# 搜索

## 什么是搜索

以可接受的成本，在所有可能的问题解决方案中找出一个最优或可行的解决方案。

问题就在于我们如何去搜索，搜索的策略是什么！

理想搜索算法自然是尽快找出最优解。 但是有效性和效率是相互矛盾的 。‘世间安得两全法，不负如来不负卿’，不可能对任意的方法能够又快又有最优解。

## 搜索算法的分类

-盲目搜索vs启发式搜索；

-全局搜索与局部搜索；

-确定性的和随机的;

## 通用的搜索框架

s:状态集

f:后继功能(或动作)的集合，用于转换状态

c:函数成本的集合(如果不考虑最优解，则可有可无)

I:初始状态集

G:目标状态的集合

状态空间对应于一个图，其中从初始状态到目标状态的路径是问题的解决方案

步骤1表示所有状态，其中初始状态和目标状态被标记。

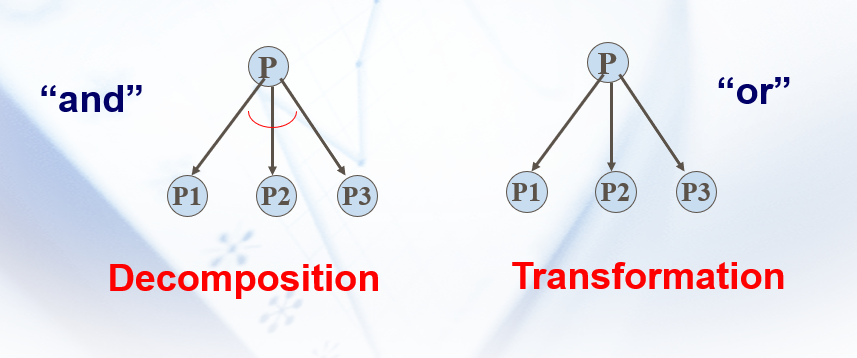
步骤2表示所有后续动作

步骤3以状态为节点，后继函数为边，得到相应的图

步骤4在图中搜索最优或可行的解(路径)

怎么表示后续的动作呢？？

通过解决等价或者分解成更简单的问题来解决难题。



第一步。将初始节点放入打开的表中

第二步。从打开的表中选择一个节点，并将其放入关闭的表中

第三步。展开选定的节点。如果该节点可以展开，则将所有展开的节点放入打开的表中。

第四步。重复步骤2-3，直到在步骤2中选择了目标节点，或者打开的表为空。

## 搜索算法举例

传统搜索——盲目搜索vs启发式搜索；

黑科技——

### ——盲目搜索

Breadth-first search (BFS)

广度优先

很好，一定能找到最优解，但是好慢呀

怎么办呢？

Depth-first search (DFS)

深度优先

也不错哦，比bfs快多了

Depth-limited search (DLS)

深度限制搜索

能不能不走完呀，dfs还是有一些慢呀，限制深度！如果已经搜很深了，然后还是没找到！爷不搜了！

Iterative deepening search (IDS)

Iterative depth-limited search with increased depth-limitation

一开始深度有限制，要是没搜到就再加深

### ——启发式搜索

不同搜索策略的关键区别在于节点扩展的顺序。

使用评估函数f(n)来估计每个节点的“合意性”，并扩展最合意的未扩展节点。

那评估函数的不同就是启发式搜索的不同了。

另外如果评估函数评估的代价，小于真实的代价，就一定能找到最优解。当然不要无限去小，尽量贴近真实的代价。

### ——基于max-min搜索的博弈树搜索

**max-min：**

min：把对手想象的很聪明，一定会让你取到min

max：自己很聪明，在所有的方法中会取 max 的

**α-β剪枝：**

剪枝+depth-limited search (DLS)策略

举个例子。

**Monte Carlo Tree Search (MCTS) 蒙德卡洛树搜索**

选择 Selection：从根节点 R 开始，递归选择最优的子节点（后面会解释）直到达到叶子节点 L。

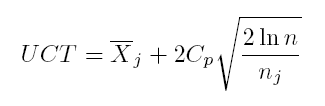
扩展 Expansion：如果 L 不是一个终止节点（也就是，不会导致博弈游戏终止）那么就创建一个或者更多的字子节点，选择其中一个 C。

模拟 Simulation：从 C 开始运行一个模拟的输出，直到博弈游戏结束。

反向传播 Backpropagation：用模拟的结果输出更新当前行动序列。

怎么去选择最优子节点呢？

**UCT**



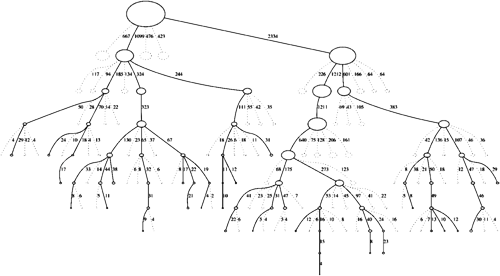
其实就是相当于代价函数

优点：

MCTS 提供了比传统树搜索更好的方法。

MCTS 不要求任何关于给定的领域策略或者具体实践知识来做出合理的决策。这个算法可以在没有任何关于博弈游戏除基本规则外的知识的情况下进行有效工作；这意味着一个简单的 MCTS 实现可以重用在很多的博弈游戏中，只需要进行微小的调整，所以这也使得 MCTS 是对于一般的博弈游戏的很好的方法。

MCTS 执行一种非对称的树的适应搜索空间拓扑结构的增长。这个算法会更频繁地访问更加有趣的节点，并聚焦其搜索时间在更加相关的树的部分。



非对称的增长

这使得 MCTS 更加适合那些有着更大的分支因子的博弈游戏，比如说 19X19 的围棋。这么大的组合空间会给标准的基于深度或者宽度的搜索方法带来问题，所以 MCTS 的适应性说明它（最终）可以找到那些更加优化的行动，并将搜索的工作聚焦在这些部分。

算法可以在任何时间终止，并返回当前最有的估计。当前构造出来的搜索树可以被丢弃或者供后续重用。

算法实现非常方便

缺点：

MCTS 算法，根据其基本形式，在某些甚至不是很大的博弈游戏中在可承受的时间内也不能够找到最好的行动方式。这基本上是由于组合步的空间的全部大小所致，关键节点并不能够访问足够多的次数来给出合理的估计。

MCTS 搜索可能需要足够多的迭代才能收敛到一个很好的解上，这也是更加一般的难以优化的应用上的问题。例如，最佳的围棋程序可能需要百万次的交战和领域最佳和强化才能得到专家级的行动方案，而最有的 GGP 实现对更加复杂的博弈游戏可能也就只要每秒钟数十次（领域无关的）交战。对可承受的行动时间，这样的 GGP 可能很少有时间访问到每个合理的行动，所以这样的情形也不大可能出现表现非常好的搜索。

幸运的是，算法的性能可以通过一些技术显著提升。

### ——黑科技

更高级的启发式算法——基于人工智能的思想的搜索算法

——进化计算 和 群智能

#### 进化计算

-DNA计算

将问题转化为了DNA序列的问题，先编码，然后通过碱基互补配对，让他们自由的组合。

它是利用DNA（ [脱氧核糖核酸](http://baike.baidu.com/view/23560.htm)）建立的一种完整的信息技术形式，以编码的DNA序列（通常意义上计算机内存）为运算对象，通过分子生物学的运算操作以解决复杂的 [数学难题](http://baike.baidu.com/view/1053667.htm)。

(1)具有高度的并行性,运算速度快,在一周的运算量相当于所有电子计算机从问世以来的总运算量.

(2)DNA作为信息的载体其贮存的容量非常之大,1m3的DNA溶液可存储1万亿亿的二进制数据,远远超过当前全球所有电子计算机的总存储量.

(3)DNA计算机所消耗的能量只占一台电子计算机完成同样计算所消耗的能量的十亿分之一.

(4)DNA分子的资源丰富.

-进化算法

其实也是满足框架的！

从初始的种群（从初始解开始）

通过基因重组和基因突变（下一状态）

自然选择（选择方法）

不同的算法，区别在于，是否基因重组和基因突变都有

以及自然选择的方法

如 选择最好的

如 按照概率的

如 随机pk，选择好的

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **GA** | **EP** | **ES** | **GP** |
| Represent-ation | **Genotype** | **Phenotype** | **Phenotype** | **Genotype** |
| Mutation | **√** | **√** | **√** | **√** |
| Recombin-ation | **√** | **×** | **√** | **√** |
| Parent Selection | **Probabilist-ic** | **Determini-stic** | **Probabilist-ic** | **Probabilist-ic** |
| Survivor Selection | **Extinctive,Determini-stic** | **Combined,Probabilist-ic** | **Combined,Determini-stic** | **Extinctive,Determini-stic** |

#### 群智能

glowworm swarm optimization(GSO)萤火虫群优化

spotted hyena optimization (SHO) 斑点鬣狗优化（SHO）

举例：蚁群算法和 粒子群算法

蚁群算法的基本原理:

1、蚂蚁在路径上释放信息素。

2、碰到还没走过的路口，就随机挑选一条路走。同时，释放与路径长度有关的信息素。

3、信息素浓度与路径长度成反比。后来的蚂蚁再次碰到该路口时，就选择信息素浓度较高路径。

4、最优路径上的信息素浓度越来越大。

5、最终蚁群找到最优寻食路径。

基本蚁群的两个过程:

(1)状态转移

(2)信息素更新



(2)信息素更新模型

蚁周模型（Ant-Cycle模型）

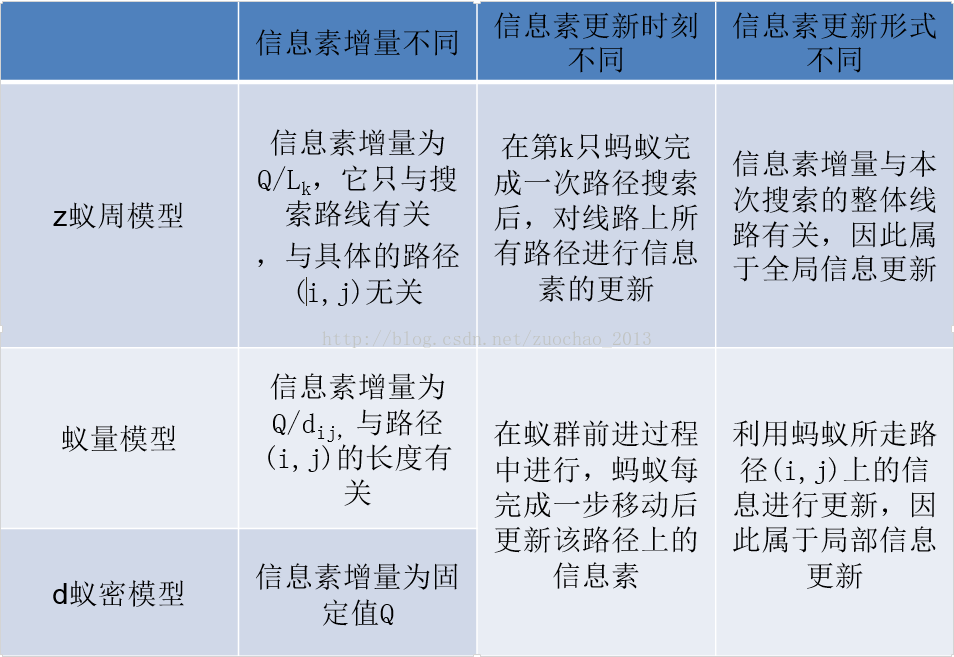
蚁量模型（Ant-Quantity模型）

蚁密模型（Ant-Density模型）

区别：

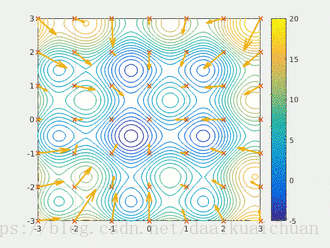
1.蚁周模型利用的是全局信息，即蚂蚁完成一个循环后更新所有路径上的信息素；

2.蚁量和蚁密模型利用的是局部信息，即蚂蚁完成一步后更新路径上的信息素。

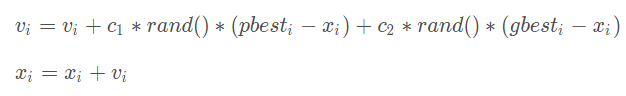


粒子群：

粒子群算法通过设计一种无质量的粒子来模拟鸟群中的鸟，粒子仅具有两个属性：速度和位置，速度代表移动的快慢，位置代表移动的方向。每个粒子在搜索空间中单独的搜寻最优解，并将其记为当前个体极值，并将个体极值与整个粒子群里的其他粒子共享，找到最优的那个个体极值作为整个粒子群的当前全局最优解，粒子群中的所有粒子根据自己找到的当前个体极值和整个粒子群共享的当前全局最优解来调整自己的速度和位置。下面的动图很形象地展示了PSO算法的过程：



PSO初始化为一群随机粒子(随机解)。然后通过迭代找到最优解。在每一次的迭代中，粒子通过跟踪两个“极值”(pbest，gbest)来更新自己。在找到这两个最优值后，粒子通过下面的公式来更新自己的速度和位置。

而粒子群优化算法的重要手段就在粒子位置和速度的更新上，通过不断改变粒子速度从而改变粒子位置让粒子群更加接近最优解的位置。速度的更新与位置的更新都有具体的公式：

其中i表示粒子的id，rand()是生成(0,1)的随机数，v和x表示粒子的速度和位置，c1和c2为自我学习因子和社会学习因子，pbest为粒子自己的最佳历史位置，gbest是社会中粒子的最佳历史位置。本文中社会代之整个种群，gbest指的就是整个种群中最好的个体位置。

