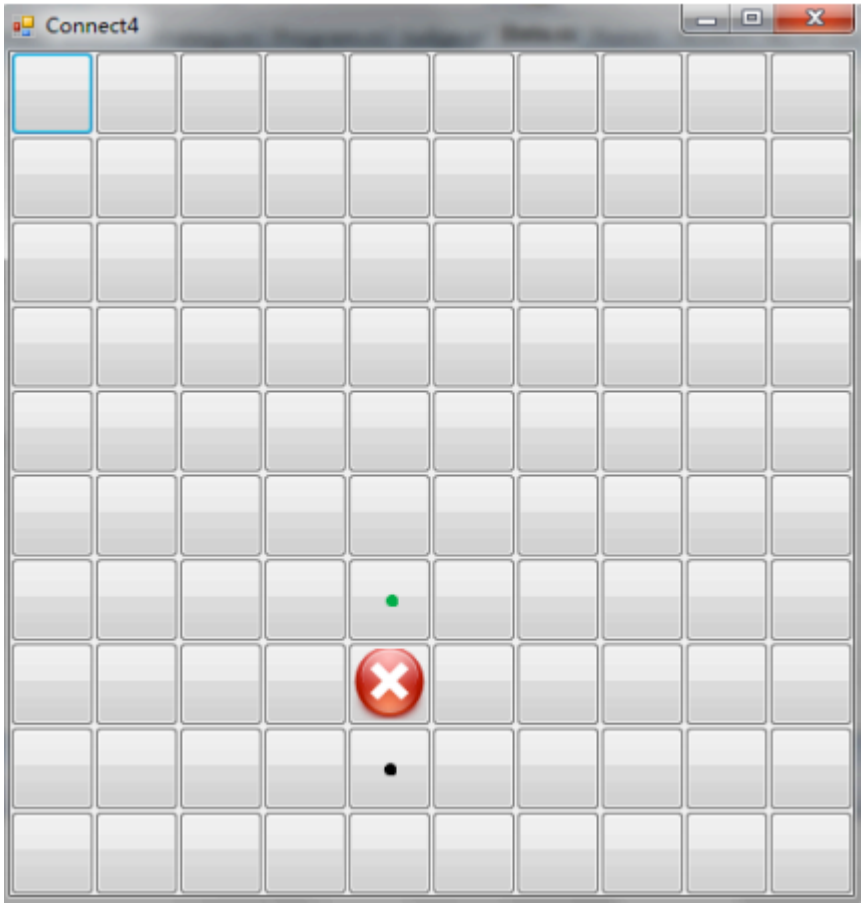
四子棋 AI 设计实验报告

一、 游戏简介

四子棋游戏是一个非常经典的游戏，游戏双方分别持不同颜色的棋子， 设 A 持白子，B 持黑子，以某一方为先手依次落子。 假设为 A 为先手， 落子规则如下： 在 M 行 N 列的棋盘中， 棋手每次只能在每一列当前的最底部落子

如图中的红点处所示， 如果某一列已经落满，则不能在该列中落子。棋手的目标是在横向、纵向、两个斜向共四个方向中的任意一个方向上， 使自己的棋子连成四个(或四个以上)， 并阻止对方达到同样的企图。先形成四连子的一方获胜，如果直到棋盘落满双方都没能达到目标，则为平局。首先， 当两个策略进行对弈时， 棋盘的大小是随机

的，而不是固定不变的(宽度和高度的范围均为[9,12])。 由于并不是所有规模的棋盘都有必胜策略，也不是所有的必胜策略都是先手必胜，所以这里带来了不确定因素。其次，每次棋盘生成之后，会同时在棋盘上随机生成一个不可以落子的位置， 如图中的



红叉所示。 当某一次落子是在黑点处时， 该列下一次可落子的位置就变为了绿点处，而不再是黑点上面的位置。

二、 实验简介

写一个强大的 AI 对抗现在已有的 100 个已按智能排好序的测试 AI，要求写成的 AI 不能出 bug，不能违反游戏规则，推荐使用 α-β 剪枝算法。

三、 基本思路与方法

本次大作业实验，我实现了 α-β 剪枝和蒙特卡洛树两种算法，最后提交蒙特卡洛树算法。

(一) α-β 剪枝算法

开始，按照实验推荐，写了一个 α-β 剪枝算法，但是效果不是很好，瓶颈出现在搜索深度与局面的评估。因为时间限制不能进行大量的搜索，搜索深度只有定在 6 层，而 6 层的深度远远不能够打败 50 以后的测试 AI，甚至 50 以前的部分测试 AI 也不能有 70%以上的胜率。搜索深度的瓶颈解决不了，就从局面估价入手进行优化，希望能找到一个非常棒的估价方式。

我先后设计了以下 4 个估价函数：

1. 考虑连续落两子胜利的方案数。返回第一子落子后对方无法必胜，第二子落子后我方胜利的方案数。对于第一子落子后，对方无法必胜，而第二子的落子方案有 2 种以上，此时我方必胜，给予一个极大评估值，否则只有一种落子方案必胜，返回值+1。
2. 用蒙特卡洛算法随机模拟，我方胜利次数-对方胜利次数作为评估值。
3. 评估对方实力与我方实力差。以对方上一次落子为中心的边长为 5 的正方形范围内，考虑对方棋子与我方棋子之差。对于对方前 4 步之内的棋子，一个算两个。即对方棋子太过紧密则认为对我方不利。

4. 预测对方棋子的连通块。以对方上一次落子为中心的边长为5的正方形范围内，考虑对方棋子的连通块的个数，以游戏规则的相连方式判读两个块是否连通。如果对方的连通块个数太少则认为对方的落子都是相互联系，布局谨慎，危险系数大。

一开始，我只设计了第一个评估函数，效果不是很好，最后我同时用 3 个评估函数(蒙特卡洛模拟评估效果不好没有采用)，以 a * f(1) + b * g(3) + c * h(4) 评估当前棋局，这样效果比以前好很多，胜率提高不少，但是仍然对于后面的测试 AI，胜率还是很低，对于 90 以上几乎为 0。

后面看了一个同学的 α-β 剪枝写得很厉害，请教以后他告诉我他的搜索深度能达到 14 层。原来他采用了迭代加深，对于估价较好的局面多搜，这样就能加大剪枝力度，并且他的估价函数也非常好，能支持加入一个点后，O(1)更新估价函数值，效果非常好。

(二) 蒙特卡洛树

最后 google 了一下关于 AI 算法的资料，发现了一个很优秀的算法——蒙特卡洛树。看完相关文献资料以后，开始写这个算法，但是发现写出来效果不太好，最后和请教同学发现有个地方理解错了。每次选择最大 UCB 值时，要从当前棋手考虑，而不是我方考虑。修改以后，效果明显比 α -β 剪枝好很多。

蒙特卡洛树这个算法有很多值得研究和思考的地方。比如：1

收益如何计算，2、参数的选取，3 如何选取落子位置。

收益如何计算？在这里我被坑了不少时间，开始我对于当前棋手来说，如果赢了则+1，输了则+0，这样计算收益是不优秀的，它不能很好的体现性能；最后我 google 一下，发现赢了+1，输了-1 这样的收益要更好一点。

参数如何选取？不管在任何地方，选择很好的参数都是一个有挑战的问题。比如之前学过的模拟退火和遗传算法。在蒙特卡洛树中，对于信心参数的选取也是需要值得思考的。经过多次对战，打印输出树的深度，发现参数最好是一开始比较大，后来比较小，这样可以在一开始更多的拓展较多的节点，使树变的更大更深，后面参数变小，就更多的是凭借收益率来选取最优儿子，这样可能会对收敛有一定的积极效果。但是这样的参数并不好选取。我设置了一个可变动的参数，每迭代一次就让这个参数减去一个变化值，在多次迭代下，它可以衰弱得很小。这样以来效果确实比一个固定了的参数要好得多。当我拿程序在同学电脑上跑的时候发现又一个问题出现了，不同机器下的速度不一样，迭代次数不一样，则衰减速度应该不一样，而且可变参数初始值也应该不一样。因为我不会如何从具体的机器上学习一个比较靠谱的参数，所以只有考虑按时间来衰减，因为时间是一样的。可是时间的精度不够高，而且不容易控制，所以可变参数这个想法我也就放弃了。最后我还是选择了一个固定的参数 0.9，效果虽然没有可变参数好，但是对于不同机器的平均效果已经不错了。

关于落子位置的选取。根节点有很多个儿子作为候选者对应每个落子位置，那么应该选取哪一个呢？文献中选取的还是 UCB 最大的值，但是我看对战过程，打印出比赛次数，收益，收益率，UCB 值，发现有时候选取 UCB 值并不是最优的。某个节点的收益率可能很小，但是因为比赛次数太少，在后面的根号调整后，它的 UCB 变得更大。显然这样的节点肯定是不靠谱的。那么选取收益率最大的吗？在有些对战中，选取收益率最大也不是靠谱的。因为有可能某个节点，它的比赛次数很少，但是它的收益率率很大。比如它只比赛了 10 次，收益为 6，但是另一个节点比赛 100 次，收益为 58。这个时候肯定不能选择 10:6 的节点，因为它比赛次数太少，所以可信度不高，也就是说我们应该在可信度高的节点中选取 UCB 最大的，这个时候前两个问题都得到了解决。可信度怎么判断呢？只有当某个节点的比赛次数不小于比赛次数平均值才认为它可信，否则它不可信也就不被视为候选者。

优化思考：

这个算法还是有很多地方值得思考的，比如总选 UCB 最大的节点扩展不一定最优，UCB 次大的说不定结果更优一点，这个时候我加了一点随机，以很小的概率随机选择一个节点扩展。还有蒙特卡洛对战的时候，对于某一方可能已有必胜，但是随机落子不一定落子正确，这个时候错过机会，而另一方可能会获胜。也就是说蒙特卡洛打法的获胜可信度也不高，可能导致必胜的还输了。如果在每一步加一个判断是否有必胜，那时间复杂度又乘了一个 10 的常数，

这样会导致迭代次数降低，也不利于算法的效果。对于一个节点，它的子节点中出现了必胜的节点，则可以把其他的子节点剪枝掉以防出现不可信的胜局。

四、 总结

有很多同学都写了蒙特卡洛树，网上也有很多蒙特卡洛树的资料，但是这个算法的效果还真是无法预料。在我多次尝试和调整下，蒙特卡洛树还是达不到对战 94 号以后的 AI 胜率很大的效果，但是听说有同学的蒙特卡洛树能随便赢 100 号，而且也达不到网上同学描述的那么好的效果。这也从侧面突出这个算法有很多值得思考的地方，它的发展潜力还是很大。

完成此次大作业实验的时候，和同学交流也收获了不少，自己也在过程中学了一些新算法，思考了更多的东西，比如如何评估棋局，如何调参，如何随机优化等。虽然自己的 AI 不是很厉害，但是已经是自己努力后的最好成果了。自己以后也会思考有没有更好的方式优化自己的 AI，在思考中取得更大的进步。

五、 测试结果

测试结果有两份，分别为在两台不同的计算机上与每个 AI 对战 3 轮的结果。总得来说对于前 94 的 AI，胜率都在 80%以上，对于 96 以后的 AI，胜率偏小。详情请见 test 文件夹。