

六子棋对弈机器人研究报告

**2021 年 春 季学期**

姓名：马俊腾 学号：18020031058

姓名：邓宇 学号：14020031013

姓名：刘成健 学号：18020031043

姓名：郭天齐 学号：18020031025

姓名：郭冉 学号：18020032005

目录

[1.六子棋游戏 1](#_Toc1222881554)

[1.1六子棋游戏描述 1](#_Toc1503135117)

[1.2六子棋游戏的公平性 1](#_Toc1877848513)

[2.六子棋对弈算法实现 1](#_Toc1842699518)

[2.1 路的定义 1](#_Toc822267779)

[2.2 路表维护 2](#_Toc142186536)

[2.2.1数据结构 2](#_Toc491779603)

[2.2.2初始化 2](#_Toc1368849571)

[2.2.3动态更新 3](#_Toc1774117020)

[2.3局面评估 5](#_Toc202663195)

[2.3.1对于棋局的评估 5](#_Toc2018488188)

[2.3.2下棋过程中对可以落子的位置的评估 6](#_Toc1478864778)

[2.4关键数据结构及其对算法性能的影响 7](#_Toc2145503698)

[3. 搜索算法 7](#_Toc1670486368)

[3.1α-β剪枝 7](#_Toc1026341005)

[3.1.1α-β剪枝原理 7](#_Toc476845895)

[3.1.2α-β剪枝算法实现 9](#_Toc1984247085)

[3.2威胁空间搜索（TSS） 10](#_Toc476773110)

[3.2.1威胁空间搜索算法原理 10](#_Toc1652572383)

[3.2.2威胁空间搜索算法实现 10](#_Toc1775271900)

[4. 两种搜索算法的比较 12](#_Toc2000061319)

[5.总结与展望 12](#_Toc1821698027)

[6.参考文献 13](#_Toc270811852)

# **[1.六子棋游戏](#_Toc39796669)**

## **[1.1六子棋游戏描述](#_Toc39796670)**

六子棋的规则与五子棋非常相似，玩家有黑白两方，各持黑子与白子，黑方先行。采用19\*19的棋盘。具体玩法：除了第一次黑方下一子外，之后白黑双方轮流每次各下两子。直的、横的、斜的连成6子（或以上）者获胜。若全部棋盘填满仍未分出胜负，则为和局。

## **[1.2六子棋游戏的公平性](#_Toc39796671)**

传统的五子棋游戏，很显然是不公平的，每当先手的黑方下出一子，盘面总是比白方多一子，而当后手白方下一字，盘面棋子却只能跟黑方持平，且现在已经有计算机专家证明出五子棋先手必胜的结论。而对于六子棋，每当一方下子结束，盘面上总是比对方多一子，所以对于传统的五子棋而言，六子棋更加公平，对于六子棋公平性的严密证明，至今还没有人能给出，但是就目前的研究成果来说，六子棋的公平性依然是极佳的。

# **[2.六子棋对弈算法实现](#_Toc39796681)**

## **2.1 路的定义**

六子棋中每一种棋形都有很多的表现形式。同时如果有几种棋形交叉时判断起来就更加复杂。如何判断棋形就成为难点,如果模板设置错误或棋形统计不完全，将严重的影响博弈系统的棋力。因此需要引入“路”的概念。所谓“路” 就是在棋盘上连续6个点能够连成一条直线,则称为1“路”。路的起点一般以这条直线上坐标最小的点。路的方向有横向、纵向、右上斜、右下斜4个方向，横向有19 × （19 - 6 + 1） = 266路，同理，纵向也是266路；右上斜有14 × （19 - 6 + 1） = 196路，同理，右下斜也是196路，因此，棋盘上共有924条路。

由于每条“路”上有6个连续点位，这样对棋型的判断就变得非常简单，不必判断它属于哪种棋型，只需要给定一条路上的己方棋子的个数，比如是4，并且对手棋子的数量为0，那么就可以判定有杀棋。

代码中关于路的定义的变量如图 1 与路相关的成员变量所示。

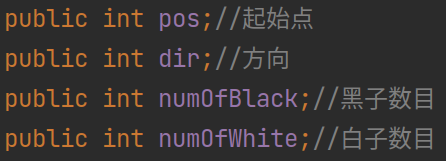


图 1 与路相关的成员变量

## **2.2 路表维护**

### 2.2.1数据结构

数据结构采用动态二维数组，第一维是路上黑棋的数量，第二维是路上白棋的数量。关键定义如图 1 路表类中的数据结构所示。



图 2 路表类中的数据结构

### 2.2.2初始化

路表中的数据结构初始化代码如图 2 动态二维数组的初始化所示。其中的每个元素roads[i][j]都是一个动态数组，存放黑棋数量为i，白棋数量为j的所有路。

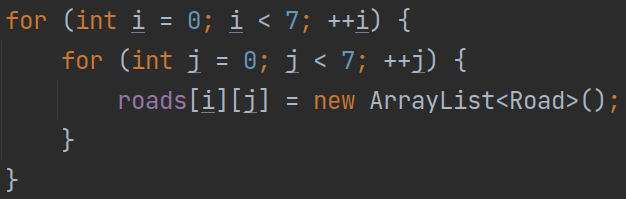


图 3 动态二维数组的初始化

初始化所有的路的方法是是遍历棋盘上的所有点，对于某个点A，根据路的定义，从4个方向，大小为6，分别得到以A为起点的所有路。关键代码如图 1 求以start的为起点的所有路的代码所示。如果某个方向上的路合法，即不会越界，那么就求得一条路。为了方便求解，这里暂时采用了二维棋盘，其余地方为一维棋盘。

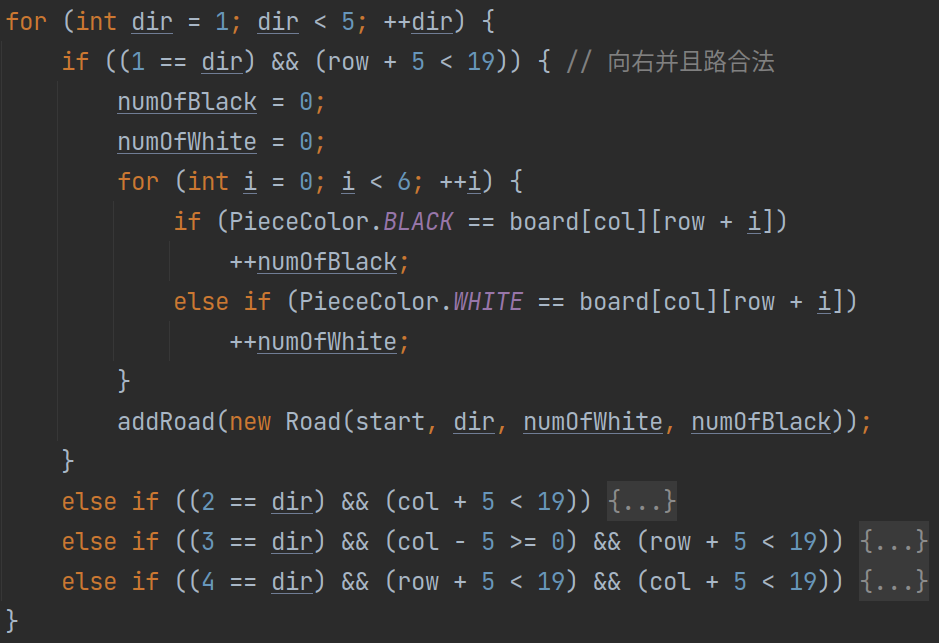


图 4 求以start的为起点的所有路的代码

### 2.2.3动态更新

当着子或者取消着子时，会影响包含这个棋子的所有路的棋子的数量，所以需要动态更新路表。关键代码如图 2 动态更新路表所示，当给定一个棋子A，方法就是找到包含A的所有路。对于任意一条路，首先删除对这条路的引用。在更新这条路上的棋子数量后，形成了一条“新”路，然后增加对“新”路的引用。在这里路的对象没有删除，删除的是引用。



图 5 动态更新路表

找到包含坐标pos的所有路的代码如图 5 找到包含坐标pos的所有路所示，方法是遍历所有的路，如果某条路包含这个这个点，就加入到ans中。

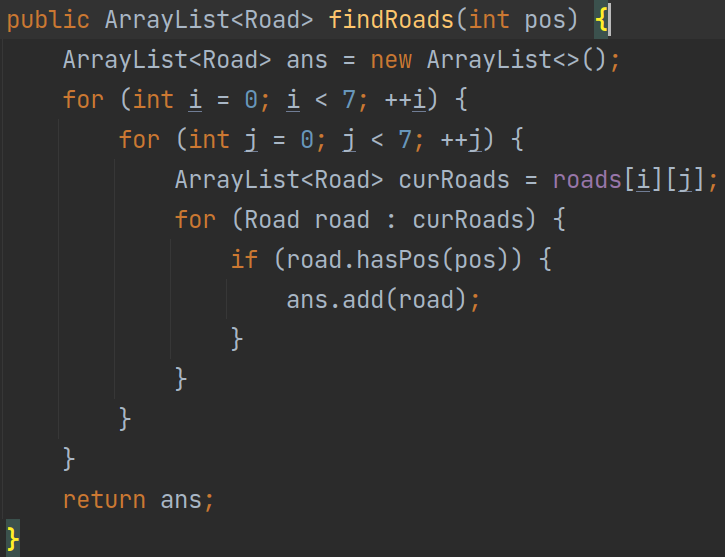


图 6 找到包含坐标pos的所有路

## **2.3局面评估**

### 2.3.1对于棋局的评估

采用的方法是遍历棋盘上的所有路，对每条路的估值求和。主体代码如图 7 评估棋局所示。

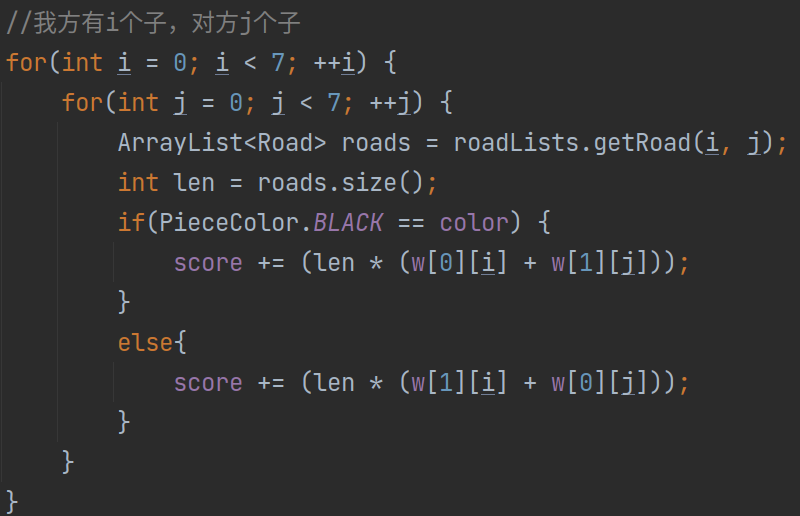


图 7 评估棋局

其中的w数组如图 8 w数组所示。当有1个黑色棋子的值为1，此后每多一个黑色棋子，便乘以925。思路是因为在一条线上的两个棋子要比不在一个线上的两个棋子分更高，因为有924条路。而2个棋子在最夸张情况下也不会总共被924条路拥有，所以我乘以925可以实现两个棋子在同一条线比不在同一条线好，再添加黑子以此类推。而当路上出现一个白色棋子，就设置的同样数量的白色棋子比同样数量的黑色棋子要有更大的负面估值，而且白色棋子在同一条线上的负面估值要比不在同一条线上严重。

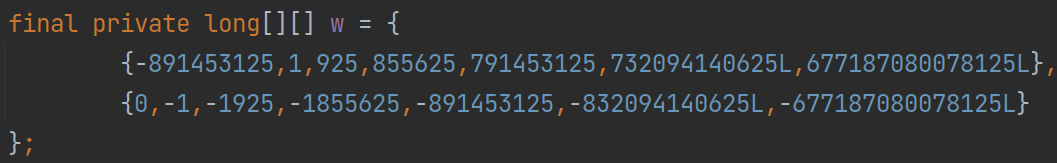


图 8 w数组

### 2.3.2下棋过程中对可以落子的位置的评估

并非所有空白位置的棋子都对局面有积极意义，而且，如果把所有空白位置的棋子都考虑进来，在搜索过程中很耗时间，有用的棋子一定只包含在非空并且黑棋数量或者白棋数量为0的路中，只有这样才能赢或者防止对手赢。所以先把包含这些棋子的路加进来（分为只包含黑棋的路和只包含白棋的路），然后对这些路根据拥有棋子的数量评分，然后从大到小排序。再从这些路中找到着法，使得着法的数量下降，加快了AlphaBeta剪枝的搜索速度。

这里用到的评估是棋子越多，估值越高，因为棋子越多对对手或对自己的威胁越大。具体代码在下面的AlphaBeta中会体现。

## **2.4关键数据结构及其对算法性能的影响**

这里面的关键数据结构就是上面提到的路表roads。它本质上是个二维动态数组，第一维代表的是路上黑棋的路上，第二维表示的是路上白棋的数量，然后每个元素roads[i][j]保存的是具有i个黑棋和j个白棋的所有路的集合。

有了这个数据结构之后，要得到只有黑棋或只有白棋的路就十分方便，比如要得到具有4个黑棋而且没有白棋的所有路，roads[4][0]，使得威胁空间搜索和AlphaBeta搜索能够更快，也更简单实现。

# **搜索算法**

搜索算法是先用威胁空间搜索算法找己方有没有杀棋或者对手有没有杀棋，如果没有的话再调用AlphaBeta剪枝搜索。

## **3.1α-β剪枝**

### **3.1.1α-β剪枝原理**

**博弈树：**

博奔过程中，双方都希望自己能够获胜。因此，当某一方当前有多个行动方案可供选择时，他总是挑选对自己最为有利，而对对方最为不利的那个行动方案。设博弈双方中一方为MAX,另一方为MIN。如果我们站在MAX方的立场上，则可供MAX方选择的若干行动方案之间是"或"关系，因为主动权操在MAX方手里，他或者选择这个行动方案，或者选择另-个行动案，完全由MAX方自已决定。当MAX方选.取任一方案走了一步后，MIN方也有若千个可供选择的行动方案，此时这些行动方案对MAX方来说它们之间则是"与"关系，因为这时主动权操在MIN方手里，这些可供选择的行动方案中的任何一个都可能被MIN方选中，MAX方必须应付对自己最不利的情况的发生。将上述博弈过程用图的形式表示出来，则得到一棵博弈树。

将走棋方定为MAX方，因为它选择着法时总是对其子节点的评估值取极大值，即选择对自己最为有利的着法；将应对方定为MIN方，因为它走棋时需要对其子节点的评估值取极小值，即选择对走棋方最为不利的、最有钳制作用的着法。

**α剪枝：**

在对博弈树采取深度优先的搜索策略时，从左路分枝的叶节点倒推得到某一层MAX节点的值，可表示到此为止得以“落实”的着法最佳值，记为α。

显然此值可作为MAX方着法指标的下界。

在搜索此MAX节点的其它子节点，即探讨另一着法时，如果发现一个回合（2步棋）之后评估值变差，即某孙节点评估值低于下界α值，则便可以剪掉以该子节点为根的子树，即不再考虑存在“软着”子树的延伸。

**β剪枝：**

同理，由左路分枝的叶节点倒推得到某一层MIN节点的值，可表示到此为止对方着法的钳制值，记为β。

显然此β值可作为MAX方无法实现着法指标的上界。

在搜索该MIN节点的其它子节点，即探讨另外着法时，如果发现一个回合之后钳制局面减弱，即孙节点评估值高于上界β值，则便可以剪掉此枝，即不再考虑存在此“软着”子树的延伸。

### **3.1.2α-β剪枝算法实现**

如图 9 达到最大深度返回所示，当搜索达到最大深度时，计算叶子结点的得分并返回。

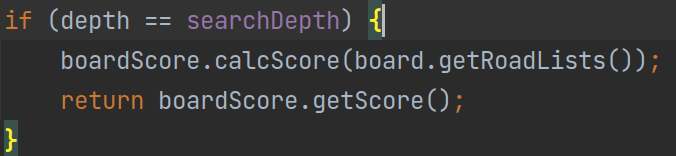


图 9 达到最大深度返回

如图 10 枚举可行落子，继续AlphaBeta剪枝所示，枚举所有预处理得到的可行落子，继续AlphaBeta剪枝。

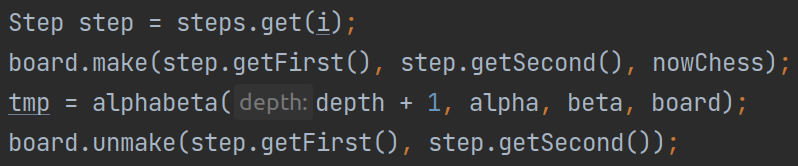


图 10 枚举可行落子，继续AlphaBeta剪枝

如图 11 剪枝所示，剪枝。如果子结点返回的值小于alpha，就剪枝，如果子结点返回的值大于beta，也剪枝。

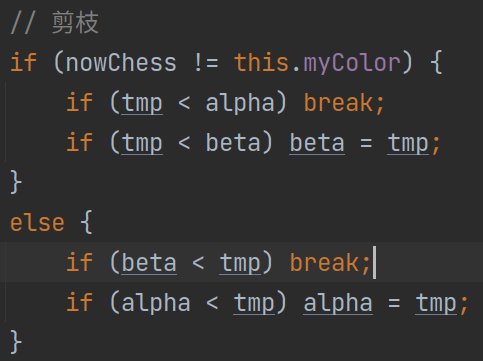


图 11 剪枝

其中，预处理得到估值最高的落子。找己方落子如图 12 找有用的己方棋所示，对于每条己方棋子不为0但是对手棋子为0的点都执行这个操作，把点加进killPos中，加前6个最优的，因为路是从最优到最劣排序的。找对手落子和下面方法类似，不过这里的对手落子是我们要下的要去堵住它的位置。

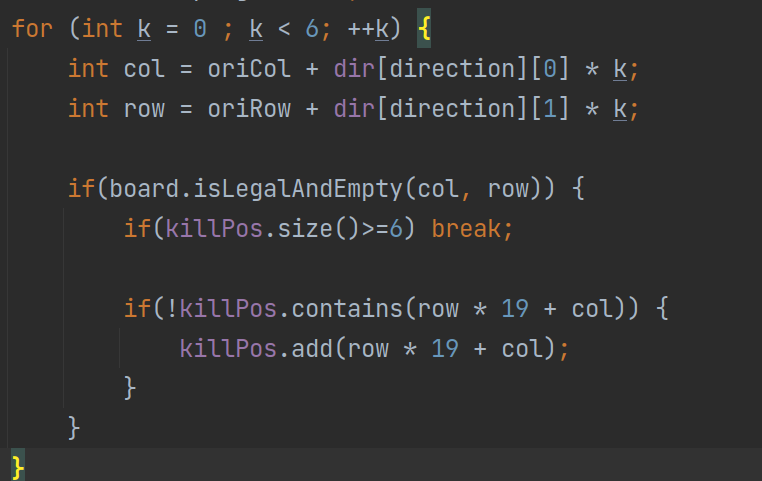


图 12 找有用的己方棋

然后根据堵住对方两个子、堵住对方一条路、或者给己方加子，得到这些选出来的落子的组合。这些组合作为最终的AlphaBeta剪枝的可供选着法。

其中，堵住对方两个子的代码如下：

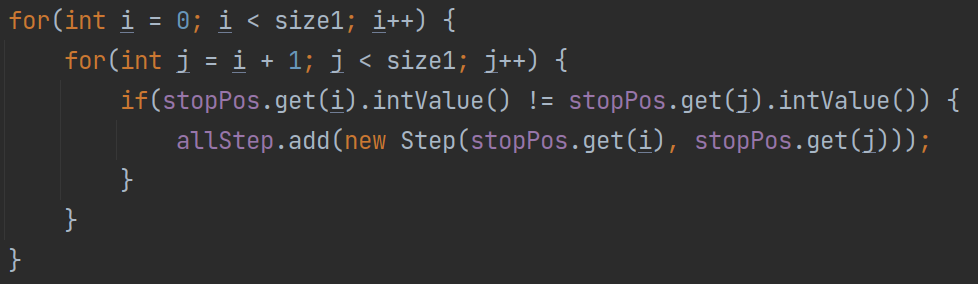


图 13 堵住对手两条路

## 3.2威胁空间搜索（TSS）

### 3.2.1威胁空间搜索算法原理

在二人博弈中,一方输棋经常因为其无论选用何种着法,都无法阻止另一方取胜所致。在赢棋或输棋之前,一方总是给另一方造成一种压迫或威胁的态势,也就是说,一方可以依赖于某种连续威胁的手段一气呵成地获胜，这种手段称之为“迫着”。正是因为这种特点,以寻找直接获胜的策略为目标时,一些不具备形成威胁的着法便可以不去考虑,只要考虑那些明显给对手造成威胁的着法就够了,这便简化了问题搜索的空间。

六子棋中广泛地存在这种迫着,于是很容易就出现某一方可通过连续迫着获胜的着法序列。TSS的目的即是优先考虑多迫着形成局部优势或者直接胜利。

### 3.2.2威胁空间搜索算法实现

找己方有没有杀棋，如图 12 加入含有杀棋的路所示，加入含有杀棋的路，即己方棋子有4个或5个，而对手棋子数为0的路。

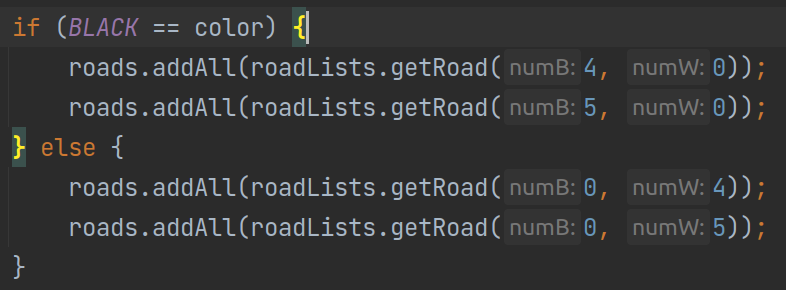


图 14 加入含有杀棋的路

然后遍历每一条路，对于每一条路，从起点开始，沿着它的方向遍历，看哪些位置是空的，就加入到step中，如果step的大小为1，那么剩下的点随便找个空位置下。

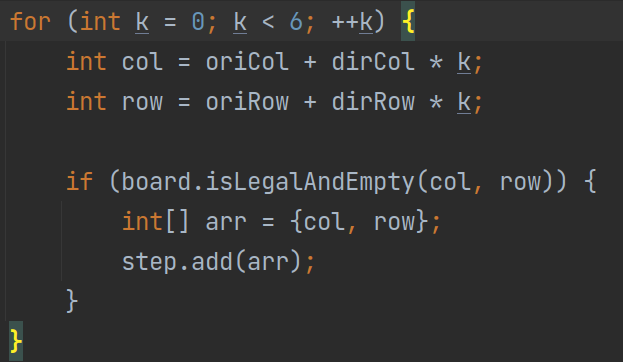


图 15 沿路的方向搜索六步

当找到后，就退出，返回step，如果没有找到的话，就返回null，如图 14 遍历所有的路返回解所示。

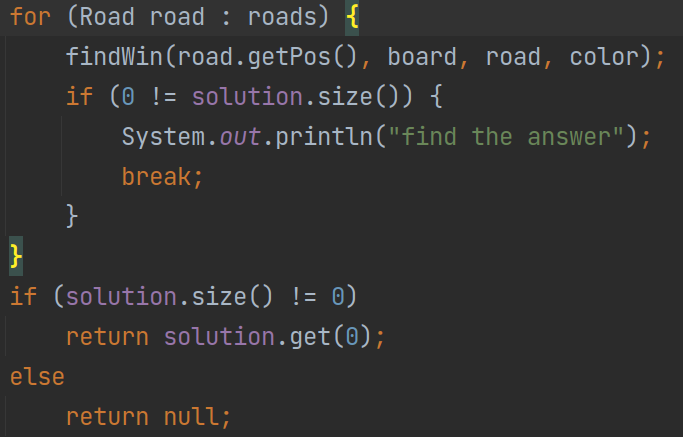


图 16 遍历所有的路返回解

类似的方法找对手有没有杀棋，有4个棋子或者5个棋子必须要堵，对于双三的情况，如果不堵的话再下一步也会形成杀棋，所以也要堵住。所以对于双三或者含4子或5子的堵法就是先堵住4子或5子的路，即需要找到两步棋它们被包含在所有的这些含4子或5子的路中，如果含4子或5子的路的数量为0，再判断双三，堵法一样。代码如图 15 堵住所有的有杀棋的路所示。

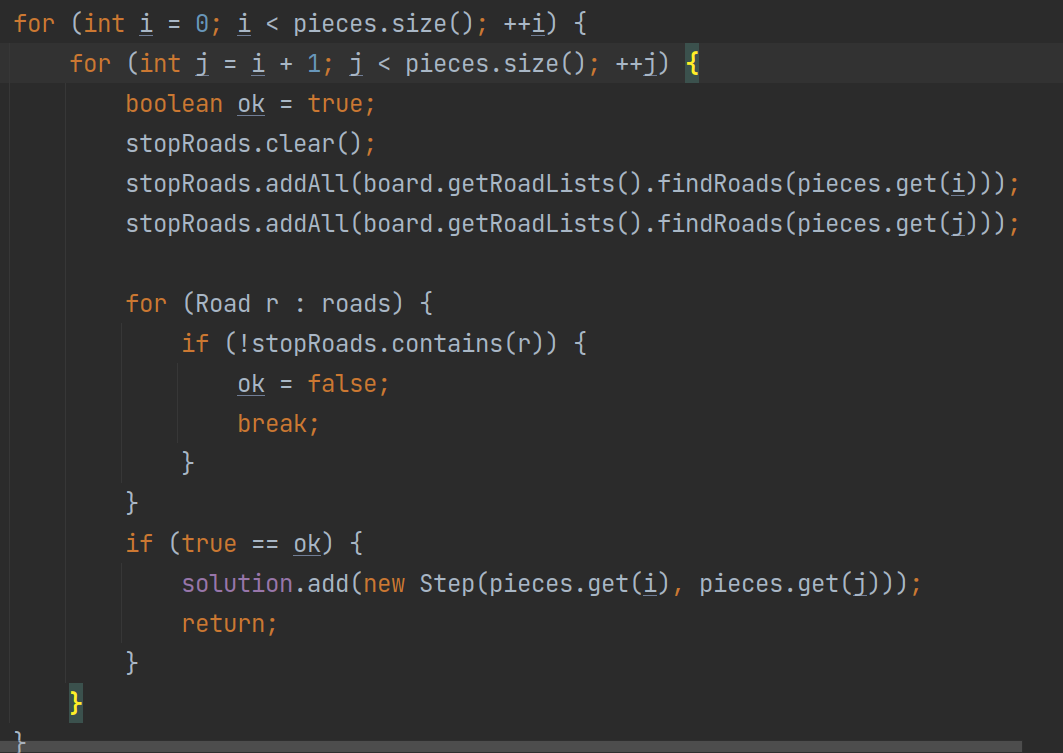


图 17 堵住所有的有杀棋的路

# 两种搜索算法的比较

威胁空间搜索是在找杀棋，而AlphaBeta剪枝是在根据后面的最优棋局来决定当前该走哪步。AlphaBeta搜索往往深度较浅，深了之后速度太慢，所以往往不能得到当前局面的实际博弈理论值。而威胁空间搜索每次找的是当前最优解，不一定是全局最优解，所以二者相互配合使用最好。

我们的搜索引擎是先用威胁空间搜索找有没有杀棋，没有的话再用AlphaBeta剪枝找当前最佳着法。

# **5.总结与展望**

经过这次的项目，让我对六子棋有了一定的认识，了解了路这一概念，它使得棋盘表示变得更简单，而我们也总是关注于只有白棋或只有黑棋的路。我对于AlphaBeta剪枝和威胁空间搜索也有了一定的认识。AlphaBeta剪枝能够剪掉很多不需要搜索的子结点，而威胁空间搜索以目标为导向，要找到杀棋，或防住对方的杀棋，使得能够更快找到赢棋的决策。其实AlphaBeta剪枝和威胁空间搜索在人下棋的思维中也是存在的。尝试一步棋怎么走最好，和AlphaBeta剪枝有关，尝试怎么下最终能赢和找杀棋，和威胁空间搜索有关，二者都是互补的算法。

这次很碰到了很多bug和小问题。最经典的是把行和列有的替换过来了，有的没有替换过来，最后发现有个地方行和列没有表示好。再就是最初删除路的引用的时候，一条return语句放错了位置，结果导致没有成功删掉对路的引用，就到了有时候同一条路会出现两次，在更新黑棋和白棋的数量时就会出错。还是比较难解决，因为一搜索数据量就有点大，就难找，对于输出的数据用vim和grep处理了下，锻炼了这方面处理数据的能力。

# **6.参考文献**

[1]徐心和, 邓志立, 王骄,等. 机器博弈研究面临的各种挑战[J]. 智能系统学报, 2008, 3(04):287-293.

[2]徐长明. 基于连珠模式的六子棋机器博弈关键技术研究[D]. 东北大学.

[3]李果. 基于遗传算法的六子棋博弈评估函数参数优化[J]. 西南大学学报(自然科学版), 2007, 29(11):138-142.

[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, 2010, 000(002):P.64-68.

[5]闵文杰. 六子棋计算机博弈关键技术研究[D]. 重庆交通大学.