蒙特卡洛模拟是当前比较流行的围棋 AI 方法，CrazyStone 是最为著名的根据随机模拟而不是围棋技巧下棋的 AI。[1]中所使用的蒙特卡洛方法囊括了最基本的两个要点，一个是随机模拟棋盘，一种是结合蒙特卡洛评估与 minmax 树搜索，文中使用的树形结构与早先的 CrazyStone 保持一致。

MCTS 方法其实和传统的根据蒙特卡洛评估函数引导的树搜索是十分不一样的，更偏重于是一个根据 Mc 模拟引导的 best-first 搜索。主要的策略有两个方面，一个是选择策略，一个是模拟策略，这两者的平衡十分重要。

要理解这方面的问题，还是要从 k 臂赌博机问题说起，作为一个赌徒，要达到的目的就是通过迭代投注获得最大化回报，通常我们假设赌徒对先验知识是未知的，但通过重复实验，他可以集中在回报率高的赌博机上下注。所以其中比较重要的问题就是如何平衡，根据已知的知识获得最大化的回报，和通过新的尝试增加知识。exploitation-exploration dilemma 就是如此。

UCB 就是应用这样的思想的一种搜索，每次在选择下一扩展节点时，根据一掌握的知识，并权衡带来的增添的知识以选择下一节点。UCB-TUNED 是在 UCB 基础上，通过经验得出的启发式函数，在[1]中得到了应用。

UCT 是 UCB 在 minmax 树搜索上的扩展，基本思想在于，将每个节点都作为一个独立的赌博机，子节点是独立的赌博机臂。搜索通过在有限时间内，实验臂序列进行而非迭代处理单个节点。

UCT 比 alpha-beta search 要好，因为有时并不需要得到最优解，而只需得到较优解，毕竟在 Go 中数深度很大而且分支参数也很大，所以找到最优解往往很不现实。

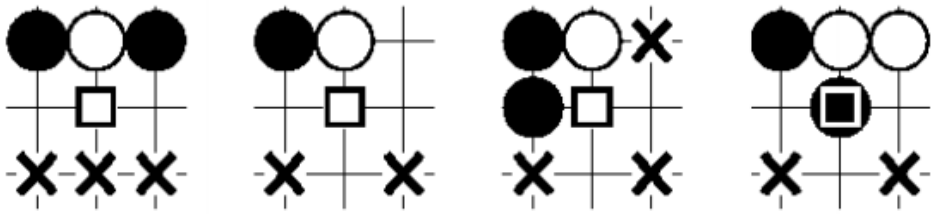
首先，在 UCT 中，树搜索可以停止于任意时刻，而且并不会过度影响其性能，其次 UCT 鲁棒性更强，因为计算得到子节点均值是根据遍历次数决定的，这个值是对最大值的平滑预测，取决于预测值的大小和这些预测值的置信度，如果一个子节点有相对较高的预测值，这个节点的被展开概率将会大于其他节点。最后，树不对称生成，因此回报率好的子节点拓展的更深。

UCT 的优点之一就是根据实际深度自动调整，对于树的每个分支，实际深度就是。。，这些分支的预测值将收敛的比其他快，UCT 会话费更多时间在其他有意思的分支上面。

通过添加领域知识，相比于原来的传统随机模拟来说，好处在于模拟看起来更具战略。在[1]中的 pattern 和其他 pattern 不同的是，pattern 仅被用于在找局部解当中，并不一定是全距最优解。因为在蒙特卡洛模拟中，找到一个更优的序列比找到更优的一步棋更为重要，如果将 pattern 应用于全部棋盘，反而会降低准确率。

我们仅仅考虑再最后一步棋附近匹配 pattern。因为局部最优解最有可能成为最后一步解。这样最后的局部序列就得到了。

首先辨认是否是 atari。如果是的话，就可以 saveing move 否则在最后的一步棋周围 8 个位置寻找 capturing stone，最后，随机下棋。



cross 代表无所谓

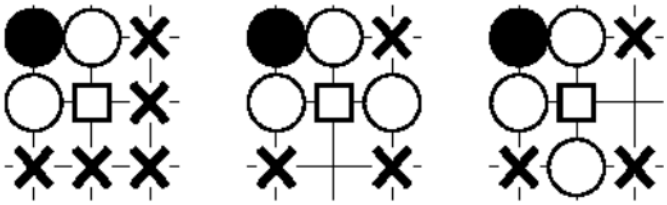


Figure 6: Patterns for Cut1. The Cut1 Move Pattern consists of three patterns. True is returned when the first pattern is matched and the next two are not matched.

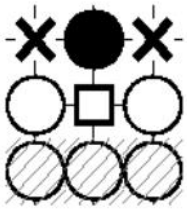


Figure 7: Pattern for Cut2. True is returned when the 6 upper positions are matched and the 3 bottom positions are not white.

Figure 8: Patterns for moves on the Go board side. True is returned if any pattern is matched. In the three right ones, a square on a black (resp. white) stone means true is returned if and only if the positions around are matched and it is black (resp. white) to play.

一个简单的 MC 树搜索。

[2]中提出了两种优化 UCT 的方法，一种是 RAVE 方法，通过共享搜索树的子树上动作的值来加快得到预估动作值的速度，为了避免 rave 的结果精确度不够，通过 MC-RAVE 综合两者的优点和缺点，从而得到综合最优的算法。

第二种方法是启发式 MC 树搜索，通过一个启发式函数来初始化新的节点的预估值

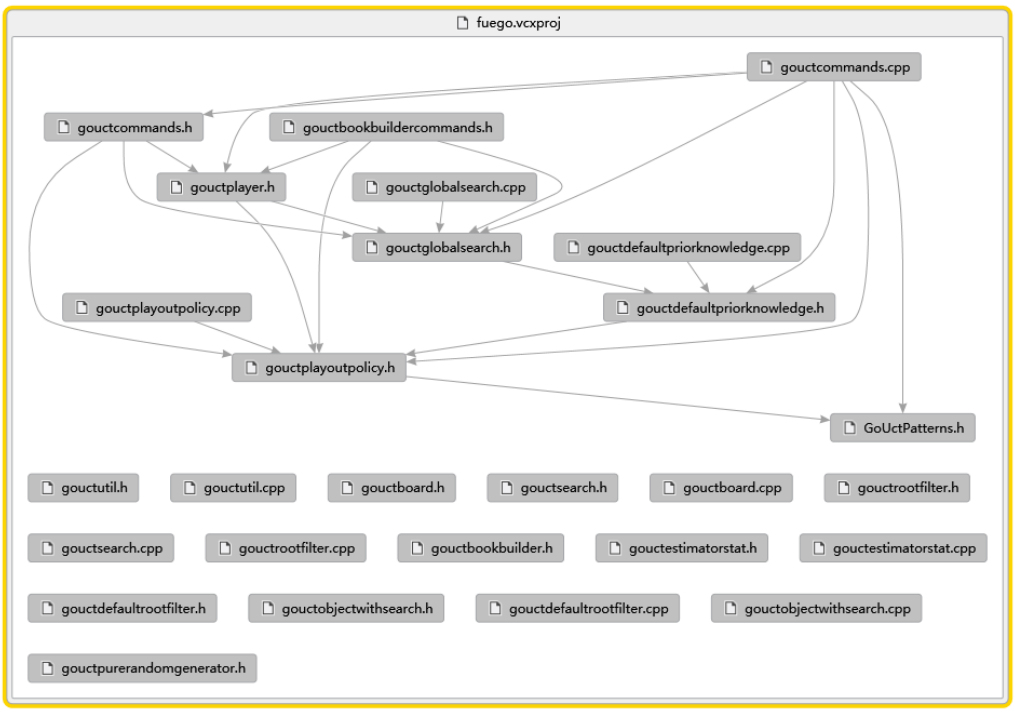
[1]3.3 提出了一种减少树的规模的 方法，从而更适应于更大的棋盘。首先根据群的定义来判断，这样就可以减少分支参数。另外区域分化可以从群理论中获得，从而得到一个更合理而精确的分数。

对于未访问过节点的拓展顺序，对于 UCT 来说，访问次数比较大的节点的分数肯定会更高，而模拟次数比较少的节点则会更偏重于拓展，因为一个位置上的任何一种可能步都会被利用，因此在更深处的节点的分数参考价值十分有限，对于这个问题有两个方法：
一个是 firstplay-urgency
UCB1 算法通过遍历每一个分支得到起始分数，由于尝试次数并不大，这种方法效率比较低，特别是在 Go 中节点数相对能仿真的次数很大，更是如此。比如如果一个 arm 一直返回 1，启示没有必要再去比较其他节点了。因此我们设置了一个固定常数 FPU，对于每一个步，我们根据下式赋值，

$$\bar{X}_j + \sqrt{\frac{\log n}{T_j(n)} \min\{1/4, V_j(T_j(n))\}}.$$

初始值都是无穷大，任何一个节点的 FPU 值都会不断更新，每一次我们都会选择 FPU 值最大的节点，较小的 FPU 保证了更早的 exploitation 如果第一次模拟效果并不好的话。
另外一个则是利用父亲节点的信息，直觉来说，在一种情景下好的一步棋在以后也很有可能是好的，其实我们可以利用祖父节点的预估值来预测这步棋的价值。

guego 的代码



关于并行化：

MCTS 的优点之一便是利于并行，分为叶并行，根并行，和树并行，在共享存储系统中，树并行可以充分利用通信带宽，不同线程并行化执行模拟，并共享内存中同一棵树。通常，我们需要一个全局所来保证对树的而修改不会冲突，但是 playout 阶段其实是相对独立不需要加锁的。所以树并行处理的优势和无锁阶段和总时间比值有关。
MANGO 中的树并行化处理在四个线程的时候具有很好的可扩展性，如果利用更细粒度的加锁算法，从而降低全局锁的开销，与此同时造成了每个节点局部锁导致的树节点规模的加大。这里主要的问题是，由于 MCTS 的选择特性，不同线程的 in-tree 操作中很多节点数据都是共享的。所以局部锁并不是很好的解决方法，但是他们的工作通过一个 virtual loss 提高了树并行的扩展特性。
fuego 通过全盘蒙塔卡罗模拟来实现，并做出了一些优化，生成一步棋的方法和 mogo 类似，通过使用 pattern、liberty 和 locality heuristics 优化。在 in-tree 步骤，子节点根据 RAVE 被选择，结合了节点的当前价值和此节点在相应节点子树中的平均价值。当扩展节点时，新节点的价值和数目根据静态启发式函数初始化。
无锁的多线程 MCTS
基本思想是在多线程中无锁共享一个树，对于硬件平台的要求导致这个方法或许不能大规模普及，但对 fuego 是很好的方法。
首先需要作出的改变是树结构的冲突修改问题，fuego 从不删除节点，新的节点被分配到预先分配的内存中，在无锁算法中，每个线程有自己的存储数组以创建新的节点。只有大昂所有节点都被创建并成功初始化之后，这些节点才会和父亲节点相连。这回导致一些存储开销，因为如果有几个节点扩展同一个节点的话，只有最后一个线程创建的节点会在将来的仿真中使用到。而且可能会导致有些节点的更新值会丢失。
每个节点的子节点信息包括，指向第一个子节点的指针，和子节点的数目。为了防止另一个线程看到不一致的状态，所有的线程都假设子节点的数目是有效的，只有当指向第一个节点的指针非空时。在连接一个父亲节点和子节点时，我们需要首先写入子节点的数目，然后指向第一个节点的指针。编译器通过声明这些变量是 volatile 来保证避免重排序这些写操作。
关于更新评估值
节点的 move 和 rave 评估值是通过计数和平均值存储的。平均值通过一个增加的算法来更新，如果通过无锁算法，会导致平均值的更新丢失，也就是计数失效。假设有个线程，读了计数同时被另一个线程进行了写操作，第一个线程将会看到一个错误的状态。在实际中，这个默认更新操作只在很小概率下发生而且对计数和平均值只有很小的影响，我们可以忽略这些影响。
比较大的问题在于如果计数是零的话，平均值没有定义，在很多情况下这个状态是有特殊意义的，比如说，在挑选子节点计算评估值的时候，分为以下三种情况考虑，如果评估计数和 rave 计数都是零，则采用 first play urgency 默认值，如果评估计数是零，则采用 rave 平均值，如果两个都不是零的话，则会采用两者的加权平均。
在 IA-32 和 Intel-64 的 cpu 架构是可行的。
在 0.3 版本的 fuego 中，每个核每秒可以跑 11400 次模拟【9*9】2750【19*19】
考虑并行化，剪枝的问题
Atari 只剩一口气， capture 提。

剪枝的问题
根据领土信息来减去特定的状态。围棋其实是对地盘的争夺，在白棋黑棋双方交替下棋时，争夺的就是特定地盘。根据围棋的规则，每下一步棋地盘的变动是有限的，我们用 hotpoint 来指定一些不会导致地盘大变动得点。Board B, P(B) 是它的父棋盘，S(B)是它的兄弟棋盘，

A 是棋盘上的所有位置，对于任意一个棋盘，地盘价值表示为：

$$T^B = \sum_{p \in B} T_p^B$$

利用 delta B 来表示棋盘领土价值的该变量，得到：

$$\Delta_B = T^{P(B)} - T^B$$

则期望的改变量为

$$E(\Delta_B) = \frac{\Delta_B}{A}$$

方差可计算得到：

$$D(\Delta_B) = E(\Delta_B^2) - E(\Delta_B)^2$$

最终得到评估函数为：

$$\Phi(B) = \frac{D(\Delta_B)}{\sum_{C \in S(B)} D(\Delta_C)}$$

衡量一个棋盘的稳定程度。

剪枝B_i条件为：

$$\exists\, B_j \ni S(B), B_j \neq B_i,$$
$$\phi(B_j) \notin [M, N] \; and \; \phi(B_j) \in [M, N], M \; and \; N \; are \; prior \; constants.$$

即表示B_j对更有利。

N 表示从当前节点开始的模拟总盘数，对于棋盘的特定点p 来说，$b_p(n)$为所有模拟中此点被黑子占领的次数，$w_p(n)$为被白子的次数，则

$$T_p^B = \frac{b_p(n)}{n} - \frac{w_p(n)}{n}$$

[1]modification of UCT with patterns

[2] Monte-Carlo tree search and rapid action value estimation in computer Go

Gelly, S., Wang, Y., Munos, R., & Teytaud, O. (2006). Modification of UCT with patterns in Monte-Carlo Go (Technical Report 6062). INRIA.

```
showboard
=
    A B C D E F G H J K L M N
13 X X X X X X X X X X X X 13
12 O O O O O O . . . . . X 12
11 O . . . . . . . . . . X 11
10 O . . . . . . . . . . X 10
 9 O . . . . . . . . . . X 9
 8 O . . . . . . . . . . X 8
 7 O . . . . . . . . . . X 7
 6 O . . . . . . . . . . X 6
 5 O . . . . . . . . . . . 5
 4 O . . . . . . . . . . . 4
 3 O . . . . . . O . . . . 3
 2 O . . . X . . . . . . . 2
 1 . O O O . . . . . . . . 1
    A B C D E F G H J K L M N
```

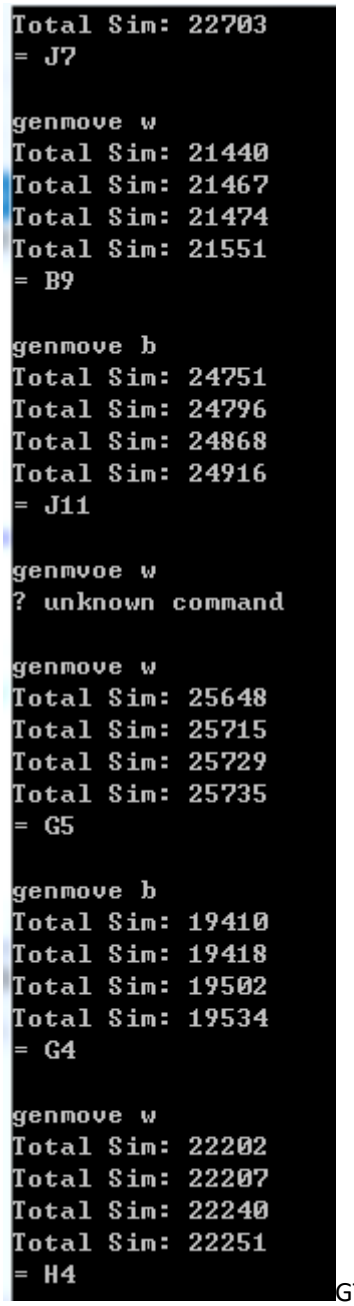
```
    A B C D E F G H J K L M N
13 O O O O O O O O O O O O O 13
12 X . . . . . . . . . . O O 12
11 X . . . . . . . . . . X O 11
10 X . . . . . . . . . . X O 10
 9 X . . . . . . . . . . X O 9
 8 X . . . . . . . . . . X O 8
 7 X . . . . . . . . . . X O 7
 6 X . . . . . . . . . . X O 6
 5 X . . . . . . . . . . X 5
 4 X . . . . . . . . . . . 4
 3 X . . . . . X . . . . . 3
 2 X . . . . . . . . . . . 2
 1 . X . . . . . . . . . . 1
    A B C D E F G H J K L M N
```

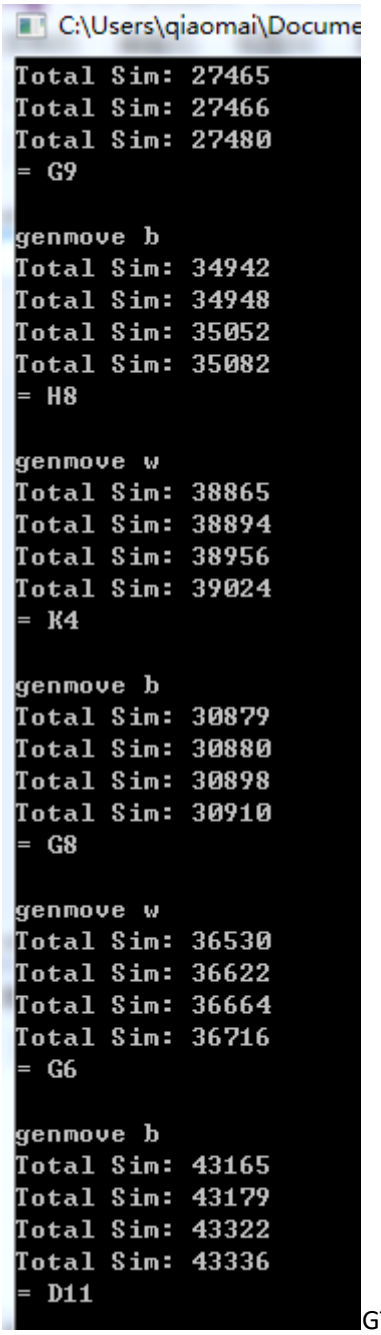
每次 raninit 出来的都很靠边。。

```
try pattern...4 succeed
pattern matched
= M11
```

```
   A B C D E F G H J K L M N
13 O O O O O O O O O O O O O 13
12 X X X . . . . . . . . O O 12
11 X . . . . . . . . . . X O 11
10 X . . . . . . . . . . X O 10
```

Hane 3 matched


Total Sim: 22703
= J7
genmove w
Total Sim: 21440
Total Sim: 21467
Total Sim: 21474
Total Sim: 21551
= B9
genmove b
Total Sim: 24751
Total Sim: 24796
Total Sim: 24868
Total Sim: 24916
= J11
genmvoe w
? unknown command
genmove w
Total Sim: 25648
Total Sim: 25715
Total Sim: 25729
Total Sim: 25735
= G5
genmove b
Total Sim: 19410
Total Sim: 19418
Total Sim: 19502
Total Sim: 19534
= G4
genmove w
Total Sim: 22202
Total Sim: 22207
Total Sim: 22240
Total Sim: 22251
= H4
GTD_P


C:\Users\qiaomai\Docume
Total Sim: 27465
Total Sim: 27466
Total Sim: 27480
= G9
genmove b
Total Sim: 34942
Total Sim: 34948
Total Sim: 35052
Total Sim: 35082
= H8
genmove w
Total Sim: 38865
Total Sim: 38894
Total Sim: 38956
Total Sim: 39024
= K4
genmove b
Total Sim: 30879
Total Sim: 30880
Total Sim: 30898
Total Sim: 30910
= G8
genmove w
Total Sim: 36530
Total Sim: 36622
Total Sim: 36664
Total Sim: 36716
= G6
genmove b
Total Sim: 43165
Total Sim: 43179
Total Sim: 43322
Total Sim: 43336
= D11
GTD