中山大学

数据科学与计算机学院

软件工程

人工智能

Alpha-Beta 剪枝算法实验报告

Submitted By:

徐伟元 16340261

熊永琦 16340258

李天译 16340122

Submitted To:

王甲海

教授

大数据与计算智能研究所

2019-1-4

Abstract

本实验使用 Javascript 编写代码完成 Alpha-Beta 剪枝算法，实现 AI 中国象棋，并提供 Web 图形化界面和人机对战功能。

1 实验题目

编写一个中国象棋博弈程序，要求用 alpha-beta 剪枝算法，可以实现人机对弈。棋局评估方法可以参考已有文献，要求具有下棋界面，界面编程也可以参考网上程序，但正式实验报告要引用参考过的文献和程序。

2 实验步骤

2.1 界面

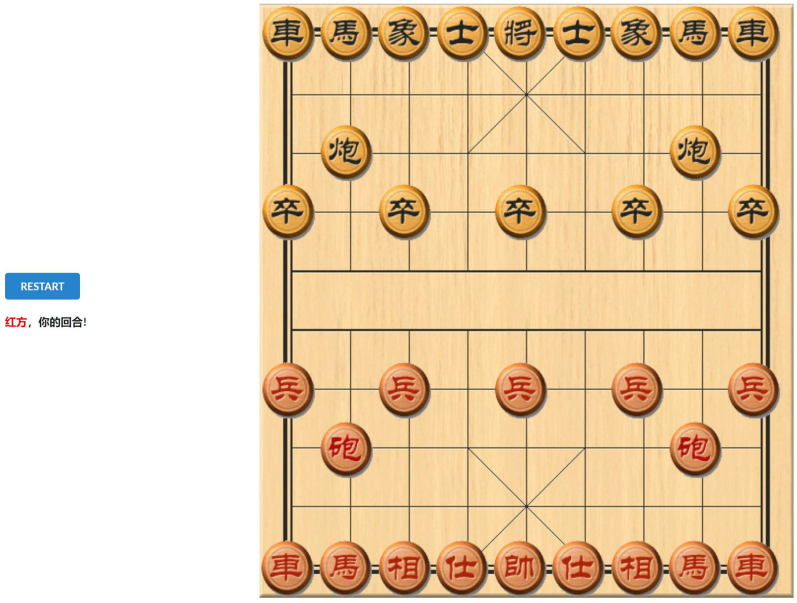
界面采用 web 页面实现，其中棋盘与棋子使用网络图片。棋盘共有 10 行 9 列，每一个棋盘点，使用一个 img 标签填充。如果棋盘点有棋子则使用棋子图片，否则使用透明图片填充。

对于棋盘状态，考虑后续需要进行搜索，使用字符串代替二维矩阵存储棋盘状态，避免存储状态开销过大，如初始棋盘状态为:

RNBAKABNR
000000000
0C00000C0
P0P0P0P0P
000000000
000000000
p0p0p0p0p
0c00000c0
000000000
rnbakabnr

这样可以节省存储开销，加深搜索深度。其中，采用西洋棋棋子英文首字母代表棋子，如 R 为 Rook，代表车，N 为 kNight，代表马。具体棋盘代码见 board.js

最终界面如下：



2.2 规则编写

主要需要满足各个棋子的行走规则：

1. 车直走
2. 马走日，注意蹩马脚
3. 象走田，注意不能过河和蹩象脚
4. 士走斜线，注意不能出九宫格
5. 帅走单步，注意不能出九宫格
6. 炮直走，吃子要跳一子

7. 兵走单步，过河前只能前进，过河后可左右，注意不能退后

具体的规则代码见 chess.js

2.3 双人对战

双人对战需要实现游戏规则的轮换，走子的确定。其中前面实现了棋子的行走规则，所以我们走子只需要完成一个走子函数，传入棋子类型，调用前面的走子规则来判断是否可行即可。棋局轮换是为了后面的人机对战做准备。首先实现一个双人对战，后面将一方改为 AI 即可。在一轮过程中，判断逻辑如下：

1. 目前的执行玩家身份
2. 执子是否为玩家所拥有的棋子
3. 判断所下位置是否合法 (棋盘内，走子规则允许)
4. 走子后，棋局是否结束

这样执行一轮走子过程，直到最后决出胜负。具体代码见 game.js(已更新为 AI 对战版本)

2.4 棋局评估方法

棋局评估主要和玩家当前拥有棋子和棋子所在位置有关。这里参考了网上博客的一个棋局评估函数，考察了不同棋子在不同位置的棋力。将当前玩家拥有棋子的棋力相加，作为当前棋局的评估值。

具体编码见 value.js。

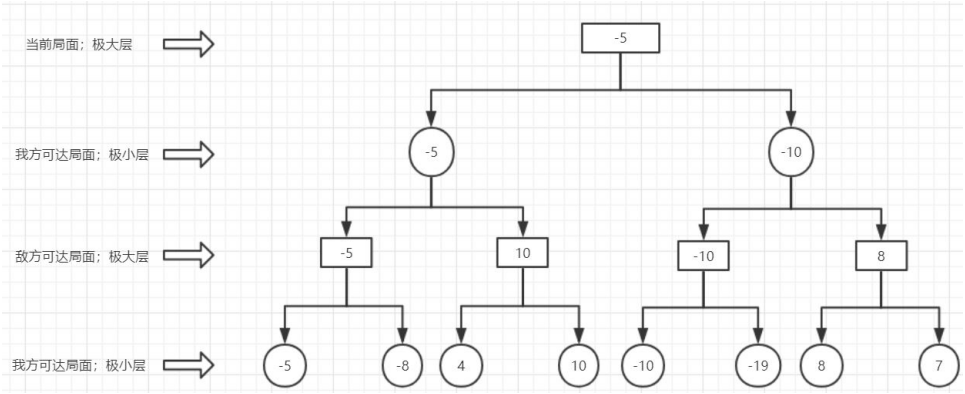
2.5 人机对战-极大极小值搜索

极大极小值搜索，主旨思想是在当前棋局，通过预测棋手双方的走步，并评估每个走步棋局的局面估价。根据当前下一个走步为我手，还是他手，

对于预测的所有下一个走步局面估价选取极大值或极小值。因为这里我们假设棋手双方都是聪明的，会选取对自己最有利的局面。

分析一下：如果下一步是我手，那么对于所有我方走步，我方一定选择对自己最有利的局面，也就是选择极大值；如果下一步是他手，那么对于所有他手走步，他足够聪明，他一定会选择对他最有利的局面，也就是极小值。

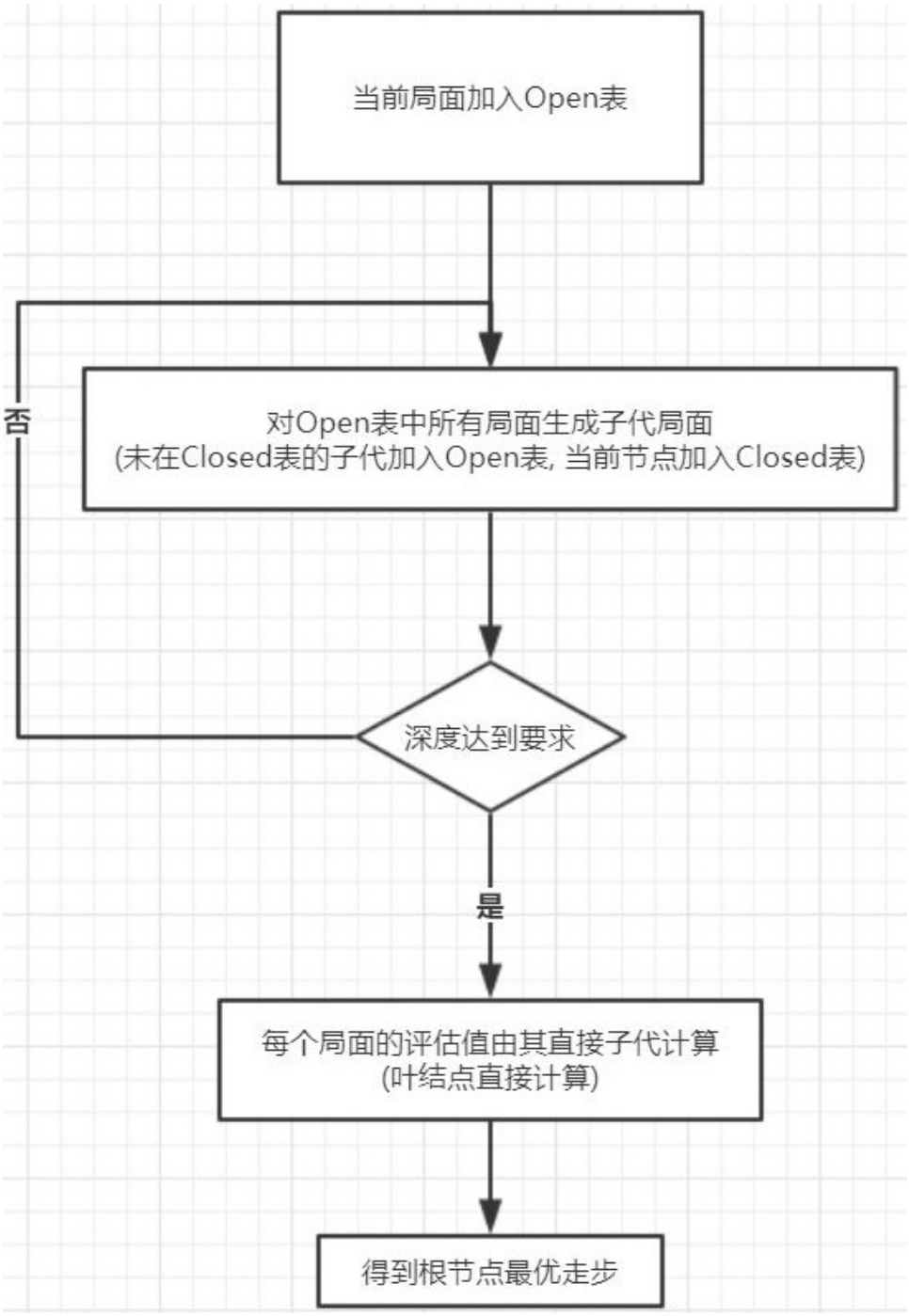
例如下图，就是一个极大极小值的例子：



接下来，考虑一下每一个局面所对应的图节点所需要记录的数据。

1. 局面的评估值
2. 局面所处深度
3. 生成的所有下一步局面，因为当前局面的评估值由这些局面计算得到
4. 所走棋子的移动位置，这样才能知晓选择局面是如何到达

那么算法流程如下图：



具体代码见 AI.js，内附详细注释。

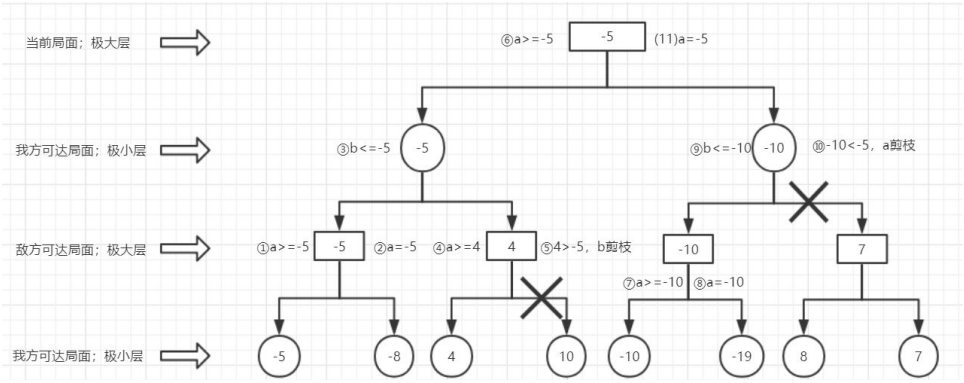
2.6 人机对战-alpha-beta 剪枝

alpha-beta 剪枝是对极大极小值搜索的优化。由于极大极小值搜索是遍历所有节点，是穷搜的算法，所以可能会使得搜索宽度过宽而搜索无用节点，导致浪费资源。这里的剪枝，就是在遇见节点值可被判定无用时，放弃对当前节点后续子代的搜索。那么在极大极小值搜索中，什么时候可以判定放弃搜索呢？首先，明确一个事实：对于一个极小层节点的评估值 b，搜索其子节点时，可以同时更新 b 值。如一个子节点评估值为 5，极小层节点评估值最大为 5，因为当前节点会选择最小的。同理，一个极大层节点的评估值 a 可以同时被评估。来看两个情况，说明何时放弃搜索：

1. 已知当前层为极小层，评估值设为 b，其父节点为极大层，评估值设为 a，且 a 最小为 10。在搜索当前节点的第一个子节点时，其评估值为 5，那么当前节点值最大为 5，而 a 就不可能搜索这个节点的后续节点了，因为前面有 10 可以选择。

2. 已知当前层为极大层，评估值设为 a，其父节点为极小层，评估值设为 b，且 b 最大为 1。在搜索当前节点的第一个子节点时，其评估值为 5，那么当前节点值最小为 5，而 b 就不可能搜索这个节点的后续节点了，因为前面有 1 可以选择。

和极大极小值搜索给出例子对应的 alpha-beta 剪枝例子如下：



对应的算法代码，采取了递归实现，因为每一个父节点的值，需要传递给子节点，以作剪枝判断。具体代码见 AI.js，内附详细注释。

3 实验结果与测试

1. 完成极大极小值搜索后，由于属于穷搜算法，搜索深度只可达两层。本人与其对战，其智能程度不足

2. 更改为 alpha-beta 剪枝后，搜索深度可达 4 层。本人与其对战，胜负各半；两位舍友与搜索深度为 2 层的剪枝 AI 对战，舍友战绩三负一平

References

[1] http://www.cnblogs.com/royhoo/p/6425658.html