**2022年春季学期**

**数据结构课程设计第5-7次课赛道\_B\_**

**实验报告**

罗旺1 唐文鑫1 戴轲翰2 余越3

1软件学院2020级9班

2软件学院2020级10班

3软件学院2020级8班

# 分工与合作

罗旺：搭建MCTS框架，算法优化

唐文鑫：设计价值函数，测试参数

戴轲翰：算法优化

余越：测试参数

# 参考文献与创新之处

参考文献:

1

论文：A Survey of Monte Carlo Tree Search Methods Cameron Browne, Member, IEEE, Edward Powley, Member, IEEE, Daniel Whitehouse, Member, IEEE, Simon Lucas, Senior Member, IEEE, Peter I. Cowling, Member, IEEE, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis and Simon Colto

链接：<https://ieeexplore.ieee.org/abstract/document/6145622>

期刊：IEEE

页码：9（UCT伪代码）

2

论文

创新之处:

为了提高MCTS的模拟效率以提高模拟次数，让MCTS得到的解更逼近最优解，我们在原论文的基础上采取了一系列的优化措施，包括调整UCB公式中的常数C，修改蒙特卡洛树结点的数据结构来用空间换时间，优化原文中的扩展策略，修改模拟策略、设计价值评估函数来准确地评估棋面以提高模拟效率等方法（具体会在算法思想中提到），改善了原文中的MCTS算法，从最开始0.8ms模拟2000次左右到最后0.8ms能够模拟2w次（最后十回合10w到100w次），在botzone中天梯分在1130左右

# 算法思想

在MCTS算法的基础上，对算法的时间复杂度进行优化（尽量用空间换时间），加入价值评估策略

算法大体过程为：

1.由当前局面建立根节点，一次生成根节点的下一层全部子节点，分别进行模拟对局；

2.从根节点开始，进行最佳优先搜索；

3.利用UCB公式计算每个子节点的UCB值，选择最大值的子节点；

4.若此节点不是叶节点，则以此节点作为根节点，重复2；

5.直到遇到叶节点，如果叶节点未曾经被模拟对局过，对这个叶节点模拟对局；否则为这个叶节点随机生成子节点，并进行模拟对局；

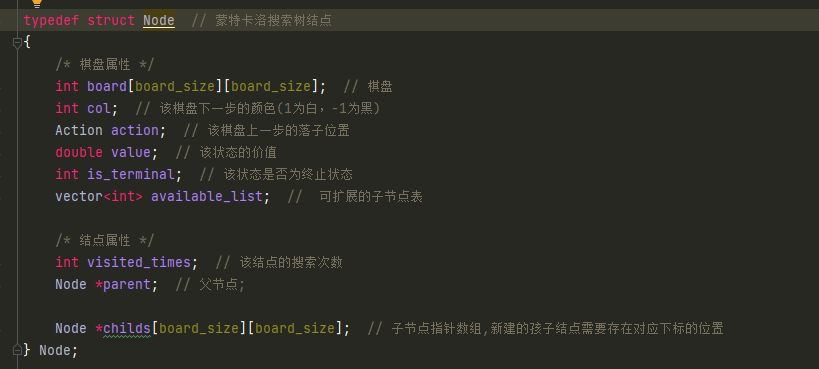
6.将模拟对局的收益(随机落子一定步数后，按一定的评估策略对状态进行评估，返回评出的价值)，按对应颜色更新该节点及各级祖先节点的总价值，同时增加该节点以上所有节点的访问次数；

7.回到2除非此轮搜索时间结束或者达到预设循环次数

8.从当前局面的子节点中挑选胜率（价值/访问次数）最高的给出最佳着法。

算法具体实现为：

1.蒙特卡洛搜索树结点的数据结构



每一个蒙特卡洛搜索树结点都保存着一种棋盘状态，包括父结点指针、当前棋盘的布局、下一子由哪方落下、该棋盘上一步的落子位置、赋予该棋盘状态的价值以及访问次数、标记该棋盘是否为终止状态、下一步的可落子点（可扩展到MCT中的子结点表），最后这两个属性是我们对于基础MCTS的用空间换时间的优化策略，即将重复用到的、耗时的数据给封装到结点中。

算法具体优化思路以及实现：

1. 模拟过程的策略优化（Simulation）

最开始我们的模拟策略就是随机落子到游戏结束，输赢给予模拟开始的状态对应的结点以相应的价值，然后进行反向传播。结果发现这种策略是非常耗时的，其一就是每模拟一步，就要获取到当前棋盘中的可落子位置，而获取这个可落子位置的函数是一个n方复杂度的函数，也就是说总体上这是个复杂度为n的三次方的一个过程。其二就是这种耗时的模拟在没有经过大量次数的模拟的情况下它是没有意义的，因为是随机落子，双方都是“傻瓜”，这样模拟完一把的价值不大，模拟的效率不够，况且还耗时。

针对这种情况，我们最开始想到的是直接用价值函数替代模拟过程，直接用价值函数评估出当前棋盘的价值，这种方式对于价值的函数的时间复杂度要求较高，但是好处是省去了大量的模拟时间，让模拟次数大大增加。但是缺点也很大，其一就是对价值函数的正确性的要求比较大，价值函数是否能够正确评估当前的棋面直接决定了模拟的正确性与效率，而好一点的价值评估函数通常又是费时的，价值评估函数在设计的社会需要衡量时间消耗与正确性这两方面，时间消耗太大，减少了模拟次数，或者正确性不好，模拟的效率不高，都会导致模拟的效率下降，得到的解不够好；另外一点就是，在游戏初期，棋面上没有什么子的时候，这个时候价值评估函数就失效了，导致游戏前期评估出来的价值不够准确，得到的解不够好。

基于以上两点问题的思考，我们最终采取了折中的办法，我们先随机落子几步后再对棋盘进行价值评估，一方面解决了游戏前期的棋面上落子少，价值函数不能很好地评估棋面的问；另一方面又减少了随机落子，让模拟次数增加，然后模拟次数的增加，又能减少价值评估函数的不准确性带来的影响(毕竟MCTS还是基于大量模拟来得到较优解的，只要模拟次数够，什么问题都能解决)。

然后最后就是模拟层数与价值函数的设计。

首先是模拟层数的设计：

模拟层数多了，会导致模拟时间消耗过大，降低模拟次数；模拟层数少了价值函数评估的正确性会降低。

最开始我们设想的是动态的调整模拟层数，经过我们大量的观看对局，发现一般游戏对局在69-71回合结束的比较多，大部分棋局能在73回合结束，而我们固定模拟层数经过大量对比测试后（在botzone上一把一把的下，每一组测试数据都至少测试了50把），发现固定模拟层数为6是最佳。又考虑到，后期棋盘上棋子够，不用过多随机落子，仅凭价值函数就能较好地评估棋面。而前期棋盘上没什么子才需要多随机落子来评估。因此我们最开始设计模拟层数公式为

**模拟层数 = 7 -（回合数 % 10）**

但是经过测试，效果还是不如固定模拟层数为6好，于是更换策略为：

**前30回合模拟层数为6，30回合以后模拟层数为3**

效果依旧不好，最终就确定固定模拟层数为6然后进行价值评估

价值评估函数的设计：

1. 扩展策略的优化（expansion）

最开始按论文中描述的，每选取一个叶节点，就将其一个子结点扩展到蒙特卡洛树中，但我们后面发现，其实每次求解过程中，根节点的下一层结点一定是全在树中的，下两层结点就不一定，所以我们决定一开始就将根节点的下一层结点全部加入到树中，然后进行模拟与反向传播。然后其余的结点就按原策略加入到树中。最终经实验证明，这种策略能够提高百分之10的模拟次数，在游戏后期甚至提高了一倍的模拟次数（因为游戏后期模拟一次的时间很短，添加完一层后就没多少结点能添加了，时间大部分用在模拟上了）

1. UCB公式中的常数C的优化

因为C值决定的是探索比，C值越大，模拟就越倾向于未模拟过或者说模拟次数较少的结点；C值越小，模拟就越倾向于价值大/胜率高的结点。所以对于MCTS中要的要一个好的较优解，C值是很重要的，而且不同游戏的最优C值应该是不同的，于是我们经过大量测试（将C值从1.0到2.0，步长0.1分为12个小数，经过大量测试比较，最后得到最优的C值为1.5，而不是论文中的根号2（1.4）

4.数据结构的空间换时间策略

4.1在选取下一个要扩展的结点的过程中，要重复判断一个结点是否为最终状态(is\_terminal)，一个结点是否是最终状态是不变的，于是将is\_terminal封装到结点的数据结构中，初值为-1表示未计算过是否是最终状态，0表示不是最终状态，1表示是最终状态。计算过一个点是否为最终状态后就将其值保存到is\_terminal属性中，后面再访问直接访问该属性即可，于是函数设计就是：



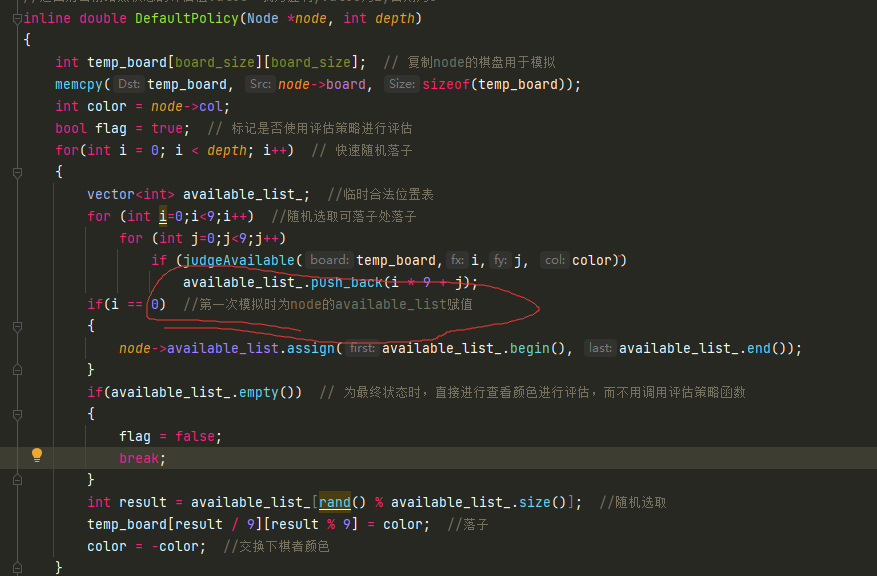
4.2在选取结点进行扩展的时候，随机选取能够进行扩展的一个子结点加入树总，即每次都要计算这个点的哪些子节点能够加入到树中，然后随机选取一个。因为每次只加入一个子节点进入树中，其余的子节点后续可能也会加入树中，也会计算这个点的哪些子节点能够加入到树中。

基于这个问题，我们将



封装到结点中，每次扩展时即为其计算可扩展的子结点表，需要扩展的时候就从表中随机选一个扩展，然后每扩展了一个子节点，就从对应的表中删去。

然后这里还有一个优化点，就是什么时候为结点计算可扩展的子节点表，我们发现，我们在扩展一个结点后，需要对结点进行模拟，而模拟的第一步随机落子，就会计算这个结点的下一个可落子点，即可扩展的子结点表，于是直接将可扩展的子节点表的初始化放入模拟中



5.其他优化：所有结点加入树中后，不断对根节点的下一层子结点进行模拟

以上就是我们组的不围棋MCTS算法的全部优化过程以及实现