论文笔记|Effective Monte-Carlo Tree Search Strat- egies for Gomoku AI

1. 关于 MCTS

1. MCTS 不同于 带有Alpha-Beta剪枝的 Minimax Tree Search
2. MCTS 是一种 best-first 搜索算法并且 generally 比 DFS 快
3. MCTS 采用的 roll-out policy 简单而快速，省去了去通过类似迭代算法估计某个节点的效用值

2. 问题背景

2.1 MCTS 算法

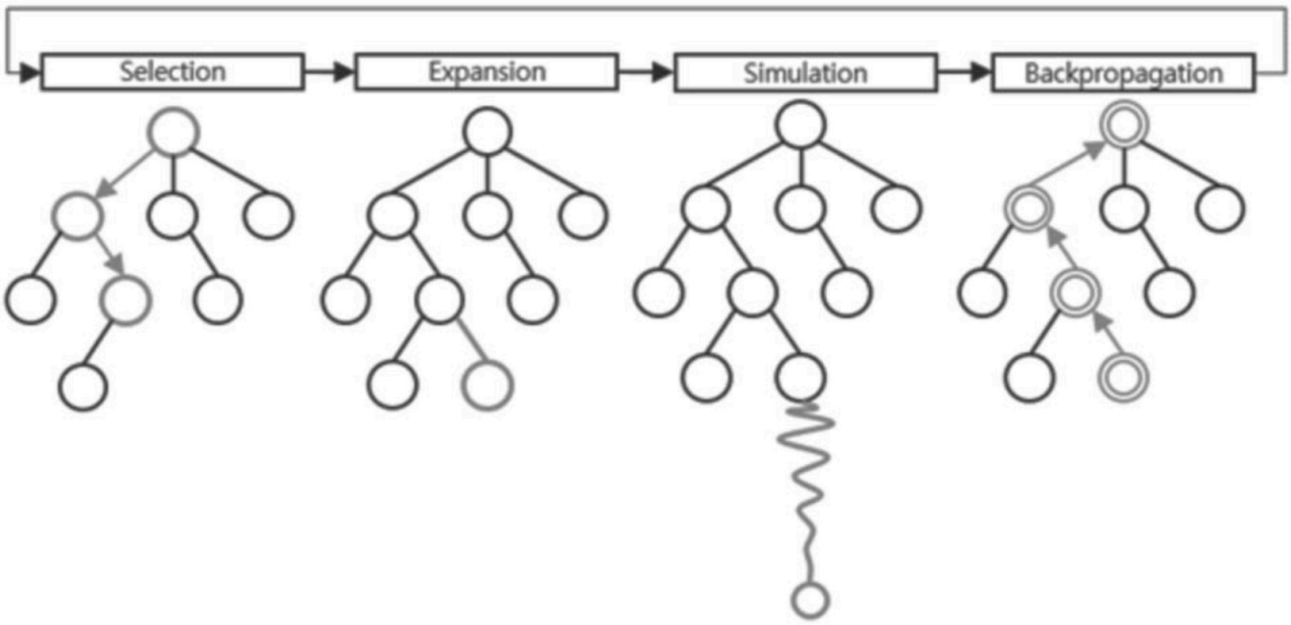

Fig. 1. Structure of Monte-Carlo Tree Search algorithm

Fig. 1 很好地展示了MCTS 的算法框架
其中 MCTS 大致包涵 4 个部分：

1. 选择过程
   the algorithm selects one existing node that want to expand child nodes
   - 这个过程通常可以用随机选取的方式
   - 但是随机选取的方式(flat-MC)也非常容易得到很差的结果
   - 故推荐采用 Upper Confidential Tree (UCT) 的方法
   - UCT 基于 Upper Confidential Bound (UCB) 来做选择
   - UCB 即会在 exploration 和 exploitation 之间进行权衡

$$\mathrm{UCB} = v_i + \mathrm{C} \times \sqrt{\frac{\ln \mathrm{N}}{n_i}} \tag{1}$$

- 每个节点的 UCB 值都基于（1）式给出，其中 N 是节点总个数，n_i 是节点 i 的子节点个数，v_i 是所有子节点结果已知的节点的胜率，C 是一个常数且通常设为 $\sqrt{2}$。

2. 扩展过程
   makes decision that which node is proper to expand the node and make value through the simulation result
   - 同样在 flat-MC 中常用的方法就是随机选取一个子节点
3. 模拟过程
   simulating from the new child node and getting result value that contains just win or lose information
   - 如果是类似随机模拟的话，这个过程会非常简单而快速
4. 回溯过程
   updates all nodes from expanded node to root node and discovers new best node to make next decision
   - 然后就可以基于一种 best-first 的搜索策略来进行数搜索

2.2 Gomoku

1. 本文中采取的Gomoku规则是：Free（即5子以上也算赢）
2. Go 中MCTS取得成就主要来源于一些高效的模拟方法
3. 在普通的MCTS中容易忽略的一点是会忽略对手可能比你更早获得胜利的情况，所以 the evaluation method with turn order factor or reasonable tree search policy are needed
4. 19x19的棋盘中有很大一部分都是不可能去下的地方，所以对这部分用适当的方法进行剪枝十分重要

3. 策略

3.1 Progressive bias

The object of the progressive bias is getting bias in selection phase to choose more significant node to simulate with domain knowledge. It consists of basic form of UCB and add some heuristic values to generate subtle bias. The formula (2) shows it.

$$\mathrm{UCB} = v_i + \mathrm{C} \times \sqrt{\frac{\ln \mathrm{N}}{n_i}} + \frac{\mathrm{H}_i}{n_i + 1} \tag{2}$$

- 它使得那些离初始状态更近的节点更多地依靠启发式的值而不是UCB值
- 在模拟样本较少的时候很有效 --> 不会乱下

3.2 Progressive Unpruning with heuristic values

The Progressive Unpruning can resolve it by control the proportion between preferred nodes and not preferred nodes.

- 渐进式向上剪枝能够通过控制偏好节点和非偏好节点的比例来解决探索大量不必要节点引发的问题
- 当一个父节点的模拟次数n_p达到了一定的阈值T，该节点p就会停止随机选择子节点，并且进行剪枝只留下 k_{init} 个启发式值最高的子节点
- 但是当 $n_p > A \times B^{k-k_{init}}$ 时，节点 k 又会“变成”未剪枝状态？？？

except k_{init} children that has highest heuristic values in ascending order. To make heuristic values, the heuristic function is called and it causes computing time cost. After that, when the sum of simulation in node p exceeds $A \times B^{k-k_{init}}$, the pruned node is unpruned progressively by according to order of k [1]. The value of A, B, k was set to 50, 1.3, 5 empirically by following the base form.

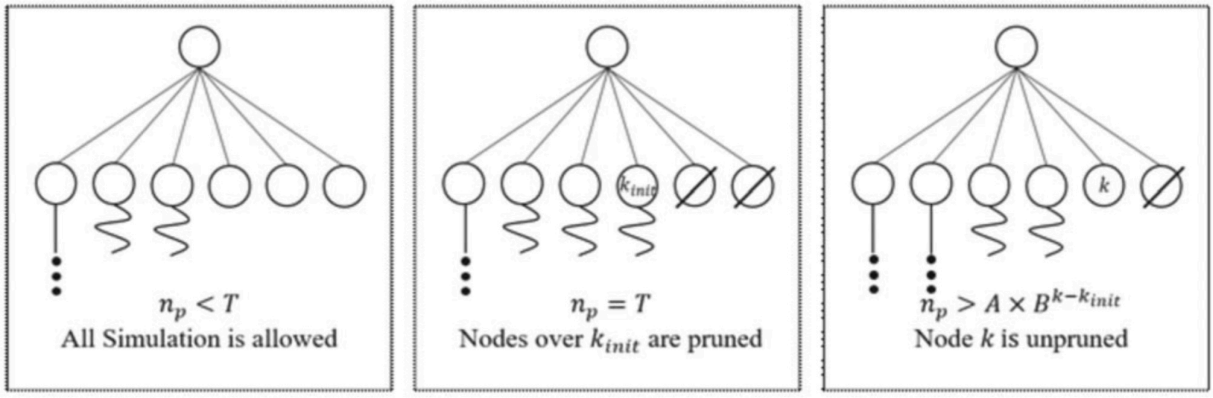

Fig. 4. Progressive unpruning

- 不同于GO的地方在于，这里的很多move的value都是0，所以每一次进行子节点选择的时候，只需要关注那些value不等于0的move就可以？？？

3.3 Heuristic knowledge

- 这里旨在计算前两个策略中的启发式值。
- 这里就简单地用一些线段长度的和来衡量离胜利条件有多近，如（3）所示。

$$H_i = \sum_l \left\{ (L_{open})^2 + \left(\frac{L_{hclosed}}{2} \right)^2 \right\} \tag{3}$$

- 其中L_{open}指的是两头都没有被堵住的线段长度，最多为5
- 其中$L_{hclosed}$指的是有且仅有一头被堵住的线段长度
- 这个公式的一些结果可以参照下表（来自论文K. L. Tan, C. H. Tan, K. C. Tan and A. Tay, “Adaptive game AI for Gomoku”, Proc. 4th Conf. Autonom. Robots Agents, pp. 507-512, 2009.）

Table 1. Threat table with heuristic value

Type of Threat	*Heuristic Value*
Half-Closed 2	1
Half-Closed 3	2.25
Open 2	4
Half-Closed 4	4
Open 3	9
Open 4	16
Closed 5 Half-Closed 5 Open 5	25

- 下表2中则列出了一些多威胁情况下的启发式值

Table 2. The threat case of multiple threats

Type of Threat	*Heuristic Value*
Half-Closed 4, Half-Closed 4	8
Open 2, Open 2, Open2	12
Open 2, Half-Closed 4, Half-Closed 4	12
Open 3, Half-Closed 4	13
Open 2, Open 2, Open2, Open 2	16
Open 2, Open 2, Open 3	17
Open 3, Open 3	18
Open 3, Open 4 Closed 5, Half-Closed 5, Open 5	25

3.4 Sequence-like Simulation with limited depth

- 在Gomoku中，模拟过程可以由local pattern来定义
- 在Gomoku中，获得胜利的最直接有效的方法就是形成double threats，这一点靠简单的局部竞争就可以达成（Go的复杂就在于它必须要关心全局的形势而不仅仅是局部）
- 对于深度= 1的普通树搜索来说，进行攻击，除非它需要进行防御
- 只需要一次威胁就可以找到隐藏的机会来制造双重威胁。当它发现双重威胁赢时返回+1，当它失败时返回-1
- 这样的模拟过程非常耗时，所以需要提前终止（深度受限）来减少时间
- 当模拟结果未能区分输赢时，将其视为平局并返回零点。但大多数模拟游戏最终都会因为有限的深度而返回平局，特别是游戏的早期阶段。为了补充这个问题，我们使用值小于1的启发式函数。如果没有值返回模拟结果，启发式值 H_i 可以代替 v_i 值
- 在渐进偏差公式（2）中，启发式值应该被归一化，以将 H_i 的变量范围中的值设置为等于 v_i。尽管大多数模拟结果回报因为有限的搜索深度而返回平局，但它可以通过依赖于启发式知识而将动作引导到适当的方式
- 由于它使用启发式和受限制的仿真策略进行大量仿真，因此很容易发现大部分仿真都产生了重复
- 所以，为了处理这个问题，这个AI选择时间效率而不是节省内存空间。 返回结果后，仿真进程不会丢弃，并且可以将其作为具有非整形条件的节点（如不可见节点到整棵树）附加到主树上。 当MCTS扩展一个节点并使用现有的模拟路径时，仿真算法重新使用它并继续在最后一个现有节点处的模拟。这种重复使用方法可以通过使用浅层深度模拟来制作简短的获胜路径的简单游戏。

4. 优化

- 之前的策略中有几个比较重要的参数：
  - 渐进未剪枝中的探索次数阈值 T
  - 合适的启发式值 H_i
  - 以及提前终止的模拟步数
- 这里探索次数阈值 T 的默认值以及相关参数为

  Tvalue is set to 50 andthe value of A, B, k in $\mathrm{A} \times \mathrm{B}^{k-k_{init}}$ was set to 50, 1.3, 5..
- 关于提前终止的步数
  - 如果较少，则模型可以很快地探索整棵树，但是得到的结果会非常接近那些默认的（<1）启发式值
  - 如果较多，则模型会花更多时间在模拟过程，但是会有更大的概率得到的有用的信息
  - 反正就是要多试试= =。

--END--