叶。为了进一步发扬这一剪枝的优点，我们最好先搜索最优的策略，这样做可以让它为我们剪掉更多的枝叶，让我们的时间更短，博弈树的效果更优。为了达成这样的一个目标我们选择优化搜索顺序。

（1）先拿到我们备选点，将这些点组合成多个点对，放入 vector 中。

（2）遍历存储下棋方案的 vector，分别对这些点对求出在棋盘上的分数。

（3）对于我方下棋顺序，应该从大到小排序。对于对方的下棋顺序，应该从小到大排序。

2.2.3路表结合 TSS 和博弈树

（1）对博弈树的影响

我们所采取的是基于路的评价策略，因此我们需要各种有效路的数目。路表可以直接为我们提供我方和对方的路的信息，所以我们可以 0(1）的时间查询到我方和对方的各个有效路的数目，并能够 0(1）时间内直接获得当前棋盘我方的得分。

（2）对 TSS 的影响

TSS 制造威胁时我们需要找到可以制造威胁的点位即可以让棋盘上产生 4 连或者 5 连，这时我们便需要快速找到 2 连或者 3 连的路，找出这些路中可下的点位，通过路表，在 TSS 选点时就可以 0(1）的找到我们可用的点，并且可以 0(1）的时间查询到我方和对方的各个有效路的数目，从而判断当前选择的下法是否为双威胁下法。

3 总结

3.1 遇到的问题及解决方案

3.1.1博弈树超时

解决方案： 我们采用了 Alpha-Beta 剪枝算法来优化博弈树搜索，减少了不必要的搜索分支，从而提高了搜索效率。此外，我们还对搜索顺序进行了优化，通过预处理并排序备选点，使得博弈树搜索的效率得到了显著提升。

3.1.2评价函数复杂度高

解决方案： 我们采用了基于路的评价策略来简化评价函数的计算过程。通过将棋盘上的连子抽象成路，并对每条路进行评分，最终综合所有路的评分来评估棋局优劣。此外，我们采用了全局初始化和局部修改的动态评分方案，以降低评价函数的计算复杂度。

3.1.3TSS 深度浅且无法快速找到目标点

解决方案： 我们优化了 TSS 算法中的威胁空间搜索策略，并结合路表来加速搜索过程。通过路表，我们可以快速获取当前棋局的有效路信息，从而指导 TSS 算法更深入地搜索威胁空间，提高了搜索深度和效率。

3.2 未来改进方向

对于胁迫搜索算法，我们只用了双威胁搜索，未来还可以加入单胁迫搜索，

找到更多的必胜局面。

对于博弈树的选点，为了优化时间，我们但单点评估加点对评估结合的选点方式，但这种方式可能会遗漏最优解，或者最优解找得太晚，未来可以选择一种更佳的筛点方式进行优化。

4 参考文献

[1] Chang-ming X, Ma Z M, Hai-tao M, et al. Generalized TSS and its optimization[C]//2013 25th Chinese Control and Decision Conference (CCDC). IEEE, 2013: 2904-2909.

[2] Xue Y, Li H, Jiang T. Alpha-Beta-TSS in Connect6[C]//The 27th Chinese Control and Decision Conference (2015 CCDC). IEEE, 2015: 3737-3742.

[3] Yen S J, Yang J K, Kao K Y, et al. Bitboard knowledge base system and elegant search architectures for Connect6[J]. Knowledge-based systems, 2012, 34: 43-54.

[4] 张小川，陈光年，张世强，等．六子棋博弈的评估函数[J]．重庆理工大学学报：自然科学，2010(2)：64-68.

[5] 齐祎霏．六子棋中基于路的双评价参数评估函数的研究与应用[D]．北京:北京工业大学,2018.

[6] 李学俊，王小龙，吴蕾，刘慧婷．六子棋中基于局部“路”扫描方式的博弈树生成算法[J]．智能系统学报，2015，10(02)：267-272