2022-05

# 长风正劲

## 五子棋博弈的实现和优化

局面评估/min-max 博弈/搜索顺序优化
深度优先搜索/迭代加深/α-β剪枝
哈希/随机数生成
字典树/对局经验记录
多文件编程/面向对象的程序设计
图形用户界面
性能优化/动态内存分配

白骏 | 王重凯 | 孙梦飞
计科 • 2021

# 综述

综合使用了多种算法和数据结构完成了五子棋博弈程序设计。

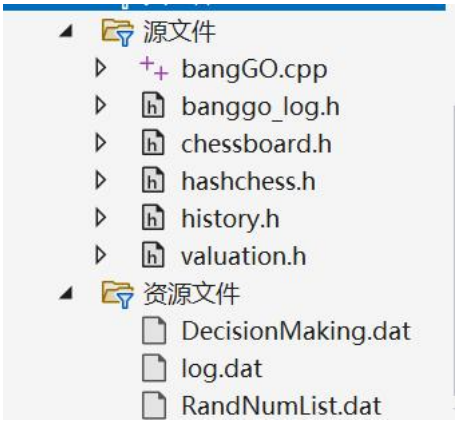


图 1 该项目的框架

该程序以**深度优先搜索**为基础。该过程融合了 **min-max 博弈**原理的思想，进行了**迭代加深，优化了搜索顺序**，并且进行了 **α - β 剪枝**。同时，使用了 **zobrist 哈希算法**完成了状态的判断，进一步**降低了搜索树的规模**，达成了“**非必要不搜索**”的目标。

为增加程序的**智能性**，又利用**字典树**等数据结构录入了开局棋谱。在决策时，程序会**检索当前局面哈希值**，如果能够获得对策，就可以降低搜索树规模、进行**灵活决策**、快速获得结果。对局面进行评估，引入了**动态修改**的思想。

本项目吸收了现代化的编程思想和成熟的算法思路。例如面向**对象的程序设计、多文件编程、动态内存分配等**。同时，设计了**一种基于自然语言的高质量随机数的生成方法**，该方法操作简单、性能良好，可以进行推广。

| 变量名 | 含义 | 属性 | 值 |
|---|---|---|---|
| **N** | **棋盘的尺寸** | **全局变量** | **15** |
| **cnt** | **最大搜索次数** | **全局变量** | **4000000** |
| **maxDeep** | **最大迭代加深深度** | **全局变量** | **10** |
| **SIZE** | **哈希表的链表数量** | **局部变量** | **100000** |
| $v_i$ | **i 连棋的基础分值** | **局部变量** | $10^i$ |
| **Δ v** | **某一手棋的分值** | **局部变量** | **-** |
| **hashVal** | **局面的哈希值** | **全局变量** | **-** |

# 对局面进行估值

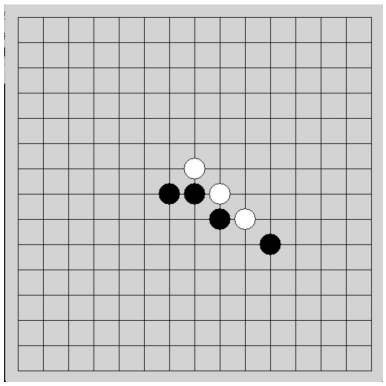

图 2 一个估值的例子

对一个特定的局面，我们需要一个估值函数，来评价局面的优劣。

对有增长空间的 1 连、2 连、3 连、4 连、5 连分别给与 $v_1$=1，$v_2$=10，$v_3$=100，$v_4$=1000，$v_5$=10000 的基础分值。若两头均有增长空间，则乘以系数 10。

以上评估算法的时间复杂度为 $O(N^2)$，在搜索层数较高时，搜索树的叶子节点较多，造成了运行缓慢。为此，引入了**动态估值**的思想。

考虑白方的棋子(8,8)，落子后使己方增加了一个 3 连，且降低了对方 2 连的增长空间，因此该棋子的价值$\delta v$ 为$(v_1 - 2v_2) - (2v_1)$。

每次落子后，把$\delta v$ 累加到估值即可。

# min-max 博弈思想

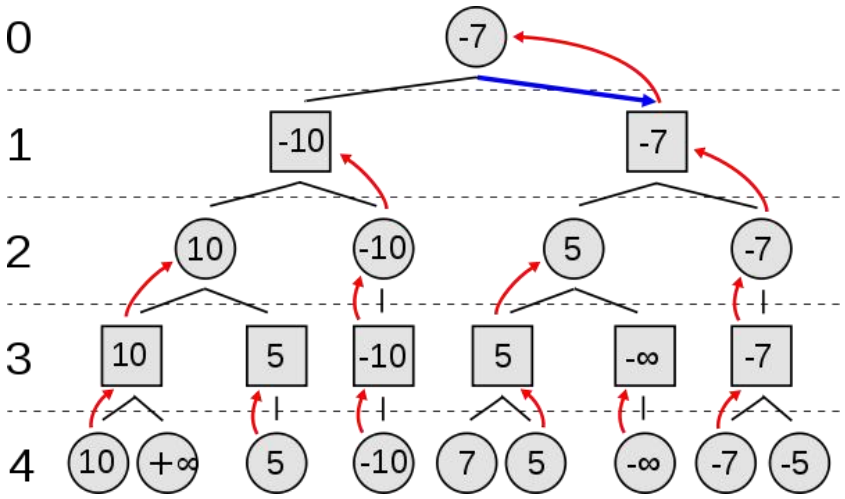


图 3 min-max 博弈示意

对于某一个特定局面，进行多种决策，会得到更多的局面，而更多的局面又可以产生新的局面。基于图论的思想，串联起这些局面，就构成了搜索树**(Search tree)**。

在搜索树的底层，我们对每个局面进行估值。

对于奇数层为 MAX 节点，而偶数层为 MIN 节点。奇数层的任务是取得最大值，偶数层的任务是取得最小值。

## α - β 剪枝优化和搜索顺序优化

如果在下层的 MAX 节点发现得到的值为 5，那么意味着上层的 MIN 节点的数值一定不大于 5。而如果本层的 MAX 节点的值已经至少为 8（比如因为下层的 MIN 节点为 8），那么就可以剪掉右边的分支（即右边的分支不必继续探索）。

可以看到，如果我们运气很好（实际中依赖估值函数预先猜几个），直接搜到了最好的情况，alpha-beta 可以剪去大部分分支。为此，我们在某次决策时，优先搜索δv 值高的招法。

## 生成高质量的随机数

大多数的随机数生成算法都返回一个[0,m]之间均匀选择的整数。C/C++的 rand 中 m 定义为 RAND_MAX，通常是 15 位整数 32767。srand(seed)函数用来设置初始种子，通常使用当前的时间作为熵来源，如 srand(time(NULL))。大多数 C/C++的 rand 实现都是 LCG/TLCG 算法。线性同余生成器在加密方面是相当糟糕的选择，它们生成质量很低的随机数序列，表现出多种偏向性。

我们采取了非常 amazing 的随机数生成算法。

首先找到一篇英语文章，对每一个单词取哈希值，然后转为 0/1 比特串。

然而，自然语言有严重的重复性，例如字母 e 出现的频率较高。and、or、to 等介词也高频出现。

我们另取数篇英语文章，进行同样的操作，然后对 0/1 比特串进行异或运算。即可生成高质量的 0/1 随机串。

## Zobrist 哈希算法

为棋盘的每一个格子分配 2 个随机数，分别对应黑方和白方。棋盘为空时，哈希值为 0。若在某一点落子，则在哈希值被更新为哈希值与对应随机数的异或值。若删除某个棋子，则再次异或即可。

该哈希算法不受着法的顺序影响，只和当前局面有关。

为了保证哈希的正确性，采用了 64 为无符号整数哈希值。

## 录入棋局

在学习过程中，程序会记住特定局面的对策，从而减少搜索时间、增加搜索深度。

利用哈希算法机型局面的检索。

# 参考资料

[1]可能是 github 上最受欢迎的五子棋 AI，源码+教程 lihongxun945/gobang: javascript gobang AI，