人工智能实验报告
$\alpha-\beta$剪枝算法

小组　lgd-cs
赖启东*　14347058
黎丁嘉　15336077
李振宇　15336092
刘键涵　15336110
洪培衔　15336059
2015级计算机科学与技术
School of Data and Computer Science, Sun Yat-Sen University, China
laiqd@mail2.sysu.edu.cn

December 26, 2017

Abstract

本次实验我们采用了$\alpha-\beta$剪枝算法作为中国象棋博弈程序算法的核心，在剪枝深度为5时，几秒就可以找到走步；剪枝深度为6时，60秒内基本可以找到走步。相比于极大极小搜索，效率上有了极大的提高。

1 Introduction

1.1 中国象棋

中国象棋博弈程序，其核心是一个搜索算法，即如何从搜索一个走步使棋局对自己最优。一个好的博弈程序应该具备：一个好的生成过程；一个好的静态估计函数。对于中国象棋而言，其难点在于如何估计当前棋盘状态，以及电脑走步的生成，因为象棋棋子种类较多，且各有各的规则，所以会出现很多种情况需要进行判断。

1.2 $\alpha-\beta$剪枝

α实际代表极大层倒推值的下界值，β实际代表极小层倒推值的上界值。该算法在极大极小搜索算法的基础上，不需要生成全部的搜索树后再进行端点静态估计和倒推值计算，而是在计算时通过比较α和β的关系而剪掉一些无用的分枝。因此$\alpha-\beta$采用的是有界深度优先搜索，当生成节点达到规定的深度时，就立即进行静态估计，而一旦某个非端点节点有条件确定倒推值时，就立即赋值。

2 Algorithm Design

2.1 程序流程图

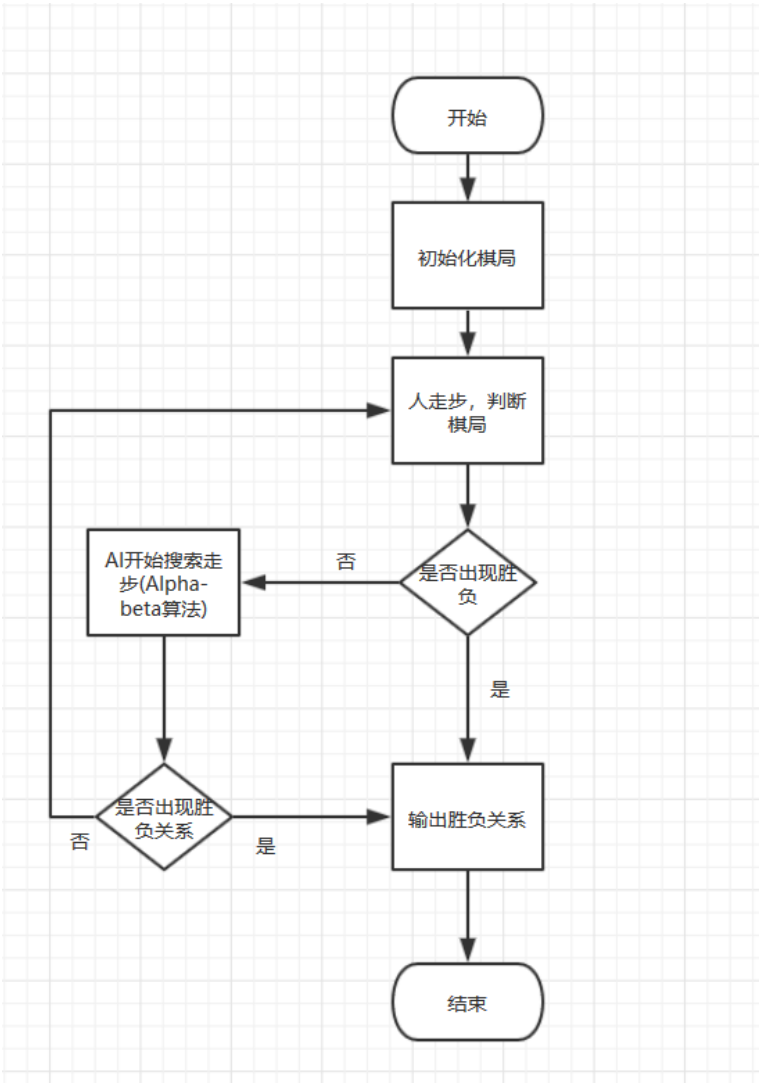

Figure 1: 图1

2.2 搜索算法

$\alpha-\beta$实际上是一种有界深度搜索，所以我们采用了递归实现。

关键函数:[1]

Evaluate()用于对当前棋局进行估值。

Generate_ALL_Legal_Moves()用于产生所有当前局面下所有的后继局面，并保存到**Legal_Moves_List[]**中，并根据该列表进行深度有界递归搜索。

Make_This_Move()和**Undo_This_Move()**分别用于将当前局面更新为一种后继局面和撤销后继局面恢复原局面。

递归算法伪代码如下：

Algorithm 1 AlphaBeta(*depth*, *alpha*, *beta*)

```
if depth为0 then
  return Evaluate()
end if
Generate_ALL_Legal_Moves()
repeat
  Make_This_Move()
  val ← -AlphaBeta(depth − 1, −beta, −alpha)
  Undo_This_Move()
  if beta ≤ val then
    return beta
  end if
  if alpha < val then
    alpha ← val
  end if
until Move is not in Legal_Moves_Lists[]
return alpha
```

2.3 棋局估价函数Evaluate()

为了较好的估计棋局，首先我们需要设置各种棋子的静态棋力值，我们查询中国象棋相关资料得知，关于棋力值大小设置，有一个基本标准。

棋子静态棋力值 兵/卒100 士200 相250 马450 炮500 车1000 将/帅10000

如果只是根据静态棋力值加和的话，棋盘估计值效果很不理想。所以我们要根据不同棋子在不同位置以及棋子可以去的位置的情况进行综合判断。比如，可以将军的“马”实际价值肯定比只能吃兵的“车”价值高，因此这时候应该给马更大的估计值。

因此，我们为了完成棋局估计，首先需要的是对应不同种类棋子的邻域获取函数，通过获取棋子的领域，遍历领域中的其它对方棋子(是否能吃)和己方的棋子(可协防己方棋子)，知道该棋子当前的实际价值。同时，我们获取邻域以后，还可以根据一个棋子周围的可走路径来定义他的灵活度，灵活度越大的棋子，越有走步的价值。灵活度不能仅由可走路径数决定，因为同样的可走路径下，车肯定比兵厉害。所以灵活度应该是棋子静态灵活度加权棋子灵活位置值的结果。

棋子灵活度加权值 兵/卒2 士3 相5 马10 炮20 车15 将/帅1

即我们评估棋局中一个棋子时，从三方面入手：棋子静态棋力值BaseValue；棋力当前位置价值(取决于可吃的对方棋子和附件的己方棋子)AttackPosValue和GuardPosValue；棋子当前位置灵活度(可走走法数)$FlexPosValue * FlexWeight$。三种评估方法相互结合，对棋盘能有一个比较好的估计。

2.4 走步生成器Generate_ALL_Legal_Moves()

走步生成器是程序中比较繁琐的部分，因为情况实在是太多，所以对于走步是否合法的判断，会涉及到许多特殊的判断。比如说相不能过河，蹩马脚等，都要作判断。而像兵这种，在过河前和过后会走法不一样，也要区别处理。

因此我们可以把走步生成器分为两个部分，一是根据棋子种类的不同走法而产生不同走步的模块，包含多个函数；二是当前走步合法判断的模块，通过switch语句，对红方、黑方所有棋子做判断。

3 Computational Experiment

3.1 实验环境

程序由MFC编写，IDE为visual studio 2017，Windows SDK版本为10.0.16299.0。

3.2 实验结果分析

可以看到搜索深度为3和为5时，程序都在较短时间内找到了走步。另外从进度条没有走满可以看出，程序并没有遍历所有的可能走步之前就找到了答案。

在搜索深度达到6以后，程序执行的时间达到了百秒级，需要等待很久才能找到走步。

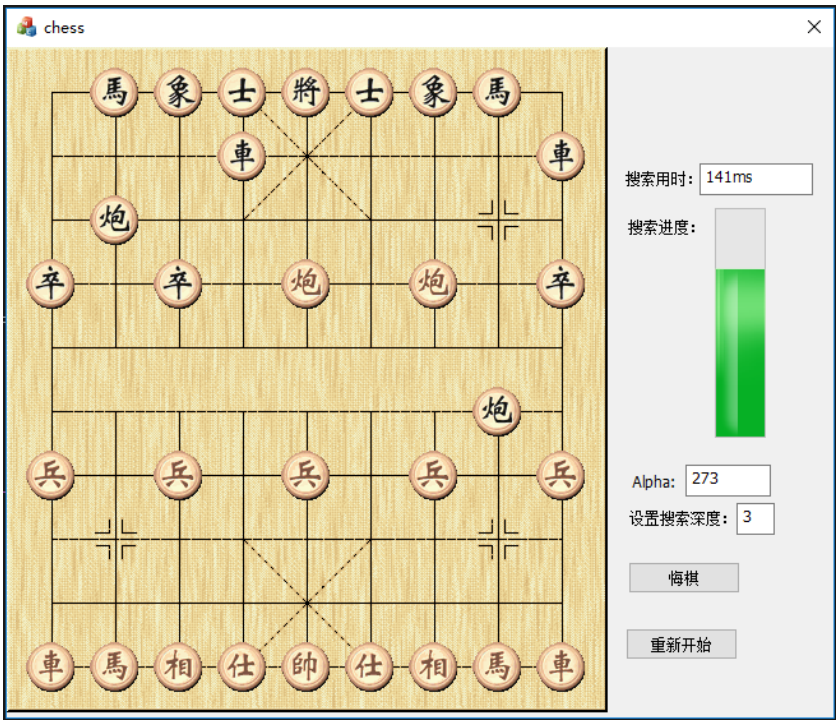

Figure 2: 搜索深度为3

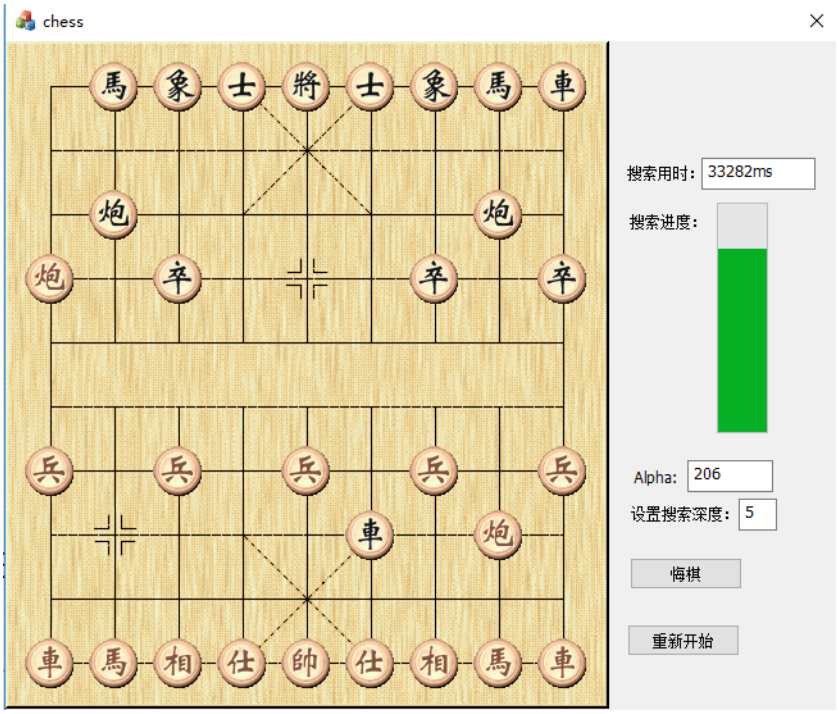

Figure 3: 搜索深度为5

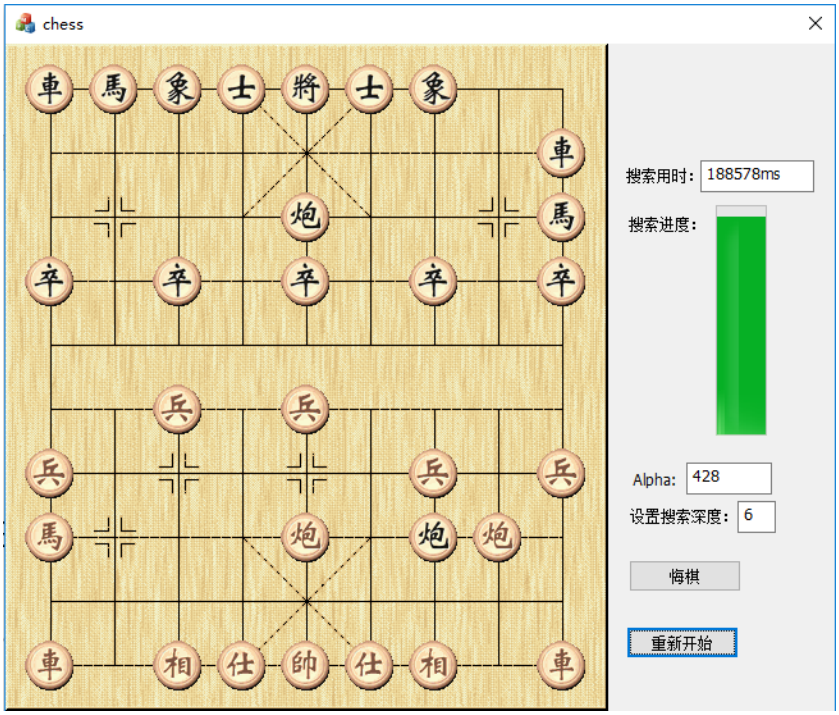

Figure 4: 搜索深度为6

3.3 算法不足

算法在深度较高时依然存在较大的问题，要么出现程序未响应(但是最后会恢复并显示结果)，要么直接报错。所以我们的程序还谈不上有多高效，甚至可以说在执行效率上存在较大的问题(不是$\alpha - \beta$)的问题。

4 Conclusion

这次实验总体偏难，主要是象棋涉及到的规则太多，程序将会十分繁杂；同时对于完成棋局估值函数，需要比较专业的象棋知识。因此这次的程序大量参考了github上的象棋代码，并且查阅了许多象棋相关的资料来完善程序。不过最后成功完成实验后，让我们对棋类博弈有了进一步认识，明白了一个好的静态估计函数是多么重要。

5 小组分工

黎丁嘉完成$\alpha - \beta$搜索引擎，并负责将各个算法模块整合到算法框架中以完成程序。
李振宇完成算法框架，留下各个模块接口。
赖启东完成走步生成模块。
洪培衔完成棋局估价函数。
刘键涵负责查阅象棋相关资料，为走步生成模块和棋局估值提供理论基础。
实验报告由黎丁嘉和赖启东完成撰写，平时实验过程记录由黎丁嘉完成。

References

[1] 象棋百科全书, "http://www.xqbase.com/computer.htm," 2017.