使用极大极小搜索和αβ剪枝及遗传算法的五子棋AI

一 问题分析

博弈树在人工智能的应用相当重要，若要寻找某赛局中最佳的步法的一个方式，是利用极小化极大演算法在树中搜寻最佳解，例如在井字游戏中电脑可以很快速地找到最佳解并做出决策

二 算法设计

一、极大极小搜索（Minimax Algorithm）

在零和博弈（有完整信息的，确定的、轮流行动的，两个参与者收益之和为0的博弈）中，双方都希望自己获胜，因此每一步都选择对自己最有利，对对方最不利的做法。

假设我们是参与博弈的一方。我们用静态估计函数f ( p ) f(p)f(p)来估计博弈双方的态势：

有利于我方的态势：f ( p ) > 0 f(p)>0f(p)>0

有利于敌方的态势：f ( p ) < 0 f(p)<0f(p)<0

双方均衡的态势：f ( p ) = 0 f(p)=0f(p)=0

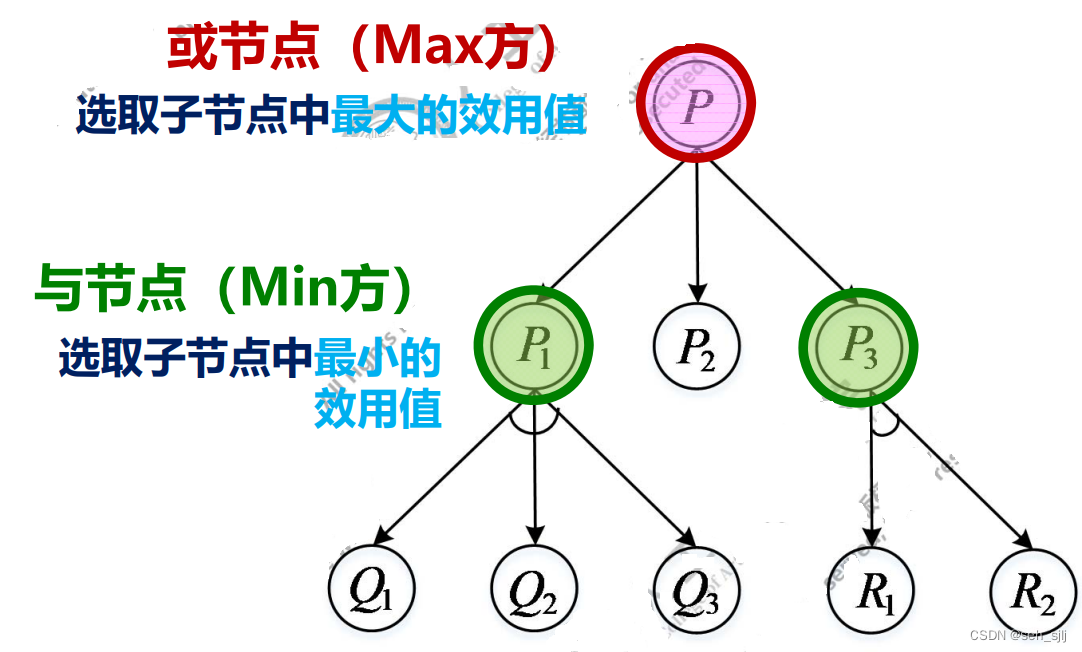
显然，我方希望f ( p ) f(p)f(p)最大化，敌方希望f ( p ) f(p)f(p)最小化。因此称我方为Max方，敌方为Min方。

在Max方的角度，因为是我们自己做决策，我们可以选择任意一种方案，所以我们只需选择收益最大的方案，也就是说每种方案之间是“或”的关系。

而对于Min方而言，因为是敌方做决策，我们无法控制敌方选择哪种策略，假设敌方足够聪明，我们应该假设敌方选择对他最有利的方案，也就是对我们最不利的方案、使我们收益最小的方案，所以对他而言每种方案之间是“与”的关系。

假设我们在进行动态博弈——你一步，我一步，且一方做完决策之后另一方知晓他所做的决策，那么我们可以把双方的行动展开成一棵树——博弈树。

在博弈树中，每个节点代表一种格局，每条边代表Max方或Min方的一步操作。那些下一步该Max方走的节点称为Max节点，下一步该Min方走的节点称为Min节点。



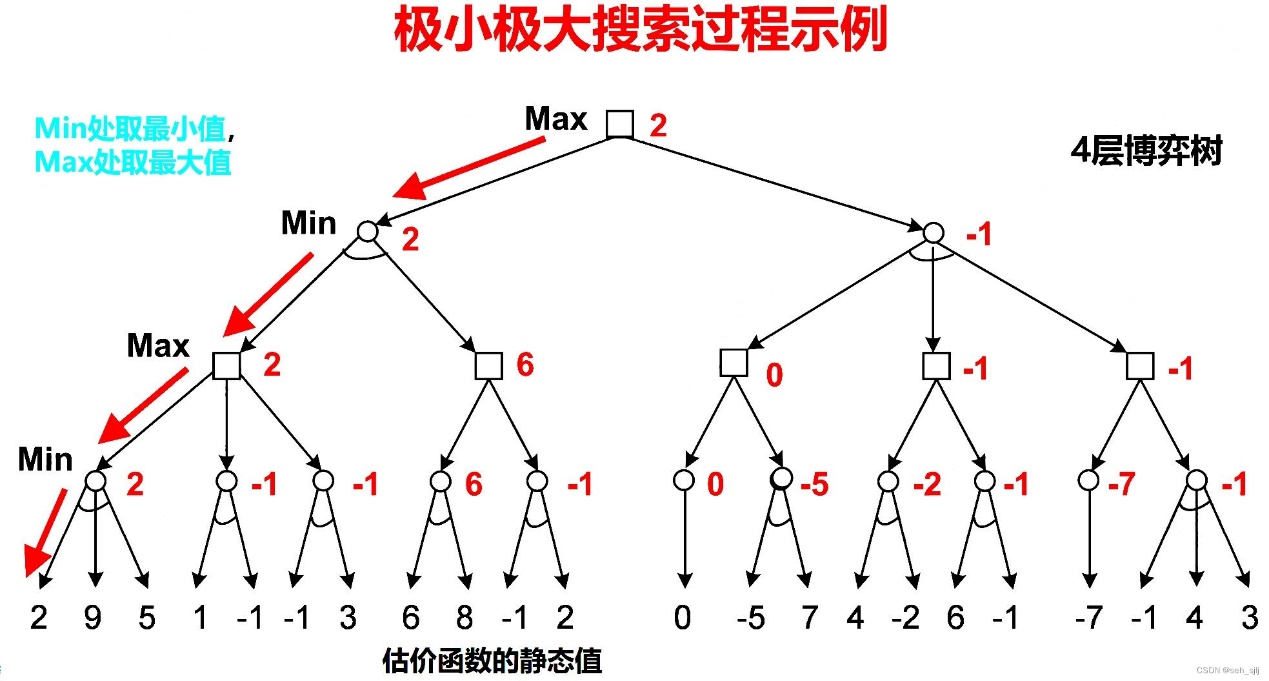
由此我们可以归纳出极小极大搜索算法（Minimax Algorithm）的一般步骤：

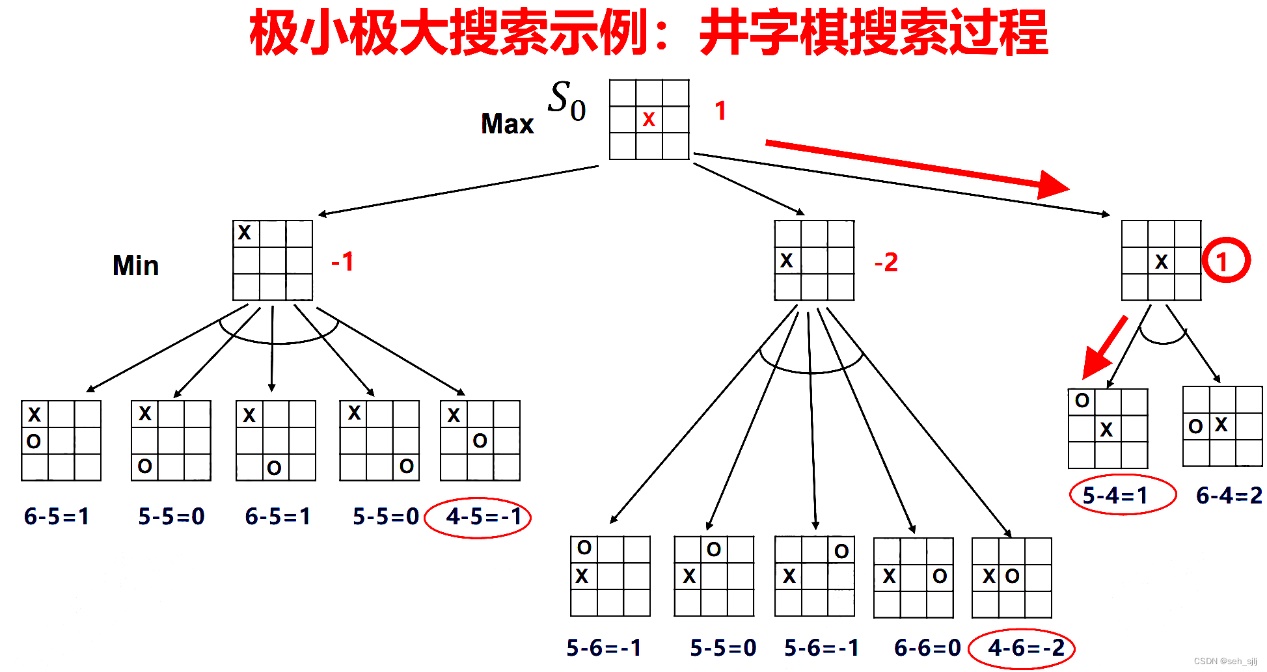
(1) 利用深度优先搜索算法生成Max方当前状态下可猜测的k步博弈树；

(2) 定义静态估计函数，计算端节点的效用值；

(3) 回溯评估：利用极大极小运算自下而上逐层推出各节点的效用值，其中在Max节点取最大值，在Min节点取最小值；

(4) 根据当前状态子节点的效用值做出最优决策，状态转移到子节点的状态，对方变为Max方，回到(1)开始新的搜索。



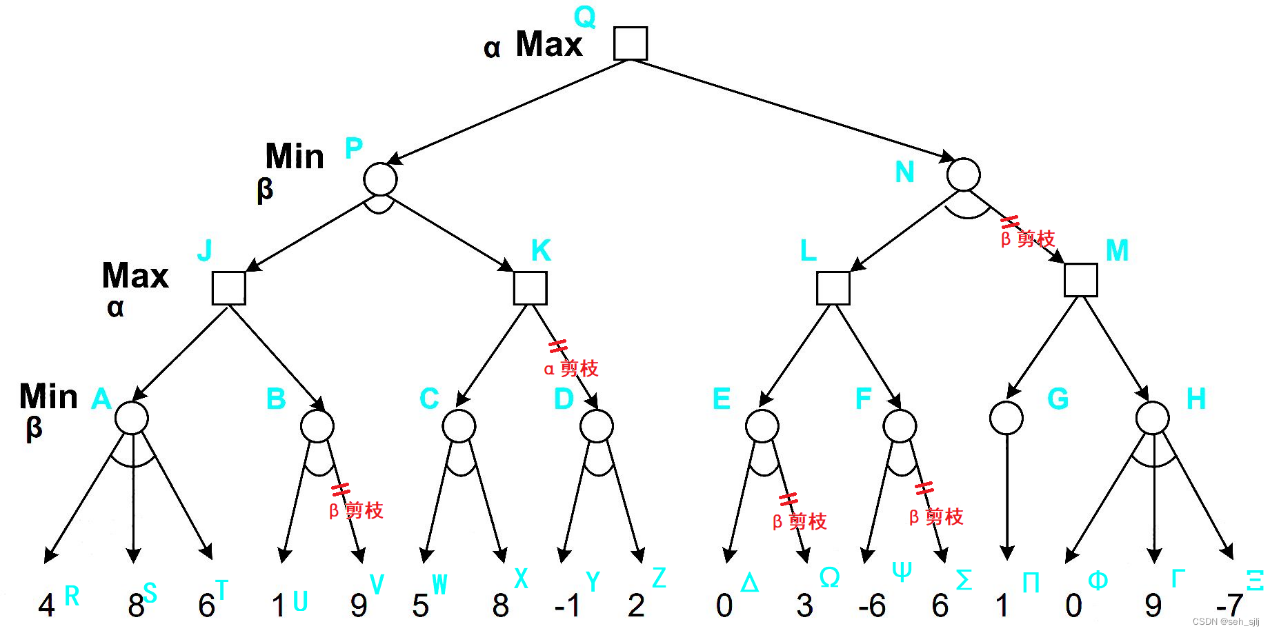


二、α-β剪枝（Alpha-Beta Pruning）

α-β剪枝是一种优化方法，在博弈树生成的过程中同时计算各节点的估计值及倒推值，通过对估值的上下限进行估计，减去没有用的分支，减少搜索范围，提高效率。

用一个实际的例子来说明：如果你和一个人在下棋，现在轮到你走。现在你有两种选择：走“A”或者走“B”。如果走“A”，那么你的局势会变好。走“B”也比较

好，但是如果你走“B”的话，对方可以在两步之内获胜，这对你是非常不利的。也就是说，你考虑到了走“B”的最坏结果，那么其他可能的结果就可以不考虑了（因为对手不傻，肯定会想方设法使你败北），那么你相当于在博弈树中剪掉了“B”的其他情况。最终，因为“A”至少不会让你在两步以内输棋，所以你选择走“A”



三 算法分析

一 极大极小搜索

极大极小搜索是一种利于人类理解的算法,其过程与人类在进行对弈的过程十分相似,下棋并不是走一步看一步,而是要对后面几步进行推演,强大的五子棋棋手甚至可以预见7步之外的局势.但是人脑算力毕竟不如计算机,借助于强大的算力可以使得AI拥有超过人类棋手的棋力.

在五子棋中借助极大极小搜索,首先得解决的第一个问题是如何对一盘五子棋的局势进行评估,即求评估函数

二 评估函数

首先一个很明显的结论，如果我在某个位置下棋，在每一行最多影响到它左边的4个，一直到它右边的4个，加上自己，这一行就一共是9个点，更远的点不受它的影响。所以对于每个位置，计算一个点的评估函数f(x)

同样地，对于列、对角线、反对角线，也是一样。

那么，每个点，我都可以得到4个方向上的可能被影响的点。每个方向上是9个点。

如果对这9个点做如下编码：如果这个点没有超过棋盘范围，是自己颜色就记为1，是对手为0，是空记为#, 那么就可以构建出一个长度为9的字符串。

每个点都可以构建这样4个长度为9的字符串。

例如，“111##1111”表示在这9个点中，左边3个都是我的棋，中间空了2个没人下，右边4个都是我的棋。

如果下面要轮到我下棋，而我就想要下在这两个空中的某一个，显然，落子在右边这个我就赢了，而落子在左边那个，我赢不了，且下一步一定会被对手堵死。

那么很容易想到的是，对于不同的状态，应当是有不同的分数。

简单考虑，如果我在长度为9的这个字符串中找到了“111”，我就给10分，而找到了“1111”我就给20分，那么对于每个点构建的4个长度为9的字符串，我都可以通过计分的方法来给每个点一个分数。

如果把这个分数乘上这个点所在位置的重要程度，正好是可以作为这个点的评估值。

首先给出一些术语的介绍：

成五：五颗同色棋子连在一起

图片包含 门, 游戏, 游戏机, 建筑

描述已自动生成

活四：有两个点可以形成五

图片包含 门, 游戏, 建筑, 钟表

描述已自动生成

冲四：只有一个点能够形成五，要么是一头被对手堵住，要么是只有中间能连起来

图片包含 门, 建筑, 游戏机, 游戏

描述已自动生成

图片包含 门, 游戏机, 游戏, 建筑

描述已自动生成

图片包含 门, 游戏, 游戏机, 建筑

描述已自动生成

余下的还有活三,眠三,活二,眠二等等此处不一一列举

最后评估值为自己的评估值减去对手的评估值。

三 遗传算法

如何才能得到一个较好的评估函数? 我们可以让AI自己生成一份权重, 再经过不断地调整达到最优.具体来讲,我们先人为地估计每个棋形的得分,例如

static const FormsScore Forms[FORMS\_SIZE] = {

    {"11111",1e5},

    {"\*1111\*",1e4},{"01111\*",3e3},{"1\*111\*",3e3},{"11\*11",3e3},

    {"\*111\*",1e3},{"0111\*",3e2},{"\*1\*11\*",3e2},

    {"\*11\*",100},{"011\*",50},

    {"\*1\*",10},{"\*10",1},

    {"00000",-1e5},

{"\*0000\*",-1e4},{"10000\*",-3e3},{"0\*000\*",-3e3},

{"00\*00",-3e3},

    {"\*000\*",-1e3},{"1000\*",-3e2},{"\*0\*00\*",-3e2},

    {"\*00\*",-100},{"\*001",-50},

    {"\*0\*",-10},{"\*01",-1}

};

这个列表里的每个元素为(棋形,得分),一共24个元素, 此时的得分只是人为的估值,我们可以生成生成一份(w1,w2,…w24) wi∈1~100的权重向量,真实的得分Score真i = wi\*Scorei ,因此,我们的目标便是生成一份最优的权重向量.我们可以借助遗传算法来完成这个过程

创建一份初始种群种群中每个个体都是随机生成的一份权重向量

WeightVec weightGentor() {

        WeightVec result;

        static std::uniform\_int\_distribution<int> uni(1, 100);

        result.reserve(FORMS\_SIZE);

for (int i = 0; i<FORMS\_SIZE;i++)

result.push\_back(uni(m\_rng));

        return result;

}

对每个个体进行适应度评估

这里我们采用的策略是让AI使用这个个体的权重和1000个随机 权重的AI对弈,然后胜率就是这个个体的适应度

float fitness(const WeightVec& weight,const Group\_Ty& againstGroup) {

        int len = againstGroup.size();

        int count = 0;

        for (const WeightVec& p : againstGroup) {

            if (m\_game->against(weight, p))count++;

        }

        return (float)count / (float)len;

    }

选择淘汰与交配繁衍

从种群中选出前K个适应度最高的个体进行交配繁衍,剩下的全 部淘汰.对这K个个体进行基因交叉,这里个体的每一项权重都视 为个体的一段基因,基因交叉就是权重的交叉.为了防止陷入局部 最优, 在这个过程结束后,产生的新个体有一定概率发生基于突变.

将整个过程重复N次,直到整个种群的基因达到近似相同

这里截取一段运行时的数据

电脑屏幕截图

中度可信度描述已自动生成

Update:1即为第一代,下方的数据即为当前一代每个个体的胜率.第一代时各个个体的胜率参差不齐,但由于第一代是优选出来的缘故,胜率都比较高.

形状, 箭头

描述已自动生成

Update 2~3 时低分个体基本被筛选出去了,为数不多剩余的低分个体基本是基因突变的结果

图片包含 建筑, 照片, 女人, 大

描述已自动生成

第10代往后,胜率基本稳定,基因也大体上趋同,再往后的迭代除了基因突变以外,大部分都是同一批基因.在20代左右就可以终止了

四 总结

极大极小搜索和αβ剪枝实现起来比较简单,但是性能开销极大,当递归深度超过5层时已经明显的吃不消.要想提高性能可以不断优化评估函数和搜索方式,当分值较高的局面被优先搜索到时,就可以快速地剪掉分值较低的局面.

用遗传算法来训练AI时为了时间考虑递归深度仅仅为1,所以训练出来的AI对高递归深度时的适用度较差.另一方面,AI是在我们人为给定的一个棋形得分基准值基础上训练的,所以实战的效果要取决于基准值是否合理.

如果不考虑基因突变,整个遗传算法过程产生的子代就是初始种群基因的排列组合,所以若初始种群数量过少,很快就会陷入到局部最优中,这时想让种群得到新的提高只能依赖基因突变,但这其实无异于全局搜索了,这也是遗传算法的局限性,难以摆脱局部最优.

目前得出的最强AI权重为{25,12,56,63,13,35,37,36,53,57,64,94,80,63,81,61,53,77,99,71,32,81,10,13}

这个AI与1000个随机AI对战的平均胜率为99.8%~100%