集训队泛做题解

中山纪念中学 周铭洵

2015 年 10 月 16 日

1 Codechef MANYLEFT

1.1 题意

一个 $N*N$ 棋盘，有些棋子，类似于跳棋，相邻的可以跳过去（假如跳过去那格是空的）然后被跳过的就消失了。然后就不停地跳直到再也跳不了，求尽量优的方案让剩下的棋子最多（不要求最优解）。范围是 $10 <= N <= 30$。

1.2 关键词

搜索，A*，Beam Search，随机算法

1.3 题解

这题大致有两个思路：搜索和随机。

搜索

搜索的话，由于不是找最优解，而且时间非常有限，考虑 A* 算法或者 Beam Search 算法。A* 算法大家比较熟悉。后面那个算法大致是强制令搜索树的每一层节点最多只有 B 个（B 是一个合适的常数）。两个算法的核心都是估价函数。其实有一个靠谱的估价函数无论是哪种算法都能得到不错的解。这题估价函数有很多种。一种效果很不错，而且比较简单的是对于一个局面的每个棋子，如果有棋子与它相邻就令当前局面的得分减去常值 C。然后这样搞基本上能拿到 90 分左右（雾）。

随机

其实仔细考虑一下的话，在这题中随机可能会比搜索要好。因为每个局面的决策实在是太多了，而且会有大量价值相差不大的局面，我们难以在时间和空间都比较紧张的情况下保证质量高的解出现在决策树当中。

那么随机调整在这题中效果非常好。因为我们可以用不多的调整来尝试使一个解变优。我们可以一开始随机一些解，找到一个较优解，然后每次重构解后面的部分，或者是去掉解中间的某些移动，然后再重构。当然如何重构解决定了我们最终解的质量。一种比较简单的方法是每次选择能使局面的相邻棋子数目减少最多的那个移动，然后一直动下去。在所给时间内尽可能的尝试，找到更优的解。目前最优的算法好像都是随机（本题中）。

其他

还有一个值得注意的是如果将棋盘黑白染色，那么黑格的棋子永远是通过移走一个白色的棋子然后跳到黑格子上。反之亦然。所以一种不错的策略是尽量移动一种颜色的棋子。实践证明这个策略还是比较有效的。

1.4 算法流程

- 求一个不错的解
- 随机砍断或去掉一些移动
- 剩下的移动先动，构造一个新解
- 比较最优性，更新答案
- 重复这个过程

1.5 时间复杂度

没法算。