李翰韬 160324 16711094

我们知道，基于启发式搜索算法的博弈树模型，通过启发式算法，对于规则所对应的信息，设计相应合适的评价函数；通过极大极小分级法做出阶段搜索策略，减少计算量；通过剪枝方法做出对分支的选择，减少计算量；从而得出当前局势下的“一步好棋”。

然而，这样的博弈树模型，象棋程序可以下赢国际大师而对于基于搜索的计算机围棋却很难。我认为主要原因可以概括为以下两点（由于本人对国际象棋规则并不熟悉，以下讨论使用中国象棋规则）：

1. 围棋棋局数量远远大于象棋棋局数量

虽然能够通过剪枝算法和各种阶段搜索策略减少程序需要遍历的决策树分支，减少程序的计算量，但是总体棋局数量的大小导致的计算量差距仍然是一个不可忽视的因素。举出一个例子，对于应用蒙特卡洛算法的 Pachi 模型，其官网上给出的模型棋力如下：

The default engine plays by Chinese rules and should be about 7d KGS strength on 9×9. On 19×19, it can hold a **KGS 2d** rank on modest hardware (Raspberry Pi, dcnn) or faster machine (e.g. six-way Intel i7) without dcnn. When using a large cluster (64 machines, 20 cores each), it maintains KGS 4d and has won e.g. a 7-stone handicap game against Zhou Junxun 9p.

可以看出，棋盘大小、处理器计算能力，都极大地影响了模型的棋力。

象棋博弈树宽度远远小于围棋博弈树，对于 361 个落子位置的围棋而言，象棋每个棋子都有固定的行走规则，每次行棋可以选择的落子方式数量显然小于围棋。

象棋博弈树深度远远小于围棋博弈树，在一般的比赛规则中，象棋规定 60 步内不吃子为和棋，则每局象棋理论博弈树最深为 32*59=1888 层。而对于围棋而言，在“禁全同”规则下，正所谓“千古无同局”，在 19 路标准棋盘下的理论棋局数量满足下式：

$$10^{10^{117}} < N(19) < 10^{10^{171}}$$

可以看出，两个数值完全不在一个数量级上。

以上导致围棋的博弈树规模、计算复杂性远远大于象棋，搜索算法的难度也直线上升。

2. 很难对围棋设计出合适有效的评价函数

对于启发式搜索算法，重中之重是设计出合理的评价函数。

对于象棋，评价函数的设计已经较为困难了。目前主流方法是对每一颗棋子进行打分，而分数随棋局时间、棋子位置、其他棋子位置而随时变化。例如，开局时棋子较多，很容易架炮，而马却容易被蹩马腿，此时炮的分数比马高；但随着进入残局，马显然更为灵活，此时马的分数比炮高。这些规则由算法的设计者规定，而算法的有效性取决于设计的评价函数是否合理。此外，象棋拥有较为明确的目标——吃掉对方尽量多的，尽量重要的棋子。根据这个方向去建立评价函数，总是合理的。

然而围棋的规则较象棋却抽象地多，围棋棋子间相互并无区别，而且很难找到类似于象棋“将军”的明确的输赢条件。对于某个棋局，或许可以设计某种评价函数，综合评价棋局此时的优劣，但很可能在下一着时彻底变化，而对于 361 个可能，设计者是无从下手的。

设计评价函数的困难也就是直接导致现今 alpha-beta 剪枝搜索算法仍是象棋 AI 的主流，而且十分有效；但围棋却进一步发展出蒙特卡洛算法(Pachi 等)，再到最近的深度学习等机器学习算法(AlphaGo)，发展出新算法的最主要原因必定是原有算法不能解决问题，而蒙特卡洛算法与原有搜索算法最大的区别就是多次对弈计算概率，而不是使用评价函数进行搜索。我认为这个因素的影响远远大于 1 中提到的决策树大小的影响。