



西安交通大学
XI'AN JIAOTONG UNIVERSITY

人工智能技术导论

第三章 搜索与群体智能

目录

3.1 图搜索策略

3.2 盲目搜索

3.3 启发式搜索

3.4 博弈搜索

3.5 群智能算法

3.6 其它优化算法

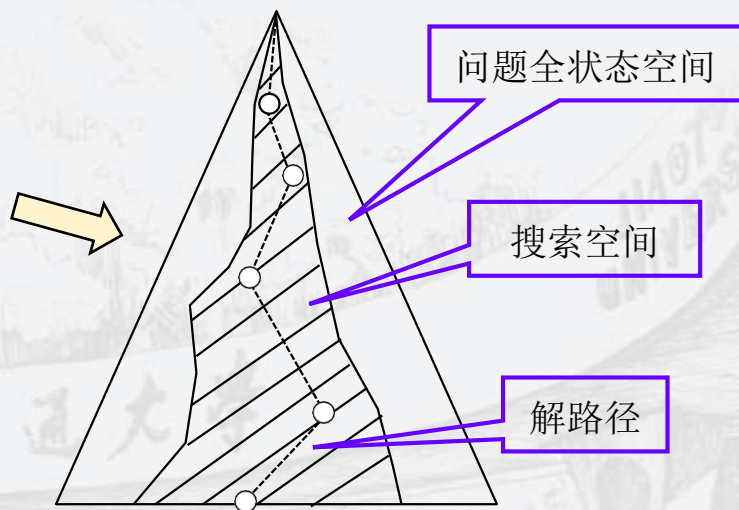




问题举例：传教士和野人问题

有3个传教士和3个野人来到河边准备渡河，河岸有一条船，每次至多可供2人乘渡。为了安全起见，传教士应如何规划摆渡方案，使任何时刻在河的两岸以及船上的野人数目总是不超过传教士的数目（但允许在河的一岸或者船上只有野人没有传教士）。

搜索空间示意图直观的展示了如何在一个比较大的问题空间中，只通过搜索比较小的范围就找到问题的解。使用不同的搜索，找到解的搜索空间范围是有区别。



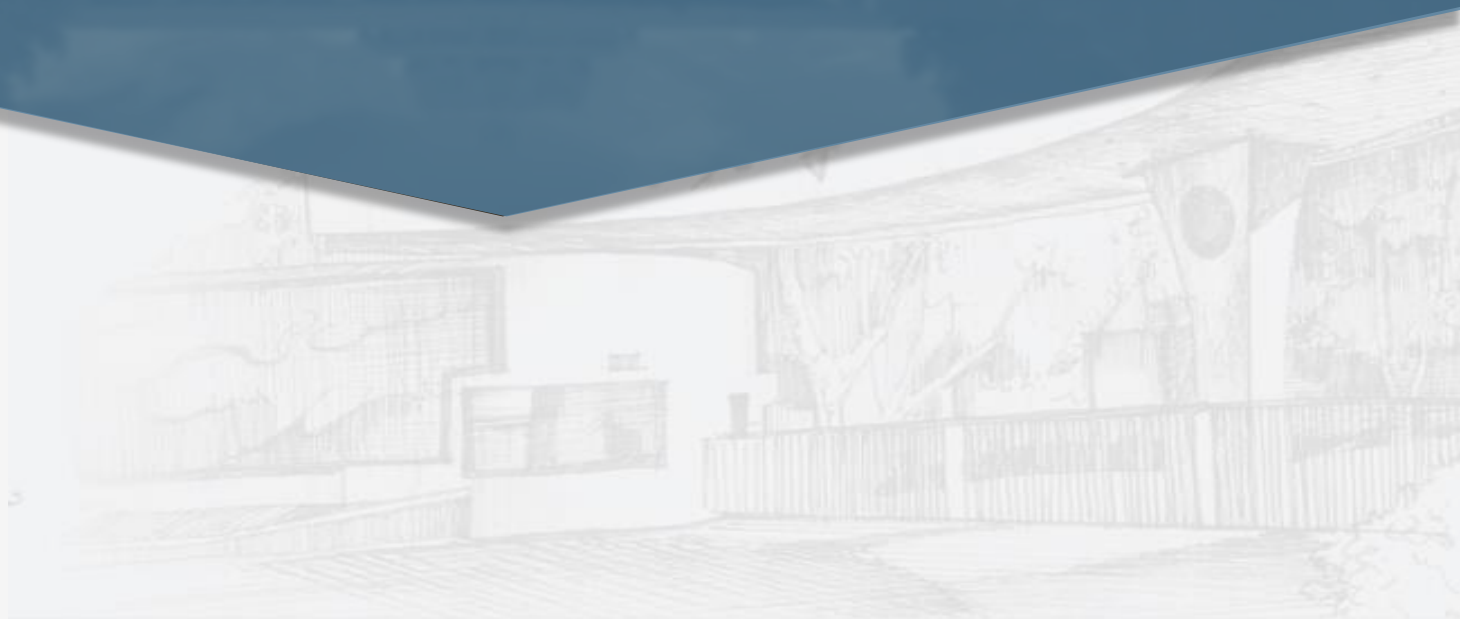
搜索策略的主要任务如何选取适当的方式：

- 盲目搜索：不考虑给定问题所具有的特定知识，系统根据事先确定好的某种固定排序，依次调用规则或随机调用规则。
- 启发式搜索：考虑问题领域可应用的知识，动态地确定规则的排序，优先调用较合适的规则使用。



西安交通大学
XI'AN JIAOTONG UNIVERSITY

3.1 图搜索策略





图搜索策略



西安交通大学
XI'AN JIAOTONG UNIVERSITY

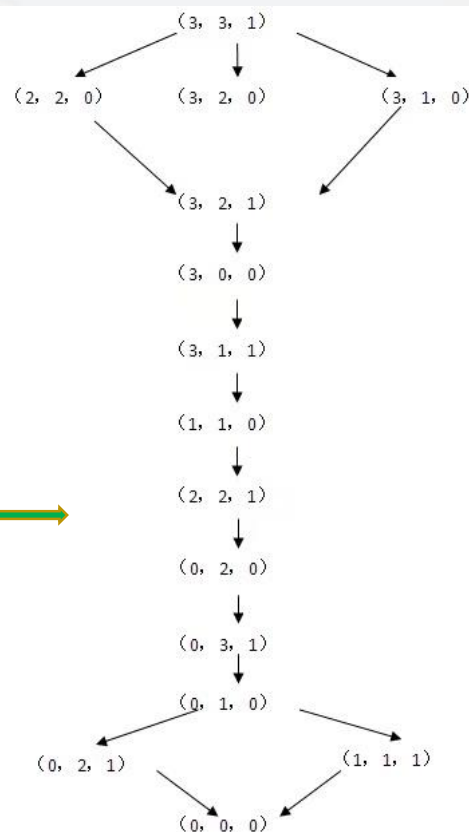
图搜索策略可看作一种在图中寻找路径的方法。初始节点和目标节点分别代表初始数据库和满足终止条件的数据库。求得把一个数据库变换为另一数据库的规则序列问题就等价于求得图中的一条路径问题。很多搜索问题都可以转化为图搜索问题。

思考：前面的传教士和野人问题如何转化为图搜索问题？

假设**初始状态传教士、野人和船均在河的左岸**，**目标是在满足问题的约束条件下到达河的右岸**。如果用在河的左岸的传教士、野人人数以及船是否在左岸表示一个状态，则该问题任何时刻状态都可以用一个**三元组表示(M, C, B)**，其中M、C分别表示在左岸的传教士、野人人数，B表示船是否在左岸，B=1表示船在左岸，B=0表示船在右岸。

则该问题初始状态为(3, 3, 1)，目标状态为(0, 0, 0)。所有满足约束条件的状态之间，构成了状态图。

通过该图找出一条从初始状态(3, 3, 1)到目标状态(0, 0, 0)的路径，就是图搜索问题所谓的路径，就是给出一个状态序列，序列的第一个状态是初始状态，最后一个状态是目标状态，序列中任意两个相邻的状态之间通过一个连线连接。

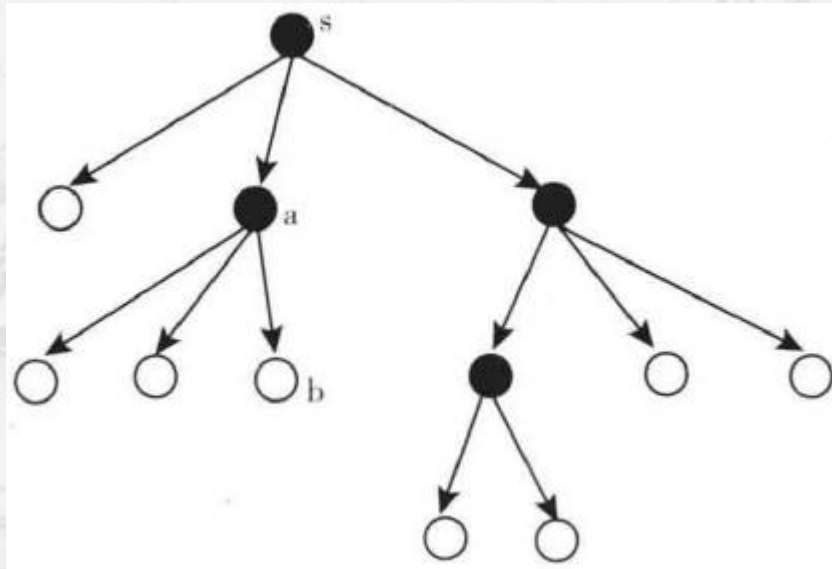




图搜索策略的效率

- ◆ 为了提高搜索效率，图搜索并不是在生成所有状态的连接图后再进行搜索，而是边搜索边生成图，直到找到一个符合条件的解，即路径为止。
- ◆ 在搜索的过程中，生成的无用状态越少，即非路径上的状态越少，搜索的效率就越高，所对应的搜索策略就越好。

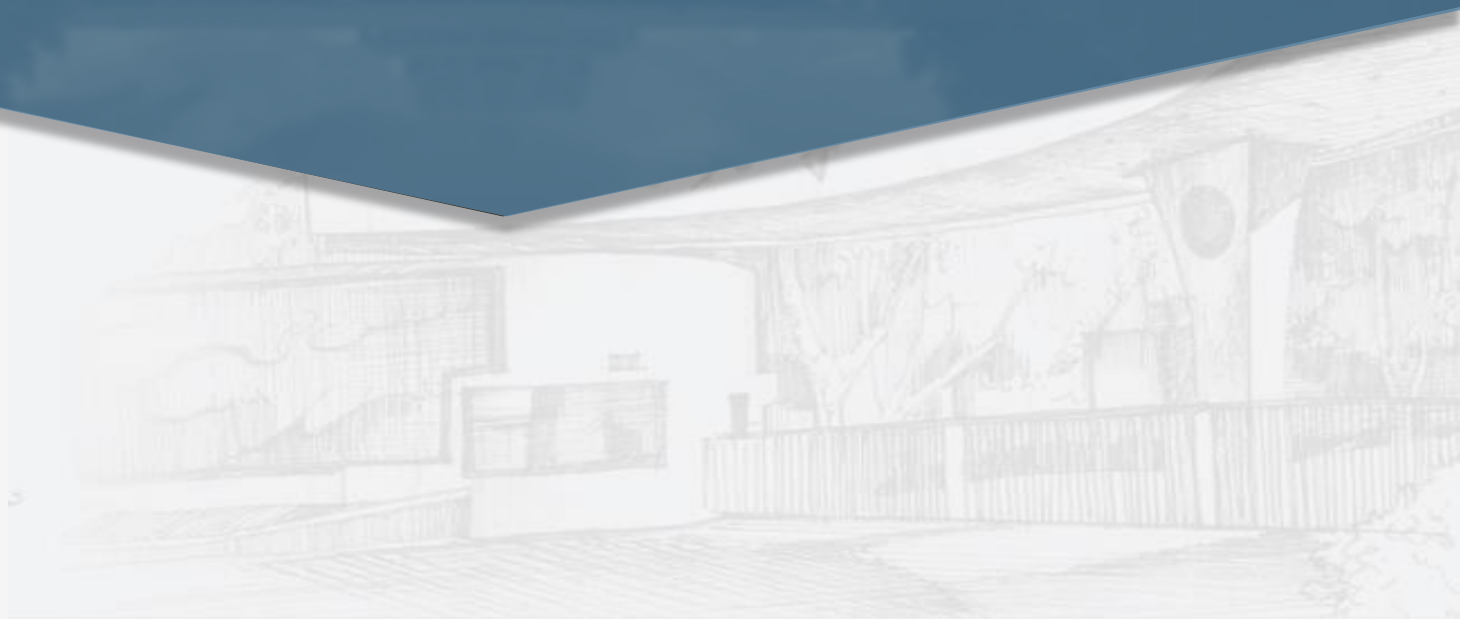
一般地，假设某个搜索过程中间情况如图所示。图中节点表示状态，实心圆表示已经扩展的节点（即已经生成了连接该节点的所有后继节点），空心圆表示还没有被扩展的节点。所谓的图搜索策略，就是如何从叶节点（空心圆）中选择一个节点扩展，以便尽快地找到一条符合条件的路径。不同的选择方法就构成了不同的图搜索策略。如果在选择节点时利用了与问题相关的知识或者启发信息，则称之为启发式搜索，否则就称之为盲目搜索。





西安交通大学
XI'AN JIAOTONG UNIVERSITY

3.2 盲目搜索





盲目搜索



西安交通大学
XI'AN JIAOTONG UNIVERSITY

如果在搜索过程中没有利用任何与问题有关的知识或者启发信息，称之为盲目搜索。深度优先和宽度优先是常用的两种盲目搜索方法。

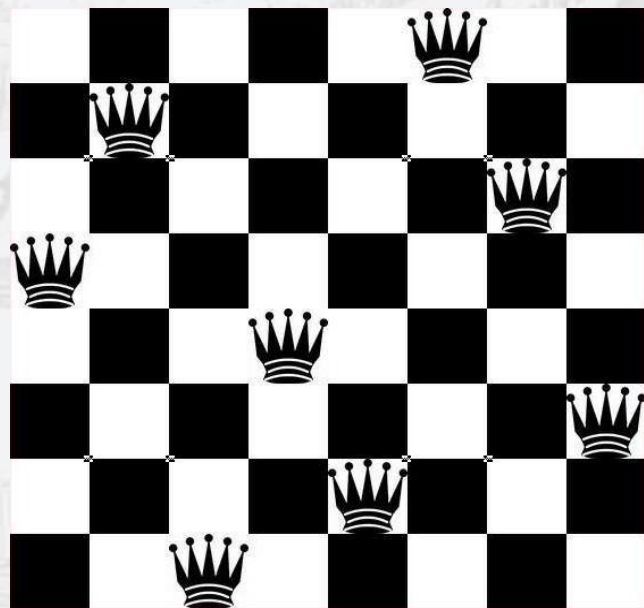
1. 深度优先搜索

深度优先搜索的基本思想是**优先扩展深度最深的节点**。深度优先搜索每次选择一个深度最深的节点进行扩展，如果有相同深度的多个节点，则按照事先的约定从中选择一个。如果该节点没有子节点，则选择一个除了该节点以外的深度最深的节点进行扩展。依次进行下去，直到找到问题的解结束；或者再也没有节点可扩展结束，这种情况下表示没有找到问题的解。

以 N 皇后问题为例，介绍深度搜索策略的过程。

例：皇后问题

N皇后问题：在一个 $N \times N$ 的国际象棋棋盘上摆放N枚皇后棋子，要满足每行、每列和每个对角线上只允许出现一枚皇后，即棋子间不许相互俘获。（为了简单说明，以4皇后问题为例）





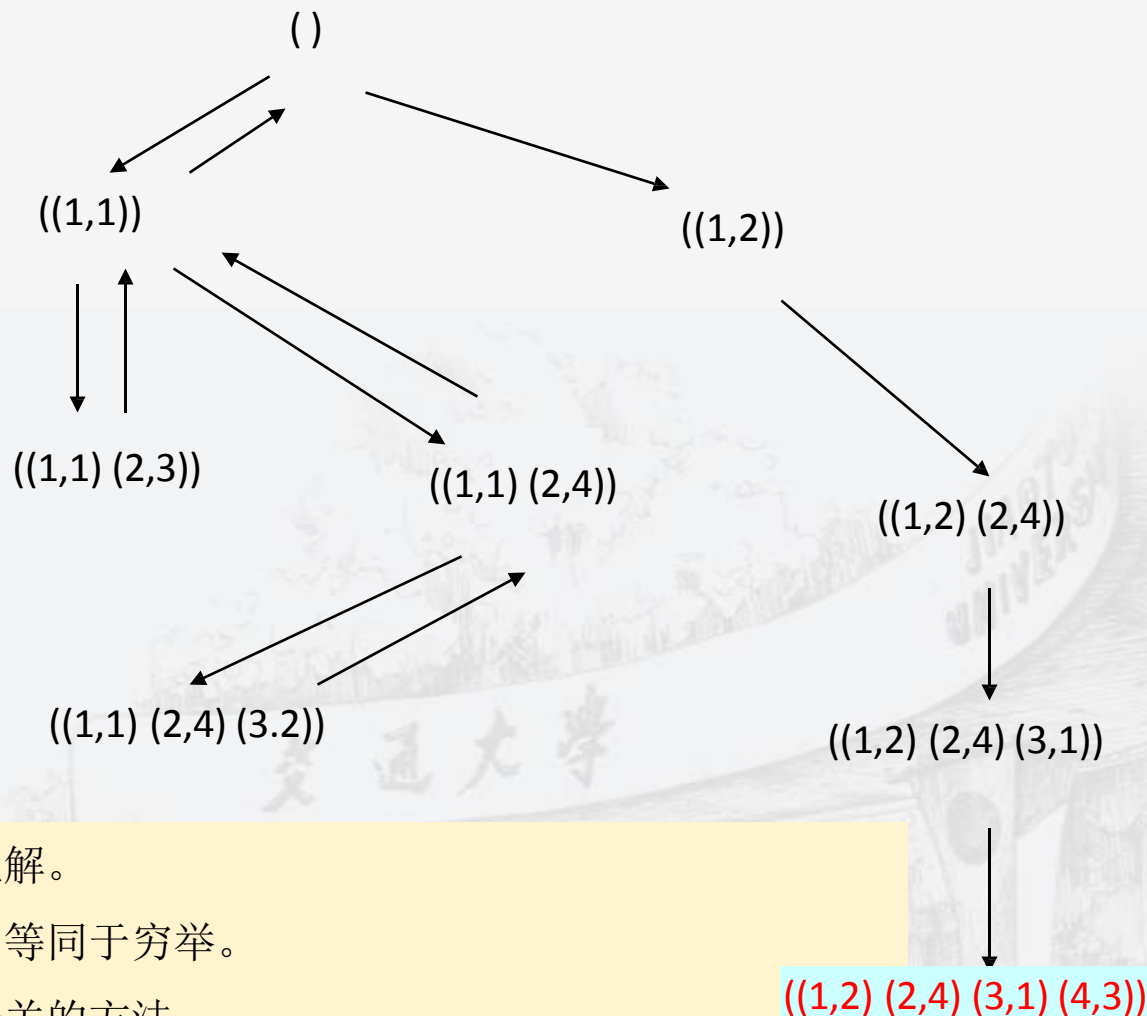
盲目搜索



西安交通大学
XI'AN JIAOTONG UNIVERSITY

例：皇后问题

	Q		
			Q
Q			
		Q	



深度优先搜索的性质

- ◆ 一般不能保证找到最优解。
- ◆ 最坏情况时，搜索空间等同于穷举。
- ◆ 是一个通用的与问题无关的方法。
- ◆ 节省内存，只存储从初始节点到当前节点的路径。
- ◆ 当深度限制不合理时，可能找不到解，可以将算法改为可变深度限制。

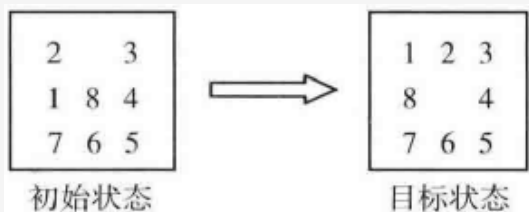


盲目搜索

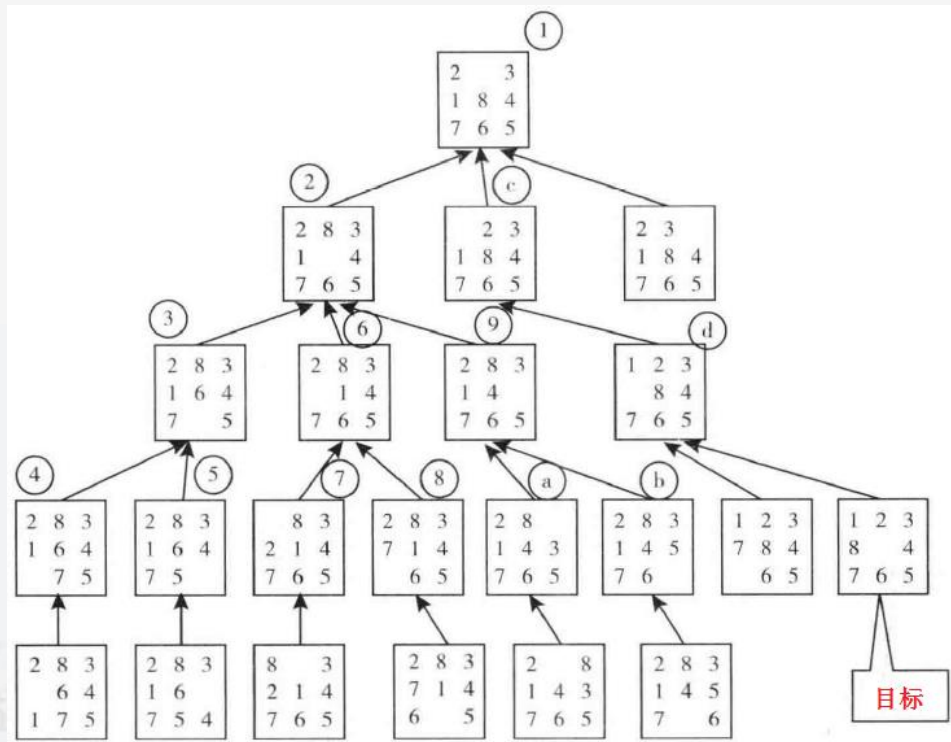


西安交通大学
XI'AN JIAOTONG UNIVERSITY

例：八数码问题



运用带有深度限制的深度优先方法求解八数码问题的搜索图，其中深度限制设置为4。图中圆圈中序号表示节点的扩展顺序(到9之后，用a、b、c、d表示)，当达到深度限制之后，回溯到稍浅一层的节点继续搜索直至找到目标节点。



除了初始节点外，每个节点用箭头指向其父节点，当搜索到目标节点后，沿着箭头所指反向追踪到初始节点，即可得到问题的解答。

深度限制与具体的问题有关，需要根据经验设置一个合理值。如果深度限制过深，则影响求解效率；如果限制过浅，则可能导致找不到解。

深度优先搜索也可能遇到“死循环”问题，也就是沿着一个环路一直搜索下去。为了解决这个问题，可以在搜索过程中记录从初始节点到当前节点的路径，每扩展一个节点，就要检测该节点是否出现在这条路径上；如果发现在该路径上，则强制回溯，探索其他深度最深的节点。



盲目搜索



西安交通大学
XI'AN JIAOTONG UNIVERSITY

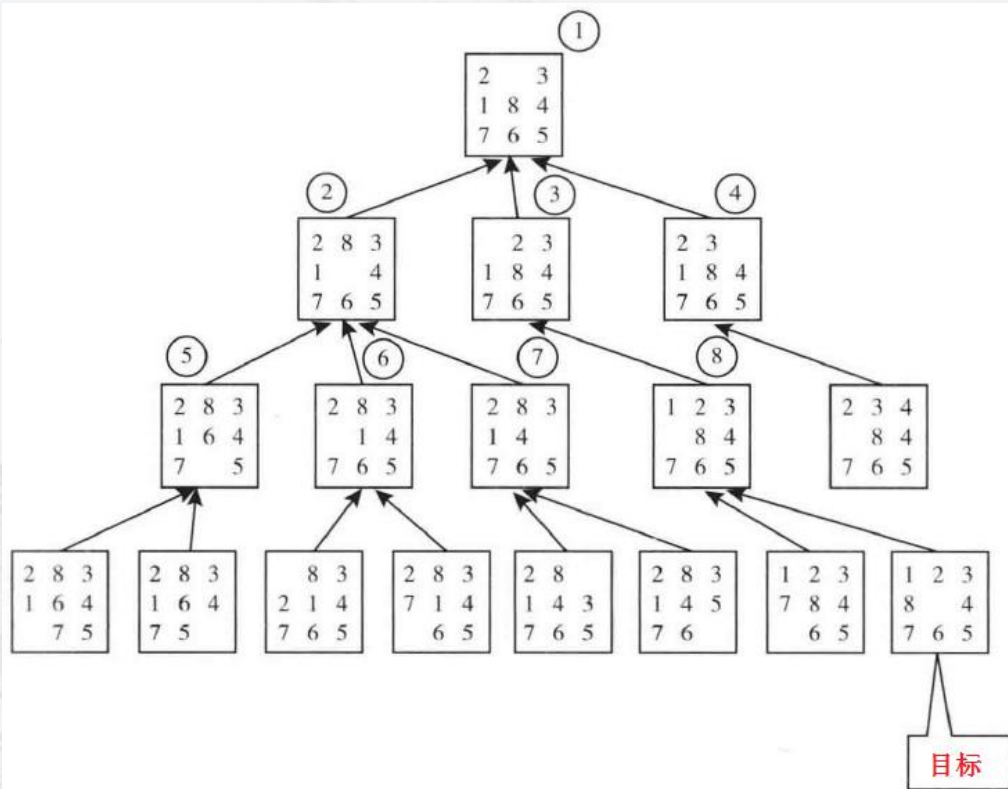
2. 宽度优先搜索

与深度优先策略刚好相反，宽度优先搜索策略是优先搜索深度浅的节点，即每次选择深度最浅的叶节点进行扩展，如果有深度相同的节点，则按照事先约定从深度最浅的几个节点中选择一个。同样是八数码问题，深度搜索策略的过程如图所示。图中同样用带有圆圈的数字给出了节点的扩展顺序，与深度优先搜索的“竖”着搜不同，宽度优先搜索体现的是“横”着搜。

宽度优先搜索与深度优先搜索有哪些不同呢？

可以证明，对于任何单步代价都相等的问题，在问题有解的情况下，**宽度优先搜索一定可以找到最优解。**

例如八数码问题，如果移动每个牌的代价都相同，则利用宽度优先找到的解一定是将牌移动次数最少的最优解。但是，由于宽度优先在搜索过程中需要保留已有的搜索结果，需要占用比较大的搜索空间，且会随着搜索深度成几何级数增加。深度优先虽不能保证找到最优解，但可采用回溯方法，只保留从初始节点到当前节点一条路径即可，大大节省存储空间，其所需要的存储空间只与搜索深度呈线性关系。

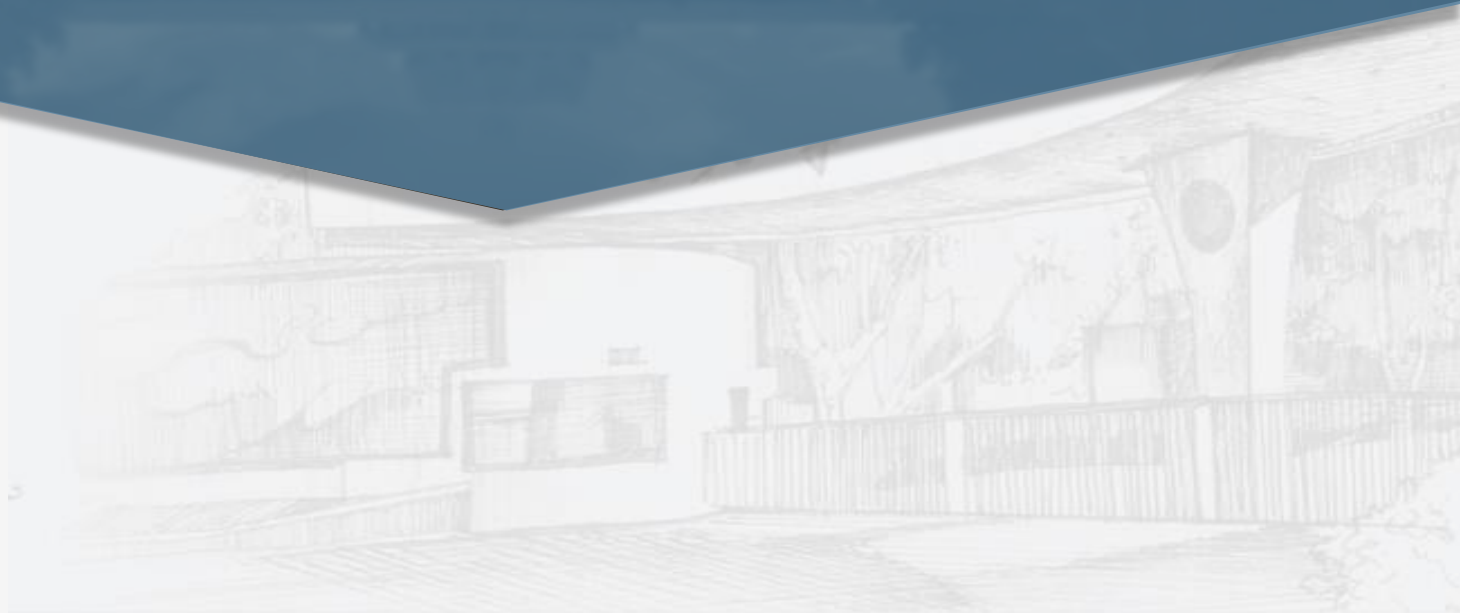




西安交通大学

XI'AN JIAOTONG UNIVERSITY

3.3 启发式搜索





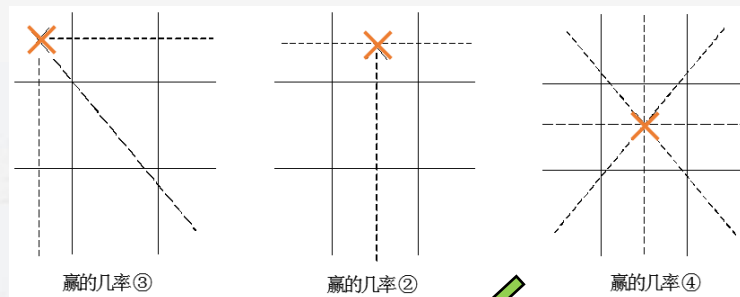
启发式搜索



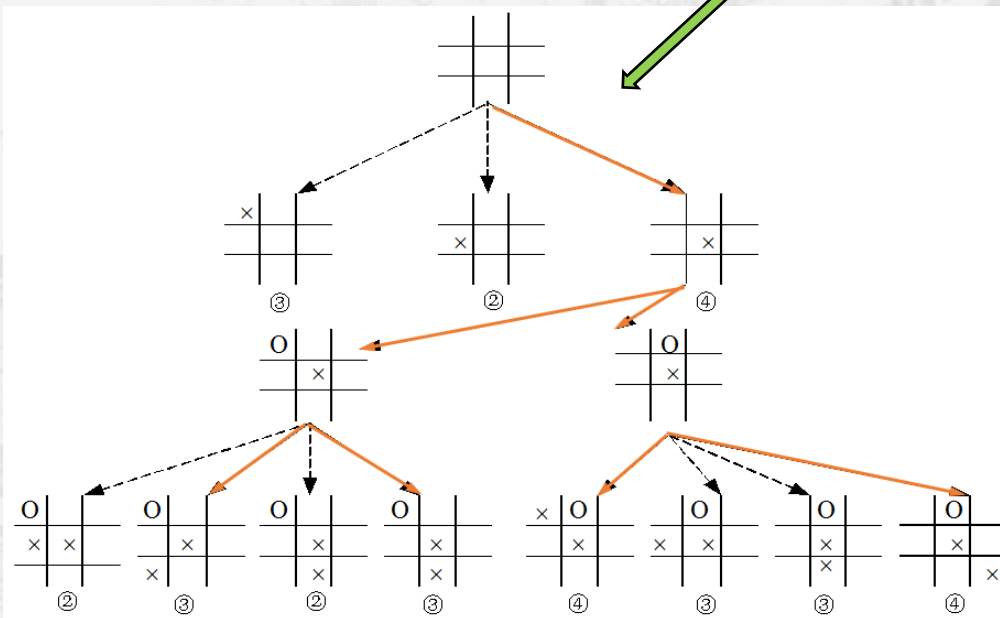
西安交通大学
XI'AN JIAOTONG UNIVERSITY

启发式搜索(Heuristically Search)又称为有信息搜索，它是利用问题拥有的启发信息来引导搜索，达到减少搜索范围、降低问题复杂度的目的，这种利用启发信息的搜索过程称为启发式搜索。

例：一字棋。在九宫棋盘上，从空棋盘开始，双方轮流在棋盘上摆各自的棋子 ×或O（每次一枚），谁先取得三子一线（一行、一列或一条对角线）的结果就取胜。



- ◆ ×和 O能够在棋盘上摆成的各种不同的棋局就是问题空间中的不同状态。
- ◆ 9个位置上摆放{空, ×, O} 有 39 种棋局。
- ◆ 可能的走法： $9 \times 8 \times 7 \times \dots \times 1$ 。





启发式搜索



西安交通大学
XI'AN JIAOTONG UNIVERSITY

启发式搜索希望引入启发知识，在保证找到最佳解的情况下，尽可能减少搜索范围、提高搜索效率。

- ◆ 求解问题系统不可能知道与实际问题有关的全部信息，因而无法知道该问题的全部状态空间，也不可能用一套算法来求解所有的问题。
- ◆ 有些问题在理论上虽然存在着求解算法，但是在工程实践中，这些算法不是效率太低，就是根本无法实现。（一字棋： $9!$ ，围棋： 10^{761} ）
- ◆ 启发信息的强度：
 - 强：降低搜索工作量，但可能导致找不到最优解。
 - 弱：导致工作量加大，极限情况变为盲目搜索，但可能找到最优解。
- ◆ 利用控制性的启发信息的情况：
 - 没有任何控制性知识作为搜索的依据，因而搜索的每一步完全是随意的。
 - 有充分的控制知识作依据，因而每一步选择都是正确的，但这不现实的。

基本思想：定义一个评价函数 f ，对当前的搜索状态进行评估，找出一个最有希望的节点来扩展。



启发式搜索



西安交通大学
XI'AN JIAOTONG UNIVERSITY

A算法

搜索过程如图所示，需要从所有的叶节点中选择一个节点扩展。为了尽快找到从初始节点到目标节点的一条耗散值比较小的路径。希望所选择的节点尽可能在最佳路径上。如何评价一个节点在最佳路径上的可能性呢？

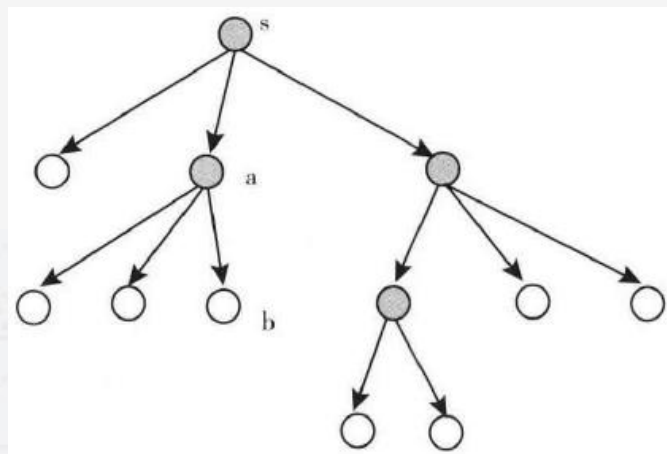
A算法给出了评价函数的定义：

$$f(n) = g(n) + h(n)$$

式中， n 为待评价的节点； $g(n)$ 为从初始节点 s 到节点 n 的最佳路径耗散值的估计值， $h(n)$ 为从节点 n 到目标节点 t 的最佳路径耗散值的估计值，**称为启发函数**； $f(n)$ 为从初始节点 s 经过节点 n 到达目标节点 t 的最佳路径耗散值的估计值，**称为评价函数**。

这里的耗散值指的是路径的代价，根据求解问题的不同，可以是路径的长度或是需要的时间，或是花费的费用等。如果 $f(n)$ 能够比较准确地估计出 s - n - t 这条路径的耗散值的话，我们每次从叶节点中选择一个 $f(n)$ 值最小的节点扩展，则有理由相信这样的搜索策略对于我们尽快搜索到一条从初始节点 s 到目标节点 t 的最佳路径是有意义的。

采用这种搜索策略的搜索算法，称之为A算法。





启发式搜索



西安交通大学
XI'AN JIAOTONG UNIVERSITY

A算法

实现A算法的关键是 $f(n)$ 的计算，其中 $g(n)$ 可以通过已有的搜索结果计算得到。如在图中节点b的 $g(n)$ 值可以通过s-a-b这条路径的耗散值计算得到，根据具体的问题， $g(n)$ 很容易计算得到。启发函数 $h(n)$ 则需要根据问题定义，同一个问题也可以定义出不同的函数，**如何定义一个好的启发函数成为用A算法求解问题的关键所在。**

- open表：保留已生成而未扩展的状态。
- closed表：记录已扩展过的状态。
- 进入open表的状态是根据其估值大小插入到表中合适位置，每次从表中优先取出启发函数值最小的状态加以扩展。

```
procedure heuristic_search
Open:=[start]; closed: =[ ]; f(s):=g(s)+h(s);           *初始化
while open≠[ ] do
begin
  从open表中删除第一个状态，称之为n;
  if n=目的状态then return(success);
  生成n的所有子状态;
  if n没有任何子状态then continue;
  for n的每个子状态do
  case子状态is not already on open表or closed表;
  begin
    计算该子状态的估价函数值;
    将该子状态加到open表中;
  end;
  case 子状态is already on open表:
  If 该子状态是沿着一条比在open表已有的更短路径而到达
  then 记录更短路径走向及其估价函数值;
  case子状态is already on closed表:
  If 该子状态是沿着一条比在closed表已有的更短路径而到达then
  begin
    将该子状态从closed表移到open表中;
    记录更短路径走向及其估价函数值;
  end;
  case end;
  将n放入closed表中;
  根据估价函数值，从小到大重新排列open表;
end;
return(failure);
end.
```



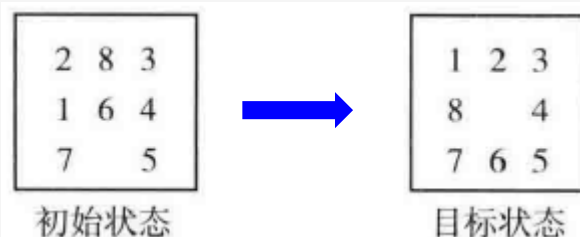
启发式搜索



西安交通大学
XI'AN JIAOTONG UNIVERSITY

A算法

例：八数码问题

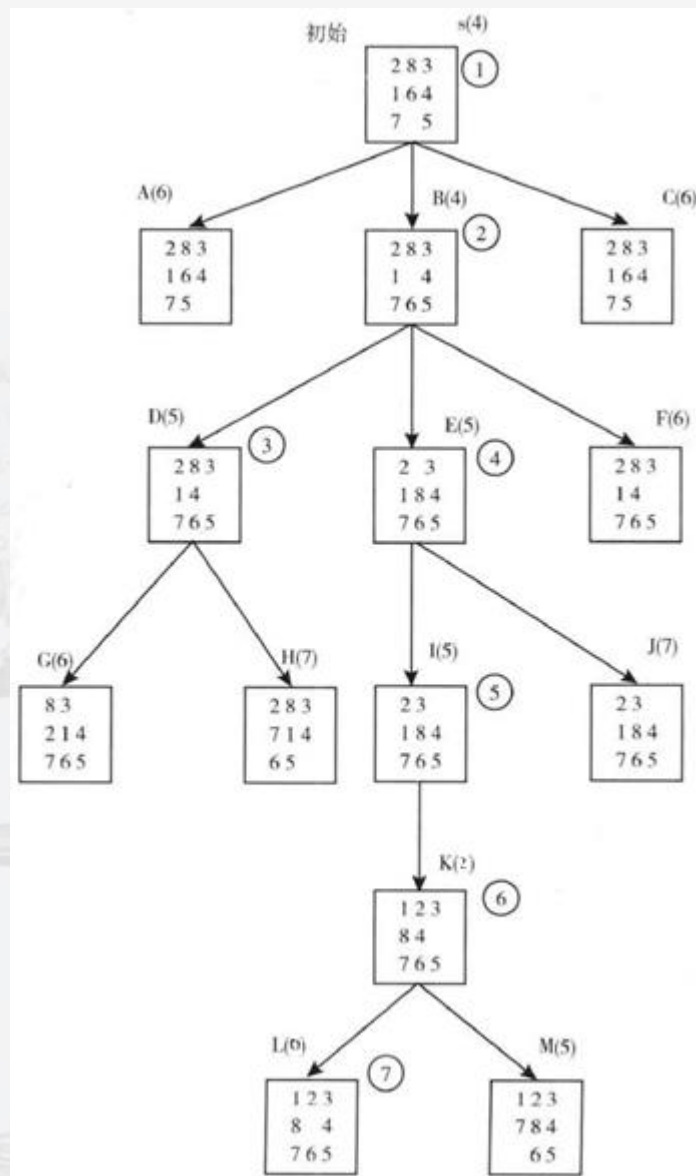


定义评价函数：

$$f(n) = g(n) + h(n)$$

式中 $g(n)$ 为从初始节点到当前节点的耗散值， $h(n)$ 为当前节点“不在位”的将牌数。其含义是将待评价的节点与目标节点进行比较，计算一共有几个将牌所在位置与目标是不一致的，而不在位的将牌个数的多少大体反映了该节点与目标节点的距离。

初始状态与目标状态进行比较，发现1、2、6、8四个将牌不在目标状态的位置上，所以初始状态的“不在位的将牌数”就是4，也就是初始状态的 h 值等于4。其它状态的 h 值也按照此方法计算。用A算法求解该八数码问题的搜索图如右图，图中字母后面的数字是该状态的 f 值，带圆圈的数字表示节点的扩展顺序。





启发式搜索



西安交通大学
XI'AN JIAOTONG UNIVERSITY

A*算法

在A算法中，由于并没有对启发函数做出任何规定，所以A算法得到的结果如何也不好评定。如果启发函数 $h(n)$ 满足如下条件：

$$h(n) \leq h^*(n)$$

则可以证明当问题有解时，A算法一定可以得到一个耗散值最小的结果，也即最佳解。满足该条件的A算法称作A*算法，如果某一问题有解，那么利用A*搜索算法对该问题进行搜索则一定能搜索到解，并且一定能搜索到最优的解而结束。

上例中的八数码A搜索树也是A*搜索树，所得的解路（s, B, E, I, K, L）为最优解路，其步数为5。

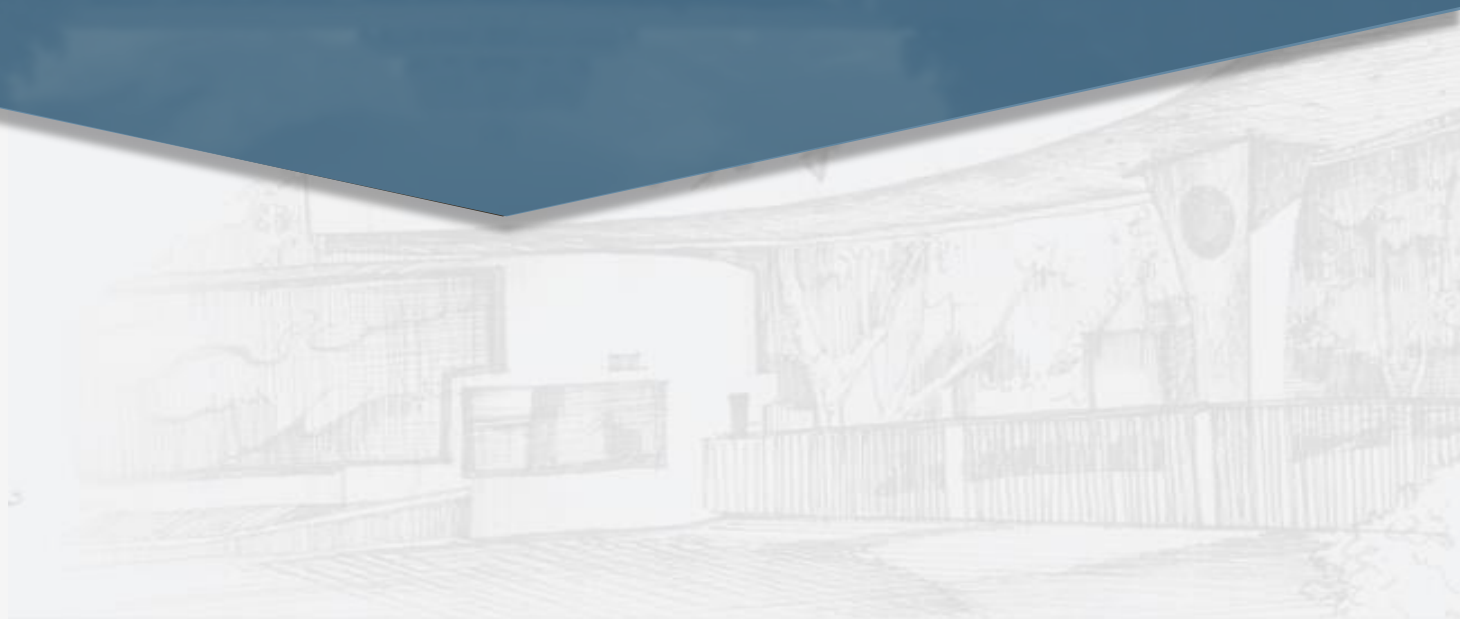
A*算法特点

- 可采纳性：当一个搜索算法在最短路径存在时能保证找到它，就称它是可采纳的。
- 单调性：在整个搜索空间都是局部可采纳的。一个状态和任一个子状态之间的差由该状态与其子状态之间的实际代价所限定。
- 信息性：在两个A*启发策略的 h_1 和 h_2 中，如果对搜索空间中的任一状态 n 都有 $h_1 \leq h_2$ ，就称策略 h_2 比 h_1 具有更多的信息性。



西安交通大学
XI'AN JIAOTONG UNIVERSITY

3.4 博弈搜索





博弈搜索



西安交通大学
XI'AN JIAOTONG UNIVERSITY

博弈本意是：下棋。引申义是：在一定条件下，遵守一定的规则，一个或几个拥有绝对理性思维的人或团队，从各自允许选择的行为或策略进行选择并加以实施，并从中各自取得相应结果或收益的过程。有时候也用作动词，特指对选择的行为或策略加以实施的过程。

博弈论，又称为对策论（Game Theory）、赛局理论等，既是现代数学的一个新分支，也是运筹学的一个重要学科。博弈论主要研究公式化了的激励结构间的相互作用，是研究具有斗争或竞争性质现象的数学理论和方法。博弈论考虑游戏中的个体的预测行为和实际行为，并研究它们的优化策略。

博弈搜索，多智能体参与的一种搜索方法。首先需定义搜索的状态空间图，即构建博弈树。搜索的规则是双方交替进行。为评估每次搜索的效果，引入一个评估函数。一个智能体的搜索目标是找到一个节点，使得评估函数值极大化；而另一个智能体的搜索目标则是使评估函数值极小化。计算机象棋、计算机围棋等使用的就是博弈搜索法。



博弈搜索

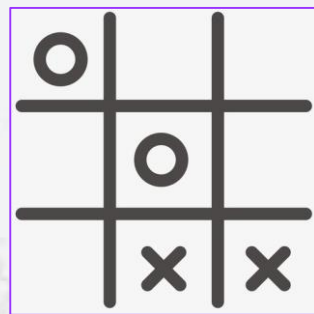


西安交通大学
XI'AN JIAOTONG UNIVERSITY

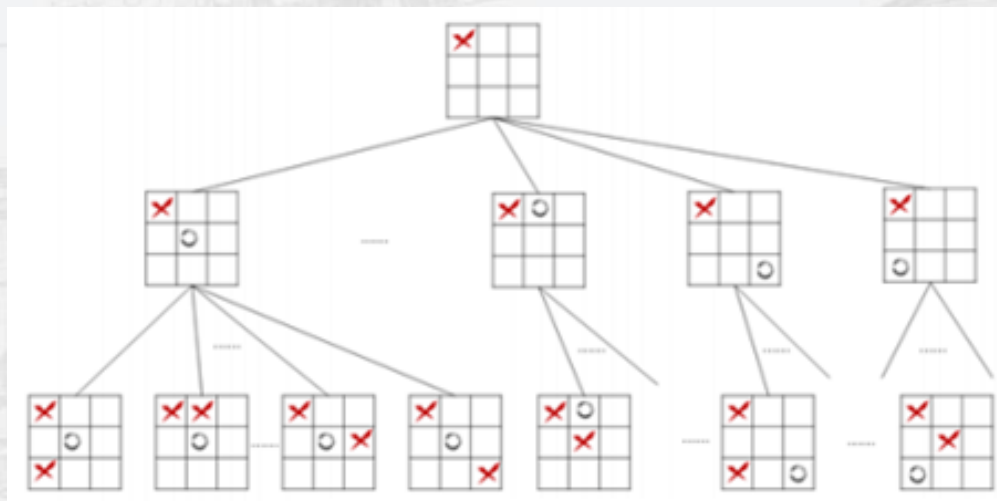
➤ α - β 剪枝搜索

由于在博弈问题中，搜索图（树）实在太大，A*搜索效率是很低的。在博弈搜索的算法中，剪枝允许我们在搜索树中忽略那些不影响最后决定的部分，启发式的评估函数允许在不进行完全搜索的情况下，估计某状态的真实效用值。

井字棋(Tic-Tac-Toe)是由两个玩家轮流在3X3的格子标记（圈或叉）的游戏，最先以横、直、斜连成一线则获胜。



在零和博弈中（各方收益和损失相加总和永远为“零”），玩家均会在可选的选项中选择将其N步后优势最大化或者令对手优势最小化。双方决策过程视作一颗决策树，一局游戏的发展过程是博弈树从根节点到叶结点的一条路径。井字棋的博弈树最高有9层，图所示井字棋博弈树前3层。





博弈搜索

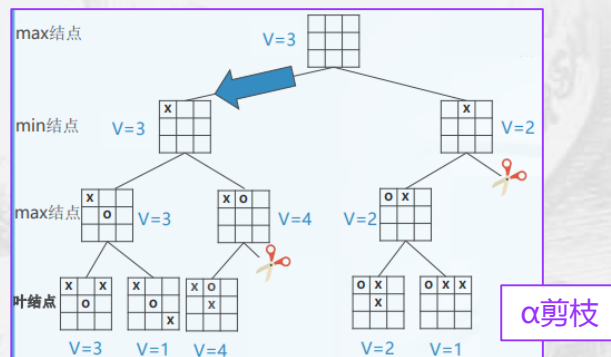
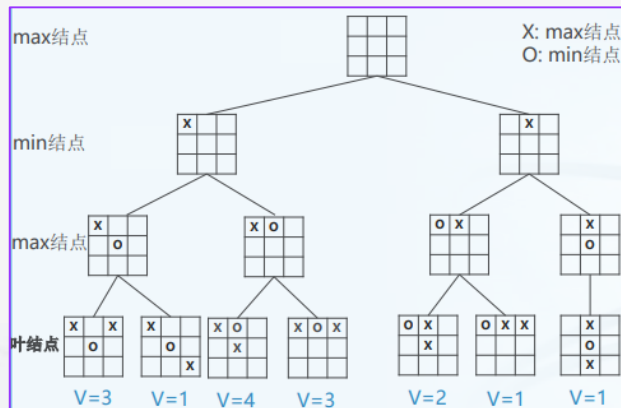
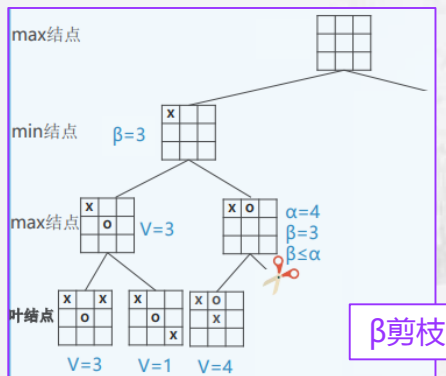
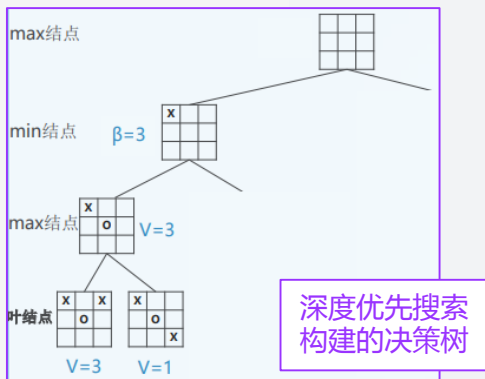


西安交通大学
XI'AN JIAOTONG UNIVERSITY

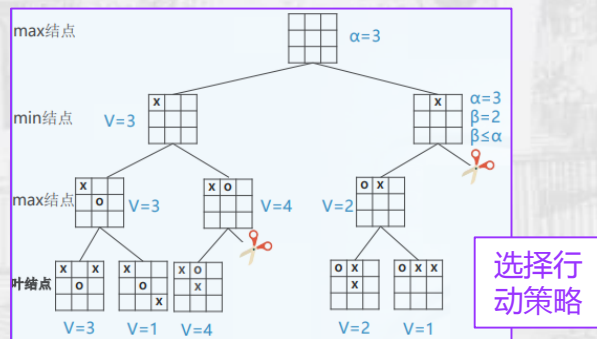
➤ α - β 剪枝搜索

算法过程可描述如下：

- 开始构建决策树；
- 将估值函数应用于叶子节点；
- 使用深度优先搜索序构建和搜索决策树，传递并更新 α 、 β 、节点minimax值
- 从根结点选择评估值最大的分支，作为行动策略。



有部分节点是否被搜索不会影响最后的结果，因此无需展开此类节点以及计算此类节点的子节点的估值。通过该方法，可节省算法的搜索时间。这种不展开搜索不必要节点的算法，被称为 α - β 剪枝搜索



引文链接: https://blog.csdn.net/qq_24178985/article/details/115834278
https://blog.csdn.net/qq_24178985/article/details/115858780



博弈搜索



西安交通大学
XI'AN JIAOTONG UNIVERSITY

➤ 蒙特卡洛树搜索

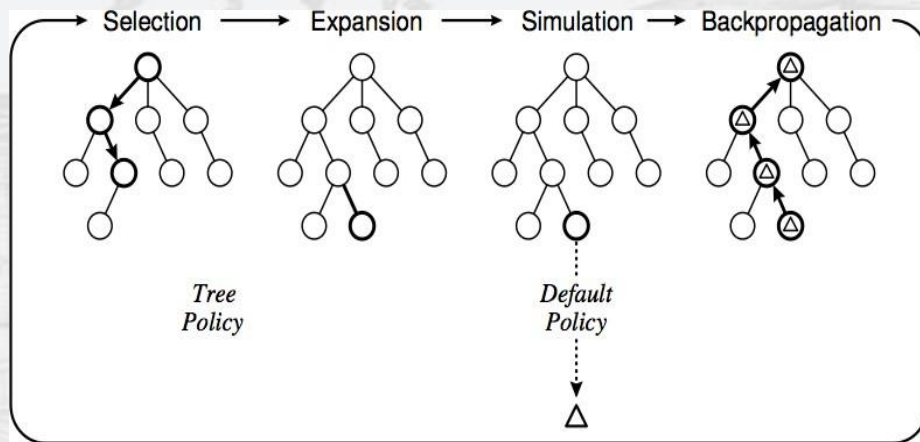
蒙特卡洛树搜索又称随机抽样或统计试验方法，通过构造符合一定规则的随机数（或更常见的伪随机数）来解决数学上的各种问题。**对于那些由于计算过于复杂而难以得到解析解或者根本没有解析解的问题，蒙特卡洛树搜索是一种有效的求出数值解的方法。**挑苹果的例子说明表示样本容量足够大，蒙特卡罗算法越近似最优解。

例：假如筐里有100个苹果，让我每次闭眼拿1个，挑出最大的。于是我随机拿1个，再随机拿1个跟它比，留下大的，再随机拿1个.....我每拿一次，留下的苹果都至少不比上次的小。拿的次数越多，挑出的苹果就越大，但我除非拿100次，否则无法肯定挑出了最大的。

用蒙特卡洛随机模拟的方法对棋局进行估值。其思想很简单，对于当前棋局，随机地模拟双方走步直到分出胜负为止。通过多次模拟，计算每个可下棋点的获胜概率，选取获胜概率最大的点走棋。在围棋程序中实际使用的是一种被称为蒙特卡洛树搜索的方法，边模拟边建立一个搜索树，父节点可以共享子节点的模拟结果，以提高搜索效率。其基本原理如图所示：

分为以下四个过程：

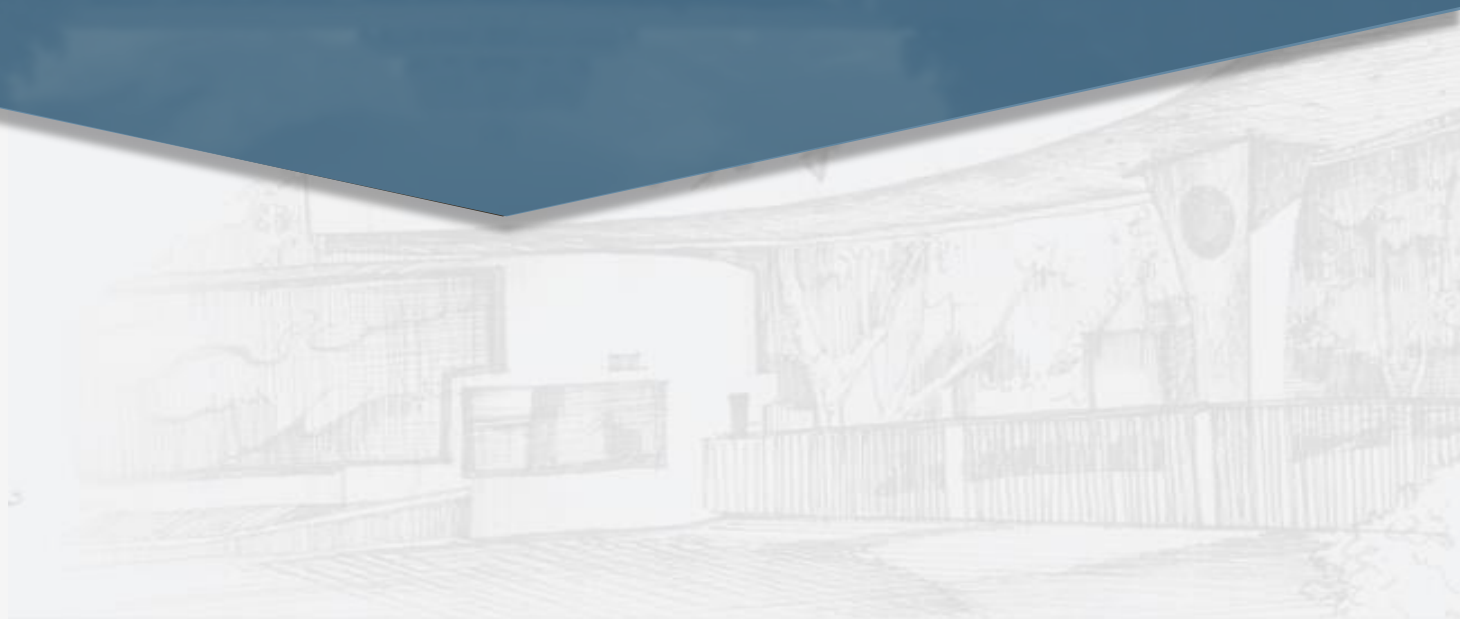
- ❑ **选择：**以当前棋局为根节点，自上而下地选择一个落子点；
- ❑ **扩展：**向选定的节点添加一个或多个子节点；
- ❑ **模拟：**对扩展出的节点用蒙特卡洛方法进行模拟；
- ❑ **回传：**根据模拟结果依次向上更新祖先节点的估计值。





西安交通大学
XI'AN JIAOTONG UNIVERSITY

3.5 群智能算法





受自然界和生物界规律的启迪，人们根据其原理模仿设计了许多求解问题的算法，已经广泛应用于组合优化、机器学习、智能控制、模式识别、规划设计、网络安全等领域。

设想这样一个场景：

一群鸟在随机搜寻食物，在这个区域里只有一块食物，所有的鸟都不知道食物在哪里，但是它们知道当前的位置离食物还有多远。那么找到食物的最优策略是什么呢？

最简单有效的就是搜寻目前离食物最近的鸟的周围区域。

群智能 (Swarm Intelligence, SI)，又称群体智能计算或群集智能计算，是指一类受昆虫、兽群、鸟群和鱼群等的群体行为启发而设计出来的具有分布式智能行为特征的一些智能算法。群智能中的“**群**”指的是一组相互之间可以进行直接或间接通信的群体；“**群智能**”指的是无智能的群体通过合作表现出智能行为的特性。



智能计算作为一种新兴的计算技术，受到越来越多研究者的关注，并和人工生命、进化策略以及遗传算法等有着极为特殊的联系，已经得到广泛的应用。群智能计算在没有集中控制并且不提供全局模型的前提下，为寻找复杂的分布式问题的解决方案提供了基础。

对一般群智能计算，通常要求满足以下五条基本原则：

- ◆ 邻近原则：群内个体具有对简单空间或时间进行计算和评估的能力；
- ◆ 品质原则：群内个体具有对环境以及群内其他个体品质作出响应的能力；
- ◆ 多样性原则：群内不同个体能够对环境中的某些变化做出不同多样反应；
- ◆ 稳定性原则：群内个体行为模式不会在每次环境发生变化时都发生改变；
- ◆ 适应性原则：群内个体能在所需代价不高情况下，适当改变自身行为模式。

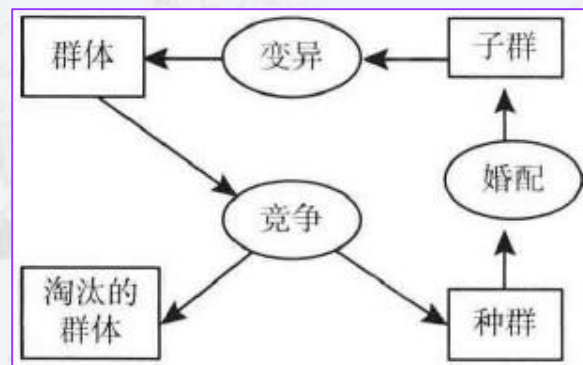
群智能计算现含遗传算法、蚁群算法、粒子群算法、蜂群算法、鸡群算法、猫群算法、鱼群算法、象群算法、狼群算法、果蝇算法、飞蛾扑火算法、萤火虫算法、细菌觅食算法、混合蛙跳算法等诸多智能算法。



遗传算法

遗传算法（Genetic Algorithm, GA）最早是由美国的 John holland 于20世纪70年代提出，该算法是**根据大自然中生物体进化规律而设计提出**的。是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，是一种通过模拟自然进化过程搜索最优解的方法。

该算法通过数学方式，将问题的求解过程转换成类似生物进化中的染色体基因的交叉、变异等过程。在求解较为复杂的组合优化问题时，相对一些常规的优化算法，通常能够较快地获得较好的优化结果。



生物进化的基本过程

适者生存：最适合自然环境的群体往往产生了更大的后代群体。

遗传算法已被人们广泛地应用于组合优化、机器学习、信号处理、自适应控制和人工生命等领域。



群智能算法 — 遗传算法



西安交通大学
XI'AN JIAOTONG UNIVERSITY

遗传算法发展史

- 1962年，Fraser提出了自然遗传算法。
- 1965年，Holland首次提出了人工遗传操作的重要性。
- 1967年，Bagley首次提出了遗传算法这一术语。
- 1970年，Cavicchio把遗传算法应用于模式识别中。
- 1971年，Hollstien在论文《计算机控制系统中人工遗传自适应方法》中阐述了遗传算法用于数字反馈控制的方法。
- **1975年，美国J. Holland出版了《自然系统和人工系统的适配》；DeJong完成了重要论文《遗传自适应系统的行为分析》。**
- 20世纪80年代以后，遗传算法进入兴盛发展时期。





群智能算法 — 遗传算法



西安交通大学
XI'AN JIAOTONG UNIVERSITY

生物遗传 vs 遗传算法

生物遗传概念	遗传算法中的应用
适者生存	目标值比较大的解被选择的可能性大
个体 (Individual)	解
染色体 (Chromosome)	解的编码 (字符串、向量等)
基因 (Gene)	解的编码中每一分量
适应性 (Fitness)	适应度函数值
群体 (Population)	根据适应度值选定的一组解 (解的个数为群体的规模)
婚配 (Marry)	交叉选择两个染色体进行交叉产生一组新的染色体的过程
变异 (Mutation)	编码的某一分量发生变化的过程



群智能算法——遗传算法



西安交通大学
XI'AN JIAOTONG UNIVERSITY

遗传算法基本思想

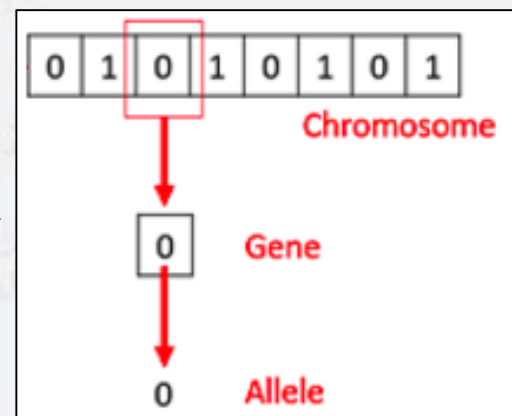
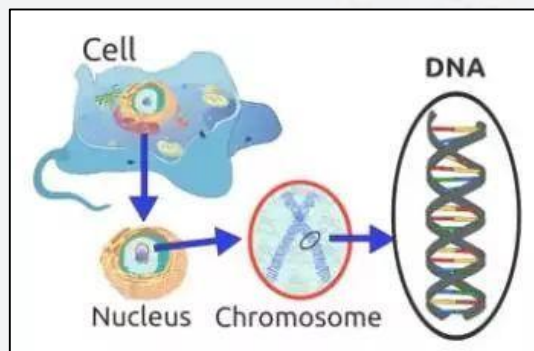
在一个生物的任何细胞中，都有着相同的一套染色体。所谓染色体，就是指由 DNA 组成的聚合体。一条染色体由基因组成，这些基因其实就是组成 DNA 的基本结构，DNA 上的每个基因都编码了一个独特的形状，传统上看，这些染色体可以被由数字 0 和 1 组成的字符串表达出来。

染色体：生物遗传物质的主要载体。

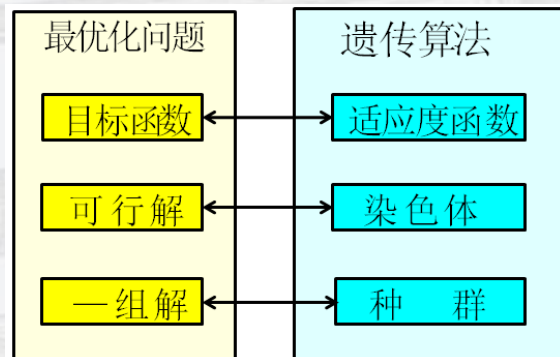
基因：扩展生物性状的遗传物质功能单元和结构单位。

基因座(locus)：染色体中基因位置。

等位基因(alleles)：基因所取的值。



基本思想：在求解问题时从多个解开始，然后通过一定的法则进行逐步迭代以产生新的解。





群智能算法 — 遗传算法



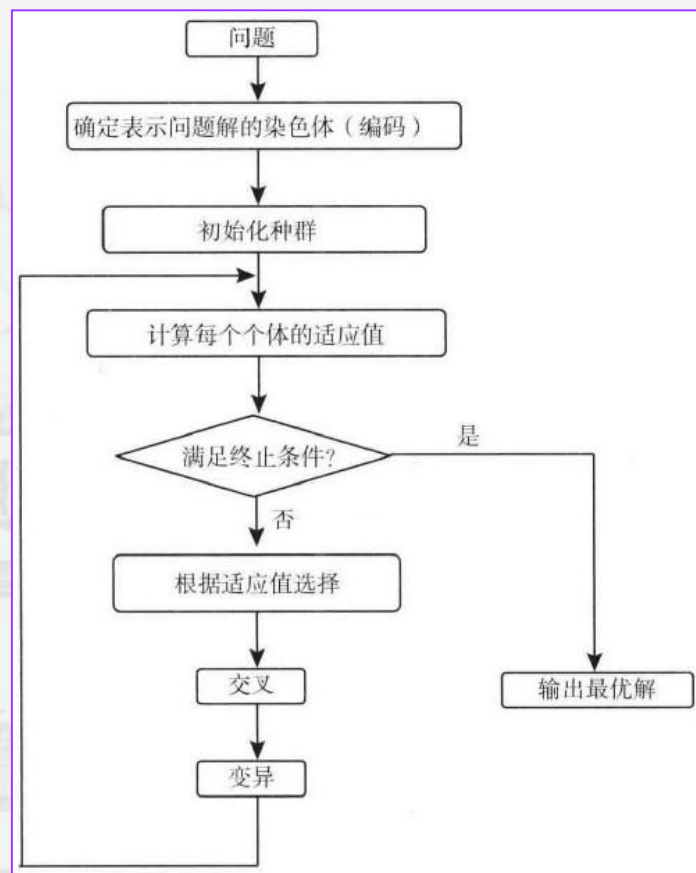
西安交通大学
XI'AN JIAOTONG UNIVERSITY

遗传算法基本要素

以初始群体为起点，经过竞争一部分个体被淘汰，另一部分进入种群。竞争过程有一个标准或者种群适应环境的**评价标准**，适应程度高的**只是进入种群的可能性比较大**，但并不一定进入种群；而适应程度低的只是进入种群的可能性比较小，但并不一定被淘汰。这一重要特性保证了种群的多样性。

遗传算法中**染色体**对应的是数据或数组，通常用一维串结构数据来表示。群体中个体的数量称为种群规模。个体对环境的适应程度叫适应度。适应度大的个体被选择进行遗传操作产生新个体的可能性大，体现了**生物遗传中适者生存的原理**。选择两个染色体进行交叉产生一组新的染色体的过程，类似**生物遗传中的婚配**。编码某一个分量发生变化，类似**生物遗传中的变异**。综合变异的作用，使子群成长为新的群体而取代旧群体。

遗传算法包含的要素有：**编码、群体设定、适应度函数、选择、交叉、变异**。



遗传算法基本流程图



► 编码

(1) 二进制编码：将问题空间的参数编码为一维排列的染色体的方法，称为一维染色体编码方法。其中最常用的符号集是二值符号集 $\{0, 1\}$ ，即是二进制编码。具体是用若干二进制数表示一个个体，将原问题的解空间映射到位串空间 $B = \{0, 1\}$ 上，然后在位串空间上进行遗传操作。

例：变量选择

变量可以看作一个离散组合优化问题，其目的是从 n 个变量中选出 d ($d \leq n$) 个变量作为输入变量进行建模，并使模型的预测精度最大。

设计一个染色体包括 n 个基因，如：00101101.....100111，其中该位为0表示不选择该位所代表的变量，为1则表示选择该位代表的变量。

优点：类似生物染色体的组成，算法易于用生物遗传理论解释，遗传操作如交叉、变异等易实现；算法处理的模式数最多。

缺点：求解高维优化问题的二进制编码串长，搜索效率低。

(2) 实数编码：是用若干实数表示一个个体，采用实数表达法不必进行数制转换，可直接在解的表现型上进行遗传操作。从而引入与问题领域相关的启发式信息来增加算法的搜索能力。



群智能算法 — 遗传算法



西安交通大学
XI'AN JIAOTONG UNIVERSITY

► 群体设定

遗传算法是对群体进行操作的，所以必须为遗传操作准备一个由若干初始解组成的初始群体。

(1)初始种群的产生：根据问题固有知识，把握最优解所占空间在整个问题空间中的分布范围，然后，在此分布范围内设定初始群体。

随机产生一定数目的个体，从中挑选最好的个体加到初始群体中。这种过程不断迭代，直到初始群体中个体数目达到了预先确定的规模。

(2)种群规模的确定：群体中个体的数量称为种群规模，群体规模影响遗传优化的结果和效率。

当群体规模大小时，会使遗传算法的搜索空间范围有限，搜索很可能出现未成熟收敛现象，使**算法陷入局部最优解**；

当群体规模太大时，适应度评估次数增加，则计算复杂；

当群体中个体非常多时，少量适应度很高的个体会被选择生存下来，但大多数个体却被淘汰，**影响配对库的形成，从而影响交叉操作**。

种群规模一般取20~100个个体。



► 适应度函数

遗传算法遵循自然界优胜劣汰的原则，适应度值表示个体的优劣并作为遗传操作的依据。个体的适应度高，则被选择的概率高，反之就低。适应度函数是区分群体中个体好坏的标准，是进行自然选择的依据。因此，适应度函数的设计非常重要。

在具体应用中，适应度函数的设计要结合求解问题本身的要求而定，一般而言，适应度函数是由目标函数变换得到的。将目标函数变换成适应度函数的最直观方法是直接将待求解优化问题的目标函数作为适应度函数。

□ 若目标函数为最大化问题，则 $Fit(f(x)) = f(x)$

□ 若目标函数为最小化问题，则 $Fit(f(x)) = \frac{1}{f(x)}$

例：变量选择

一个染色体包括n个基因，如：00101101.....100111，其中该位为0表示不选择该位所代表的变量，为1则表示选择该位代表的变量。

对通过染色体确定的输入变量子集，且采用某种回归算法在数据集上进行建模，对所建模型效果（染色体对应的变量选择方案）进行对比和分析时，可以考虑采用预测误差平方和（PRESS）作为评价标准。

$$PRESS = \sum_{i=1}^N (y_i - \hat{y}_i)^2$$



► 适应度函数

在遗传算法中，将所有妨碍适应度值高的个体产生，从而影响遗传算法正常工作的问題统称为**欺骗问题（Deceptive Problem）**。

- 过早收敛：缩小这些个体的适应度，以降低这些超级个体的竞争力。
- 停滞现象：改变原始适应值的比例关系，以提高个体之间的竞争力。

为在一定程度上解决这个问题，**提出了适应度尺度变换方法**，通过不同的适应度评价标准来防止优化过程中过早收敛、陷入局部最优解，能产生明显的效果。

在算法初期运行中，为了维护种群多样性，可以降低种群个体之间的适应度差异程度；在算法运行后期中，为了保证优秀的个体遗传下去，要提高个体之间的适应度差异程度。针对以上需求，提出了几种适应度尺度变换（Fitness Scaling）的方法：

- 线性尺度变换： $f' = af + b$
- 乘幂尺度变换： $f' = f^K$
- 指数尺度变换： $f' = e^{-af}$



► 选择

选择操作也称为复制（reproduction）操作：从当前群体中按照一定概率选出优良的个体，使它们有机会作为父代繁殖下一代子孙。判断个体优良与否的准则是各个个体的适应度值，即个体适应度越高，其被选择的机会就越多。

(1) 适应度比例方法或蒙特卡罗法：各个个体被选择的概率和其适应度值成比例。
规模为 M 的群体中个体 i 被选择的概率为：

$$p_{si} = \frac{f_i}{\sum_{i=1}^M f_i}$$

(2) 排序方法：根据适应度大小顺序对群体中个体进行排序，然后把事先设计好的概率按排序分配给个体，作为各自的选择概率。在排序方法中，选择概率仅仅取决于个体在种群中的序位，而不是实际的适应度值。

虽然适应值大的个体仍然会排在前面，有较多的被选择机会，但两个适应度值相差很大的个体被选择的概率相差就没有原来大了。只要符合“**原来适应度值大的个体变换后被选择的概率仍然大**”这个原则，就可以采用各种变换方法。



➤ 选择

(2) 排序方法:

- **线性排序**: 群体成员按适应值大小从好到坏依次排列 x_1, x_2, \dots, x_M , 然后根据一个线性函数给第 i 个个体 x_i 分配选择概率:

$$p_i = \frac{a - bi}{M(M + 1)}$$

- **非线性排序**: 将群体成员按适应值从好到坏依次排列, 并按下式分配选择概率:

$$p_i = \begin{cases} q(1-q)^{i-1} & i = 1, 2, \dots, M-1 \\ (1-q)^{M-1} & i = M \end{cases}$$

可用其他非线性函数来分配选择概率, 只要满足以下条件:

- (1) 若 $P = \{x_1, x_2, \dots, x_M\}$ 且 $f(x_1) \geq f(x_2) \geq \dots \geq f(x_M)$, 则 p_i 满足

$$p_1 \geq p_2 \geq \dots \geq p_M$$

- (2) $\sum_{i=1}^M p_i = 1$



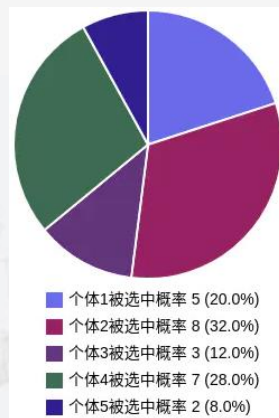
► 选择

(3) 转盘赌选择：按个体的选择概率产生一个轮盘，轮盘每个区的角度与个体的选择概率成比例。适应度越大，选中概率也越大。但实际在进行轮盘赌选择时个体的选择往往不是依据个体的选择概率，而是根据“**累积概率**”来进行选择。

个体	1	2	3	4	5	6	7	8	9	10	11
适应度	2.0	1.8	1.6	1.4	1.2	1.0	0.8	0.6	0.4	0.2	0.1
选择概率	0.18	0.16	0.15	0.13	0.11	0.09	0.07	0.05	0.03	0.02	0.01
累积概率	0.18	0.34	0.49	0.62	0.73	0.82	0.89	0.94	0.97	0.99	1.00

第2轮产生一个随机数：0.32

第1轮产生一个随机数：0.81



(4) 锦标赛选择方法：从群体中随机选择个个体，将其中适应度最高的个体保存到下一代。这一过程反复执行，直到保存到下一代的个体数达到预先设定的数量为止。

(5) 随机竞争方法：每次按赌轮选择方法选取一对个体，然后让这两个个体进行竞争，适应度高者获胜。如此反复，直到选满为止。

(6) 最佳个体保存方法：群体中适应度最高的个体不进行交叉而直接复制到下一代中，保证遗传算法终止时得到的最后结果一定是历代出现过的最高适应度的个体。



群智能算法 — 遗传算法



西安交通大学
XI'AN JIAOTONG UNIVERSITY


► 交叉

对两个相互配对的染色体按某种方式相互交换其部分基因，从而形成两个新的个体。

交叉概率 (P_c) 用来确定两个染色体进行局部互换以产生两个新的子代的概率。每次从群体中选择两个染色体，同时生成0和1之间的一个随机数，然后**根据这个随机数确定这两个染色体是否需要交叉**，如果这个随机数低于交叉概率，就进行交叉然后沿着染色体的长度随机选择位置，再根据位置进行基因交换。

(1) 一点交叉：在个体串中随机设定一个交叉点，实行交叉时，该点前或后的两个个体的部分结构进行互换，并生成两个新的个体。

(2) 二点交叉：随机设置两个交叉点，将两个交叉点之间的码串相互交换。



$A = 9$	8	4		5	6	7		1	3	2
$B = 8$	7	1		2	3	9		5	4	6
$A' = 9$	8	4		2	3	9		1	3	2
$B' = 8$	7	1		5	6	7		5	4	6

(3) 多点交叉：或称广义交叉，是指在个体编码串中随机设置多个交叉点，然后进行基因交换。



群智能算法 — 遗传算法



西安交通大学
XI'AN JIAOTONG UNIVERSITY

► 变异

将个体染色体编码串中的某些基因座上的基因值用该基因座上的其它等位基因来替换，从而形成新的个体。

```
1 0 1 1 0 1 0 0 1 0 1 1 0 0 1
0 0 1 1 0 1 0 1 1 0 1 1 0 0 1
```

交叉算子因其全局搜索能力而作为主要算子，变异算子因其局部搜索能力而作为辅助算子。遗传算法通过交叉和变异这对相互配合又相互竞争的操作而使其具备兼顾全局和局部的均衡搜索能力。所谓相互配合是指当群体在进化中陷于搜索空间中某个超平面而仅靠交叉不能摆脱时，通过变异操作可有助于这种摆脱。

变异属于辅助性的搜索操作。变异概率 (P_m) 一般不能大，以防止群体中重要的、单一的基因被丢失。

(1) 位点变异: 群体中的个体码串，随机挑选一个或多个基因座，并对这些基因座的基因值以变异概率作变动。

(2) 逆转变异: 在个体码串中随机选择两点（逆转点），然后将两点之间的基因值以逆向排序插入到原位置中。

(3) 插入变异: 在个体码串中随机选择一个码，然后将此码插入随机选择的插入点中间。



遗传算法的特点

- 遗传算法对所求解的优化问题没有太多的数学要求，搜索过程中不需要问题的内在性质，**可直接对结构对象进行操作**。所谓结构对象，泛指集合、序列、矩阵、树、图、链和表等各种一维、二维甚至三维结构形式。因此，遗传算法具有非常广泛的应用领域。
- 利用随机技术指导对一个被编码的参数空间进行**高效率搜索**，能够非常有效地进行概率意义的全局搜索。
- 采用群体搜索策略，**易于并行化**。
- 仅用**适应度函数值来评估个体**，并在此基础上进行遗传操作，使种群中个体之间进行信息交换，适应度函数不仅不受连续可微的约束，而且其**定义域也可以任意设置**，对适应度函数唯一要求是能够算出可以比较的正值，使其应用范围大大扩展，非常适合于传统优化方法难以解决的复杂优化问题。



例：流水车间调度问题

问题描述： n 个工件要在 m 台机器上加工，每个工件需要经过 m 道工序，每道工序要求不同的机器， n 个工件在 m 台机器上的加工顺序相同。工件在机器上的加工时间是给定的，设为： $t_{ij}(i=1, \dots, n; j=1, \dots, m)$

问题目标： 确定 n 个工件在每台机器上最优加工顺序，使最大流程时间最小。

数学模型：

$c(j_i, k)$ ：工序 j_i 在机器 k 上的加工完工时间，

$\{j_1, j_2, \dots, j_n\}$ ：工件的调度

n 个工件、 m 台机器的完工时间：

$$c(j_1, 1) = t_{j_1 1}$$

$$c(j_1, k) = c(j_1, k-1) + t_{j_1 k}, \quad k = 2, \dots, m$$

$$c(j_i, 1) = c(j_{i-1}, 1) + t_{j_i 1}, \quad i = 2, \dots, n$$

$$c(j_i, k) = \max\{c(j_{i-1}, k), c(j_i, k-1)\} + t_{j_i k}, \quad i = 2, \dots, n; k = 2, \dots, m$$

最大流程时间： $c_{\max} = c(j_n, m)$

调度目标：确定 $\{j_1, j_2, \dots, j_n\}$ 使得 c_{\max} 最小

假设条件：

- (1) 每个工件在机器上加工顺序是给定的。
- (2) 每台机器同时只能加工一个工件。
- (3) 一个工件不能同时在不同的机器上加工。
- (4) 工序不能预定。
- (5) 工序准备时间与顺序无关，且包含在加工时间中。
- (6) 工件在每台机器上加工顺序相同，且是确定的。



例：流水车间调度问题 — 算法设计

编码：对于调度问题，通常不采用二进制编码，而**使用实数编码**。将各个生产任务编码为相应的整数变量。调度方案是生产任务的一个排列，其排列中每个位置对应于每个带编号的任务。根据一定评价函数，用遗传算法求出最优的工件加工排列。

该问题的**编码方式是用染色体表示工件的顺序**，对于有四个工件的情况，第 k 个染色体 $v_k=[1, 2, 3, 4]$ ，表示工件的加工顺序为： j_1, j_2, j_3, j_4 。

适应度函数：调度的目标是使最大流程时间最小，所以适应度函数为：

$$eval(v_k) = \frac{1}{c_{\max}^k}$$

其中， c_{\max}^k 是第 k 个染色体 v_k 的最大流程时间。

例6.1 Ho 和 Chang(1991) 给出的5个工件、4台机器问题的加工时间表：

工件 j	t_{j1}	t_{j2}	t_{j3}	t_{j4}
1	31	41	25	30
2	19	55	3	34
3	23	42	27	6
4	13	22	14	13
5	33	5	57	19



群智能算法 — 遗传算法



西安交通大学
XI'AN JIAOTONG UNIVERSITY

例：流水车间调度问题

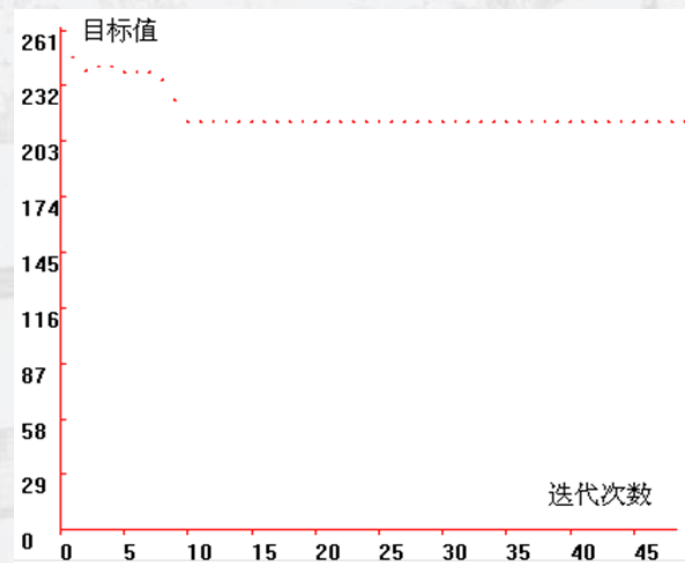
例6.1 Ho 和 Chang(1991) 给出的5个工件、4台机器问题

用穷举法求得最优解：4-2-5-1-3，加工时间：213；最劣解：1-4-2-3-5，加工时间：294；平均解的加工时间：265。

用遗传算法求解，交叉概率=0.6，变异概率=0.1，种群规模=20，迭代次数=50。

总运行次数	最好解	最坏解	平均	最好解的频率	最好解的平均代数
20	213	221	213.95	0.85	12

用遗传算法求解绝大部分都能够得到最优解，即使没有找到最优解，也能够找到比较好的解。上例中用遗传算法找到的解中最差的也有221，比平均解265要好很多。



遗传算法收敛图



群智能算法 — 蚁群算法

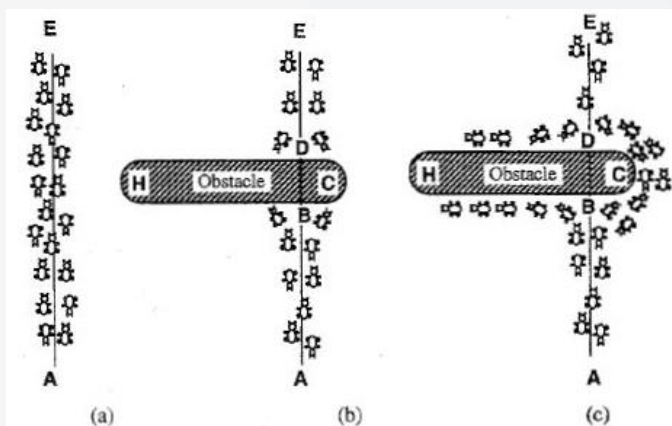


西安交通大学
XI'AN JIAOTONG UNIVERSITY

蚁群算法

蚁群算法 (Ant Colony Optimization, ACO) 是由一群无智能或有轻微智能的个体 (Agent) 通过相互协作而表现出智能行为, 从而为求解复杂问题提供了一个新的可能性。蚁群算法最早是由意大利学者Coloni A., Dorigo M. 等于1991年提出。蚁群算法是一种**仿生学算法**, 是由自然界中蚂蚁觅食的行为而启发的。在自然界中, **蚂蚁觅食过程中**, 蚁群总能够按照寻找到一条从蚁巢和食物源的最优路径。

图(a)中有一群蚂蚁, 假如A是蚁巢, E是食物源。这群蚂蚁将沿着蚁巢和食物源之间的直线路径行驶。假如在A和E之间突然出现了一个障碍物, 如图(b), 那么, 在B点 (或D点) 的蚂蚁将要做出决策, 到底是向左行驶还是向右行驶?



一开始路上没有前面蚂蚁留下的信息素(pheromone), 蚂蚁朝着**两个方向行进的概率是相等的**。但是当有蚂蚁走过时, 它将会在它行进的路上**释放出信息素**, 并且这种信息素会以一定的速率散发掉。信息素是蚂蚁之间交流的工具之一。它后面的蚂蚁通过路上信息素的浓度, 做出决策, 往左还是往右。很明显, 沿着短边的路径上信息素将会越来越浓, 如图(c), 从而吸引了越来越多的蚂蚁沿着这条路径行驶。



群智能算法 — 蚁群算法



西安交通大学
XI'AN JIAOTONG UNIVERSITY

蚁群觅食习性

- **环境：**有障碍物、有其它蚂蚁、有信息素。
- **信息素跟踪：**按照一定的概率沿着信息素较强的路径觅食。
- **信息素遗留：**会在走过的路上会释放信息素，使得在一定的范围内的其他蚂蚁能够觉察到并由此影响它们的行为。
- **觅食规则：**范围内寻找是否有食物，否则看是否有信息素，每只蚂蚁都会以小概率犯错。
- **移动规则：**都朝信息素最多的方向移动，无信息素则继续朝原方向移动，且有随机的小的扰动，有记忆性。
- **避障规则：**移动的方向如有障碍物挡住，蚂蚁会随机选择另一个方向。
- **信息素规则：**越靠近食物播撒的信息素越多，越离开食物播撒的信息素越少。



群智能算法 — 蚁群算法



西安交通大学
XI'AN JIAOTONG UNIVERSITY

蚁群优化算法的第一个应用是著名的旅行商问题

旅行商问题 (Traveling Salesman Problem, TSP) :

在寻求单一旅行者由起点出发, 通过所有给定的需求点之后, 最后再回到原点的最小路径成本。

蚂蚁搜索食物的过程:

通过个体之间的信息交流与相互协作最终找到从蚁穴到食物源的最短路径。

蚁群系统的模型

m 是蚁群中蚂蚁的数量;

$d_{xy}(x, y=1, \dots, n)$ 表示元素(城市)和元素(城市)之间的距离;

$\eta_{xy}(t)$ 表示能见度, 称为启发信息函数, 等于距离的倒数, 即 $\eta_{xy}(t) = \frac{1}{d_{xy}}$

$b_x(t)$ 表示 t 时刻位于城市 x 的蚂蚁的个数, $m = \sum_{x=1}^n b_x(t)$

$\tau_{xy}(t)$ 表示 t 时刻在 xy 连线残留信息素, 初始时刻各路径上信息素相等: $\tau_{xy}(0) = C(const)$

$P_{xy}^k(t)$ 表示在 t 时刻蚂蚁 k 选择从元素(城市) x 转移到元素(城市) y 的概率, 也称随机比例规则。



群智能算法 — 蚁群算法



西安交通大学
XI'AN JIAOTONG UNIVERSITY

蚁群优化算法的第一个应用是著名的旅行商问题

蚁群系统的模型

$$P_{xy}^k(t) \text{ 表示如下: } P_{xy}^k(t) = \begin{cases} \frac{[\tau_{xy}(t)]^\alpha [\eta_{xy}(t)]^\beta}{\sum_{y \in allowed_k(x)} [\tau_{xy}(t)]^\alpha [\eta_{xy}(t)]^\beta} & \text{if } y \in allowed_k(x) \\ 0 & \text{否则} \end{cases}$$

其中, $allowed_k(x) = \{0, 1, \dots, n-1\} - tabu_k(x)$ 表示蚂蚁 k 下一步允许选择的
城市; $tabu_k(x)$ ($k = 1, 2, \dots, m$) 记录蚂蚁 k 当前所走过的城市; α 是信息素启发式因子, 表示轨迹的
相对重要性。

α 值越大	该蚂蚁越倾向于选择其它蚂蚁经过的路径, 该状态转移概率越接近于贪婪规则。
当 $\alpha=0$ 时	不再考虑信息素水平, 算法就成为有多重起点的随机贪婪算法。
当 $\beta=0$ 时	算法就成为纯粹的正反馈的启发式算法。

用参数 $1-\rho$ 表示信息素消逝程度, 蚂蚁完成一次循环,

各路径上**信息素浓度消散规则**为: $\tau_{xy}(t) = \rho\tau_{xy}(t) + \Delta\tau_{xy}(t)$

蚁群的**信息素浓度更新规则**为: $\Delta\tau_{xy}(t) = \sum_{k=1}^m \Delta\tau_{xy}^k(t)$

其中, $\tau_{xy}(t)$ 为当前路径上的信息素; $\Delta\tau_{xy}(t)$ 为路径 (x, y) 上的信息素的增量。



群智能算法 — 蚁群算法



西安交通大学
XI'AN JIAOTONG UNIVERSITY

蚁群优化算法的第一个应用是著名的旅行商问题

蚁群系统的模型

$\Delta\tau_{xy}^k(t)$ 的一种模型称为 **蚂蚁圈系统 (Ant-cycle System)**，第 k 只蚂蚁留在路径 (x, y) 上的信息素的增量为：

$$\Delta\tau_{xy}^k(t) = \begin{cases} \frac{Q}{L_k} & \text{若第} k \text{只蚂蚁在本次循环中从} x \text{到} y \\ 0 & \text{否则} \end{cases}$$

其中， Q 为常数； L_k 为优化问题的目标函数值，表示第 k 只蚂蚁在本次循环中所走路径的长度。

由于这种方法**利用了全局信息 Q/L_k** ，则具有以下优点：

- ❑ 保证了残留信息素不至于无限累积；
- ❑ 如果路径没有被选中，那么上面的残留信息素会随时间的推移而逐渐减弱，这使算法能“忘记”不好的路径；
- ❑ 即使路径经常被访问也不至于因为 $\Delta\tau_{xy}^k(t)$ 的累积，而产生 $\Delta\tau_{xy}^k(t) \gg \eta_{xy}(t)$ 使期望值的作用无法体现；
- ❑ 充分体现了算法中全局范围内较短路径(较好解)的生存能力；
- ❑ 加强了信息正反馈性能；
- ❑ 提高了系统搜索收敛的速度。



蚁群算法的参数选择

(1) 信息素启发因子 α :

- 反映了蚁群在路径搜索中随机性因素作用的强度;
- α 值越大, 蚂蚁选择以前走过的路径的可能性越大, 搜索的随机性减弱;
- 当 α 过大时会使蚁群的搜索过早陷于局部最优。

(2) 期望值启发式因子 β :

- 反映了反映了蚁群在路径搜索中先验性、确定性因素作用的强度;
- β 值越大, 蚂蚁在某个局部点上选择局部最短路径的可能性越大;
- 虽然搜索的收敛速度得以加快, 但蚁群在最优路径的搜索过程中随机性减弱, 易于陷入局部最优。

(3) 信息素挥发度 $1-\rho$:

- 当要处理的问题规模比较大时, 会使那些从来未被搜索到的路径(可行解)上的信息量减小到接近于0, 因而降低了算法的全局搜索能力;
- 而且当 $1-\rho$ 过大时, 以前搜索过的路径被再次选择的可能性过大, 也会影响到算法的随机性能和全局搜索能力;
- 反之, 通过减小信息素挥发度 $1-\rho$ 虽然可以提高算法的随机性能和全局搜索能力, 但又会使算法的收敛速度降低。



群智能算法 — 蚁群算法



西安交通大学
XI'AN JIAOTONG UNIVERSITY

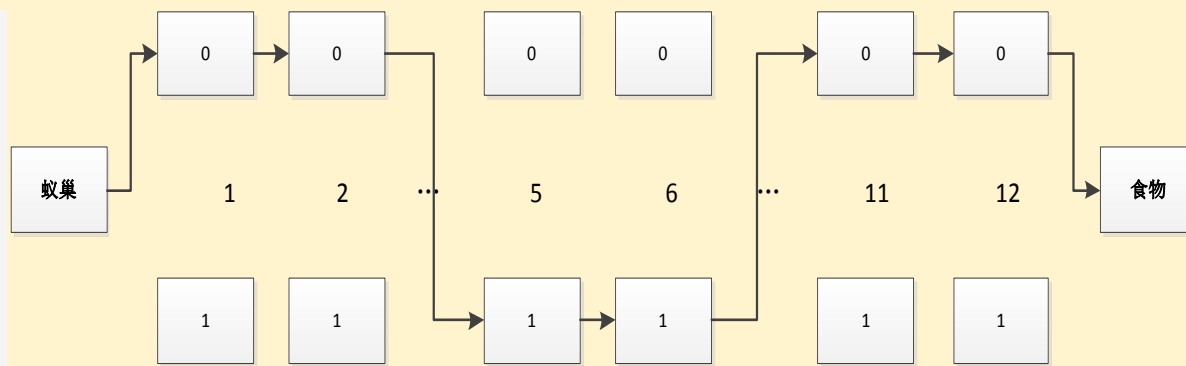
例：制粉系统变量选择问题

问题描述：要制粉系统中可测量的变量主要有以下12个，即磨机负荷、入口负压、出口负压、出入口差压、入口温度、出口温度、粗粉出口负压、细粉出口负压、排粉出口风压、磨机电机电流、排粉机电流、通风量。

问题目标：针对制粉出力建模的变量选择在本质上是一个离散组合优化问题，其目的是从以上12个变量中选出 d ($d \leq 12$) 个变量作为输入变量进行建模，并使模型预测精度最大。

解决思路：

对于该变量选择问题，将选中的变量标记为1，未被选中的标记为0，这样在12个变量中进行变量选择即为在一个 2×12 的矩阵中寻找最优路径问题，得到的最优路径即为变量选择的最优解，形式为12位二进制数。





粒子群算法

粒子群优化 (Particle Swarm Optimization, PSO) 算法是由美国普渡大学Kennedy和Eberhart于1995年提出, 基本概念**源于对鸟群觅食行为**的研究。

生物学家Craig Reynolds在1987年提出了非常有影响的鸟群聚集模型, 其中每个个体遵循下面三条简单的规则, 就可以非常接近的模拟出鸟群飞行的现象。



- (1) 避免与邻域个体相冲撞;
- (2) 匹配邻域个体的速度;
- (3) 飞向鸟群中心, 且整个群体飞向目标。

- ❑ 1997年, Eberhart针对最初PSO算法只适于连续优化问题的缺点, 提出PSO算法的**离散二进制版**, 用来解决工程实际中的组合优化问题;
- ❑ 1998年和1999年, Shi和Clerc等通过提出**惯性模型及压缩因子模型**, 进一步完善了PSO算法理论, 现有PSO算法研究几乎全部基于上述两种模型。

人们对PSO优化研究兴趣的日益增长主要有2个方面的原因:

一是工程领域, 尤其是在智能控制领域, 由于大规模非线性系统中存在传统优化算法不能有效解决的问题, 如模糊系统中隶属度函数的确定等问题;

二是PSO算法本身是模拟自然演化这一过程的, 不仅对优化函数本身没有什么要求, 既不要求连续也不要求可微, 而且相对其它算法还具有参数设置少、操作简单等优点。



PSO的发展

- ❑ 1999年，Eberhart等开始对算法中粒子信息共享方法的研究；
- ❑ 2000年，Carlisle提出了动态环境下自适应PSO算法；
- ❑ 2002年，Coello第一次把PSO算法用于多目标优化；
- ❑ 2002年，Clerc利用代数和数学分析理论对离散状态和连续状态下粒子的轨迹与参数设置之间关系进行了深入研究，并给出了四种改进模型；
- ❑ 2004年，PSO模型得到了重新泛化，Mendes等提出一种全息PSO模型；
- ❑ 2005年至今，PSO算法的研究进入了高峰。

与此同时，与PSO算法相关的**著作、会议及网络资源**也得到了空前发展。

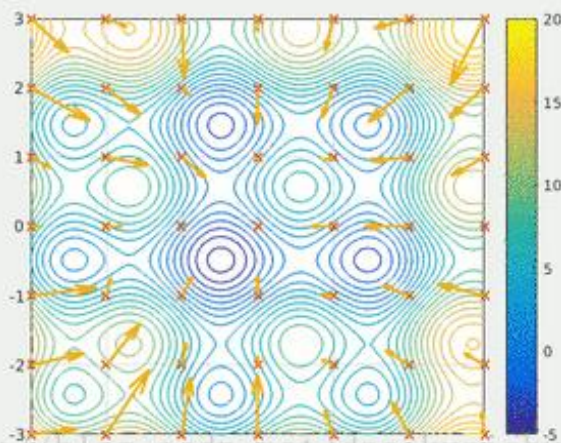
- ❑ 1999年在首届进化计算会议上(Congress on Evolutionary Computation, CEC)上PSO算法即被定为会议讨论议题；
- ❑ 2001年，Kennedy等出版了优秀著作“Swarm Intelligence”，对PSO及其应用作了全面而系统的论述；
- ❑ 2003年4月，在美国印第安纳波利斯举办了首届IEEE群体智能研讨论坛，PSO算法被列为会议的主要议题。



算法介绍

在粒子群算法里，鸟被抽象成为没有质量和体积的粒子，即将每个个体在 G 维搜索空间中以一定的速度飞行，该速度决定粒子飞行的方向和距离。粒子 i 在 G 维空间中的位置表示为向量： $x_i = (x_{i1}, x_{i2}, \dots, x_{iG})$ ，飞行速度表示为矢量： $v_i = (v_{i1}, v_{i2}, \dots, v_{iG})$ ，每个粒子都有一个由适应度函数(Fitness function)决定的适应度值。并且知道到当前为止，自己发现的最好粒子 $pbest_i$ 和现在的位置 x_i ，这个可以看作粒子自己的飞行经验。除此之外，每个粒子还知道到目前为止整个群体所有粒子找到的最优位置 $gbest$ ， $gbest$ 是 $pbest_i$ 中最好值，这个是整个群体的经验。

PSO初始化为一群随机粒子，每个粒子在搜索空间中单独搜寻最优解，并将个体极值与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在每一次迭代中，粒子通过跟踪两个“极值”来更新自己。另外，也可不用整个种群而只是用其中一部分最为粒子的邻居，那么在所有邻居中的极值就是局部极值。



算法过程演示图



群智能算法——粒子群算法



西安交通大学
XI'AN JIAOTONG UNIVERSITY

算法介绍

粒子通过下式来更新自己的速度和位置：

$$\mathbf{V}_{ig}^{(t+1)} = \omega(t) \times \mathbf{v}_{ig}^{(t)} + c_1 \times \text{rand}() \times (pbest_{ig}^{(t)} - x_{ig}^{(t)}) + c_2 \times \text{rand}() \times (gbest_g^{(t)} - x_{ig}^{(t)})$$

$$x_{ig}^{(t+1)} = x_{ig}^{(t)} + v_{ig}^{(t+1)}$$

其中， $i=1, 2, \dots, n$ ， $g=1, 2, \dots, G$ ；

$\omega(t)$ 为在第 t 次迭代的惯性权重因子；

$v_{ig}^{(t)}$ 为第 i 个粒子第 g 维在第 t 次迭代的速度；

$pbest_{ig}^{(t)}$ 为 t 次迭代中第 i 个粒子所找到的个体最优解；

$gbest_g^{(t)}$ 为第 t 次迭代时的群体所有粒子所找到的全局最优值；

$x_{ig}^{(t)}$ 为第 t 次迭代第 i 个粒子第 g 维的位置；

c_1, c_2 为加速因子(或者称为学习因子)，分别调节向全局最优粒子和个体最优粒子方向飞行的最大步长，若太小，则粒子可能远离目标区域，若太大则会导致突然向目标区域飞去，或者飞过目标区域；

rand() 函数是产生[0, 1]之间的随机数，增加搜索方向的随机性和算法多样性。

公式右边由三部分组成，
第一部分为“惯性”或“动量”，反映了粒子的运动“习惯”，代表粒子有维持自己先前速度的趋势；
第二部分为“认知”，反映了粒子对自身历史经验的记忆或回忆，代表粒子有向自身历史最佳位置的逼近趋势；
第三部分为“社会”，反映了粒子间协同合作与知识共享的群体历史经验，代表粒子有向群体或邻域历史最佳位置逼近的趋势。



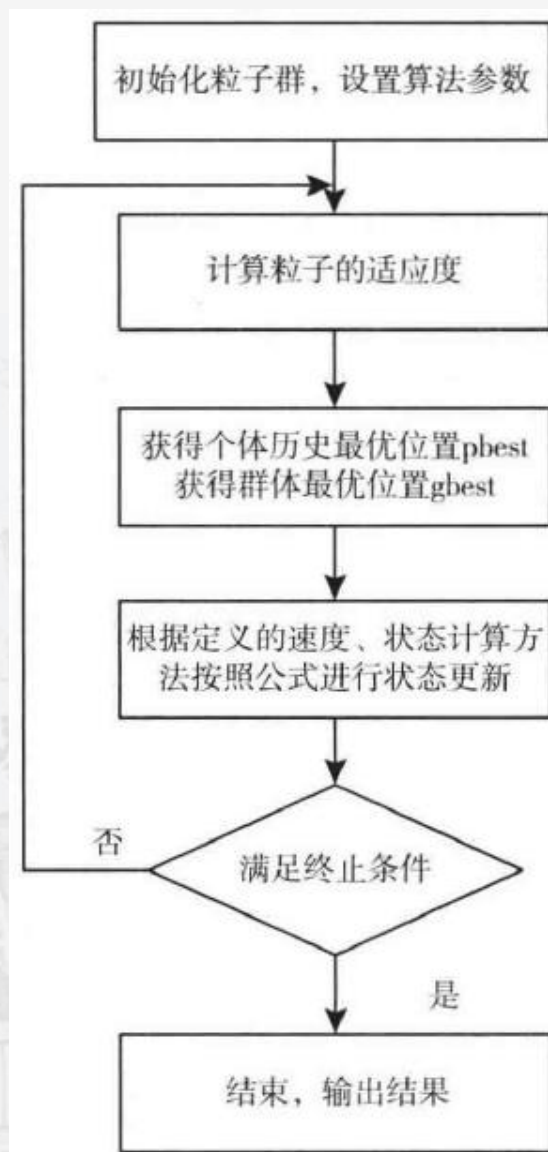
群智能算法——粒子群算法



西安交通大学
XI'AN JIAOTONG UNIVERSITY

PSO算法流程

- ① 初始化每个粒子，在允许范围内**随机设置每个粒子的初始位置和速度**；
- ② 评价**粒子的适应度**，即计算每个粒子的目标函数；
- ③ 设置每个粒子的 P_i 。对每个粒子，将其适应度与其经历过的最好位置 P_i 进行比较，如果优于 P_i ，则将其作为**该粒子的最好位置** P_i ；
- ④ 设置全局最优值 P_g ，对每个粒子，将其适应度与群体经历过的最好位置 P_g 进行比较，如果优于 P_g ，则将其作为当前**群体的最好位置** P_g ；
- ⑤ 更新粒子的速度和位置，即根据公式（可能会有多种变形）**更新粒子的速度和位置**；
- ⑥ 检查终止条件，如果未达到设定条件（**预设误差或者迭代的次数**），则返回第②步。





参数设置

为防止粒子远离搜索区域空间，粒子的每一维速度都被钳位在 $[-v_{gmax}, +v_{gmax}]$ 之间， v_{gmax} 太大，粒子将飞离最优解，太小将会陷入局部最优。

早期实验将 ω 固定为1.0，将 c_1 和 c_2 固定为2.0。因此 v_{gmax} 成为唯一需要调节的参数，通常设每维变化范围10%~20%。Suganthan实验表明， c_1 和 c_2 为常数时可得到较好的解，但不一定必须为2。

粒子群优化算法**初始群体**的产生方法与遗传算法类似，**一般采用随机产生**。粒子群优化算法的种群大小根据问题的规模而定，同时要考虑运算时间。

粒子的适应度函数根据具体问题而定，通常将目标函数转换成适应度函数，与遗传算法类似。

在基本的粒子群优化算法中，**粒子的编码使用实数编码方法**。这种编码方法在求解连续的函数优化问题中十分方便同时对粒子的速度求解与粒子的位置更新也很自然。

$$V_{ig}^{(t+1)} = \omega(t) \times v_{ig}^{(t)} + c_1 \times rand() \times (pbest_{ig}^{(t)} - x_{ig}^{(t)}) + c_2 \times rand() \times (gbest_g^{(t)} - x_{ig}^{(t)})$$

位置更新方程中各部分的影响

- (1) **只有第1部分，即 $c_1=c_2=0$** ，则粒子将一直以当前的速度飞行，直到达边界。由于它只能搜索有限的区域，所以很难找到好解；
- (2) **没有第1部分，即 $\omega=0$** ，则速度只取决于粒子当前位置和其历史最好位置，速度本身没有记忆性，假设一个粒子位于全局最好位置，它将保持静止，此时更像一个局部算法；
- (3) **没有第2部分，即 $c_1=0$** ，则粒子没有认知能力，也就是只有“社会”。在粒子的相互作用下，有能力达到新的搜索空间。但对复杂问题，容易陷入局部最优点
- (4) **没有第3部分，即 $c_2=0$** ，则粒子间没有社会共享信息，就是只有“认知”。因为个体间没有交互，规模为 M 的群体等价于 M 个单个粒子的运行，因而得到最优解的机率非常小。



例：PSO算法整定PID参数

问题描述：PID参数传统整定方法许多工程应用领域仍然在使用，智能优化算法能够在解空间中实现自动搜索，效率较高且往往能够得到最优值。

问题目标：使用PSO算法整定PID参数，使系统阶跃响应的偏差趋于零，有较快的响应速度，即较短的上升时间，有较小的调节时间，控制超调量在5%以内。

解决思路：

用PSO解决该优化问题最重要的两个方面——**编码和适应度函数**。

编码：PSO的一个优势就是采用实数编码，对于PID的三个参数可以直接编码为 (k_p, k_i, k_d) ；

适应度函数：

- 控制偏差： $J_1 = IAE = \int_0^T |e(t)| dt$
- 进一步考虑，系统反应和超调： $J_2 = ITAE = \int_0^T t |e(t)| dt$
- 进一步考虑上升时间： $J_3 = (1 - e^{-\beta}) \cdot (\sigma + \int_0^T |e(t)| dt) + e^{-\beta} \cdot (t_s - t_r)$
- 考虑消除负调： $J_4 = (1 - e^{-\beta}) \cdot (\sigma + \int_0^T |e(t)| dt) + e^{-\beta} \cdot (t_s - t_r) + \gamma \cdot t_r$
- 考虑消除负调： $J_5 = c \cdot |\min(y(t))| + (1 - e^{-\beta}) \cdot (\sigma + \int_0^T |e(t)| dt) + e^{-\beta} \cdot (t_s - t_r) + \gamma \cdot t_r$

粒子数一般取20~40。其实对于大部分的问题10个粒子已经足够可以取得好的结果，不过对于比较难的问题或者特定类别的问题，粒子数可以取到100或200。本问题较复杂，取40~100个粒子。

一般来说迭代次数只要使粒子能够收敛到一定程度就可以了，当迭代次数增加时，运算时间也成倍地增加，过大的迭代次数会耗费大量的时间。本问题设置的是当达到迭代次数时终止。



猫群算法

猫群算法（Cat Swarm Optimization, CSO）是由Shu-An Chu等人在2006年首次提出。这个算法的提出是**基于对猫的行为的观察**。在日常生活中，猫一般都是懒散地躺在某处，处于一种休息和渴望的状态，然而即使在这种状态下，它们也对周围保持着高度的敏感，尤其对于周围活动的物体具有强烈的好奇心。一旦猫发现自己感兴趣的目标，便对其进行追踪，迅速的捕获猎物。

对活动物体强烈的**好奇心和迅速的捕猎能力**便是猫的两个鲜明的特点，而猫群算法中的两个行为模式（搜寻模式和跟踪模式）便是分别基于这两个特点而设计的。

1) 搜寻模式

在搜寻模式中，猫缓慢而谨慎的移动着，在它转移到下一个地点之前将会仔细的观察周围的环境。其中定义了4个要素，分别是**记忆池大小**（seeking memory pool, SMP），**个体上选中维的改变范围**（seeking range of the selected dimension, SRD），**需要改变的维的个数**（counts of dimension to change, CDC）和**自身位置标识位**（self position consideration, SPC）。

记忆池定义了每一只猫的搜寻记忆的大小，用来存放猫曾经搜寻到的点。SRD代表了选中维或选中的基因上变异的范围。CDC代表是个体总维数以内的一个随机值，而SPC是一个布尔值，它指示了猫现在所在的位置是否能称为候选位置。无论SPC为真或假，记忆池的大小SMP都不会改变。。

2) 跟踪模式

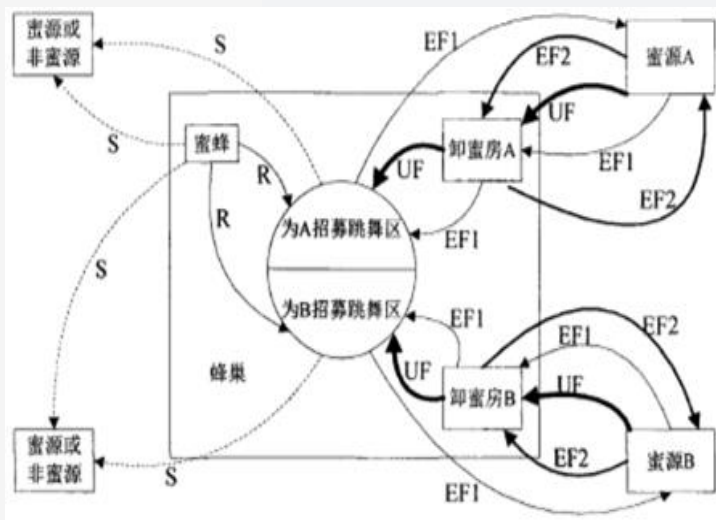
在跟踪模式中，猫开始追踪目标。**猫群算法的追踪模式与粒子群算法类似**，同样采用速度-模型来更新猫的位置和速度。



蜂群算法

蜂群算法（Bee Colony Algorithm）根据蜜蜂各自的分工不同，对**蜂群信息共享和交流**及进行采蜜活动等观察后进行计算机模拟，以此找到问题最优解，是一种基于群智能的全局优化算法。

除蜂王外的整个蜂群的蜜蜂分为三类，即采蜜蜂、观察蜂和侦察蜂，**整个蜂群的目标是寻找花蜜量最大的蜜源**。在蜂群算法中，采蜜蜂利用先前的蜜源信息采蜜，寻找新的蜜源并与观察蜂分享蜜源信息；观察蜂依据与采蜜蜂分享的信息寻找新的蜜源；侦察蜂的任务是寻找一个新的更有价值的蜜源，它们在蜂房附近随机地寻找蜜源。**蜂群算法通过对三类蜜蜂功能的模拟，以寻找最短路径、最大蜜源为目标**，用于寻求问题的最优解。蜜蜂采蜜的群体智能就是**通过不同角色之间的交流转换及协作来实现的**。具体采蜜过程如图所示。



最初阶段，蜜蜂以侦察蜂形式出现，且对周围蜜源没有任何了解，由于蜜蜂内在动机和外条件不同，侦察蜂有两种选择：①成为雇佣蜂，开始在蜂巢周围随机搜索蜜源；②成为跟随蜂，在观察完摇摆舞后开始搜索蜜源。

假设发现蜜源后，该侦察蜂变成一只雇佣蜂，雇佣蜂利用其自身属性记住蜜源位置，并立刻投入到采蜜中。采蜜完成后蜜蜂带着满载花蜜返回蜂巢，将花蜜卸载到卸蜜房，卸载完成后雇佣蜂有三种可能的行为：①放弃自己发现的花蜜量不高的蜜源，变成一个不受约束的非雇佣蜂；②在招募区跳摇摆舞，招募一些待在蜂巢中跟随蜂，带领其再次返回所发现的蜜源；③不招募其他蜜蜂，继续回到原来的蜜源采蜜。

在现实生活中并不是所有的蜜蜂一开始就立刻采蜜，另外大多数蜜蜂在一次采蜜完成后都会选择回到招募区跳摇摆舞来招募更多的蜜蜂去采蜜。



狼群算法

狼群算法（Wolf Colony Algorithm）是基于狼群群体智能，模拟狼群捕食行为及其猎物分配方式，抽象出**游走、召唤、围攻三种智能行为**以及“胜者为王”的头狼产生规则和“强者生存”的狼群更新机制，提出一种新的群智能算法。

算法采用基于狼主体的自下而上的设计方法和基于职责分工的协作式搜索路径结构，通过狼群个体对猎物气味、环境信息的探知、人工狼相互间信息的共享和交互以及人工狼基于自身职责的个体行为决策最终实现了狼群捕猎的全过程模拟。

算法步骤：

Step1 初始化。设定狼群规模，搜索空间的维数，以及第 i 只人工狼的空间位置；

Step2 竞争头狼。选取适度值最好（位置最优）的 q 只人工狼作为竞争者，这 q 只人工狼在自己周围的 h 个方向进行搜索（游走）。

Step3 头狼召唤。头狼通过嚎叫发起召唤行为，召集周围的 k 只狼向头狼所在位置靠拢，这些狼奔袭途中，若狼 i 的位置优于头狼，则其作为头狼发起召唤行为；否则，狼 i 继续奔袭直到其与头狼之间的距离小于一定时加入到对猎物的围攻行为。

Step4 围攻猎物。将距离猎物较近的狼与头狼联合对猎物进行围攻，并将距离猎物最近的头狼的位置视为猎物的移动位置。

Step5 狼群更新机制。将狼群捕获的猎物按照“由强到弱”的原则进行分配，最终导致最弱小的狼被饿死，较强壮的狼能够继续生存下去，如此使狼群具有更好的适应能力。在算法中移除目标函数最差的 u 只人工狼，然后通过随机生成新的人工狼，使得该算法不易陷入局部最优。



烟花算法

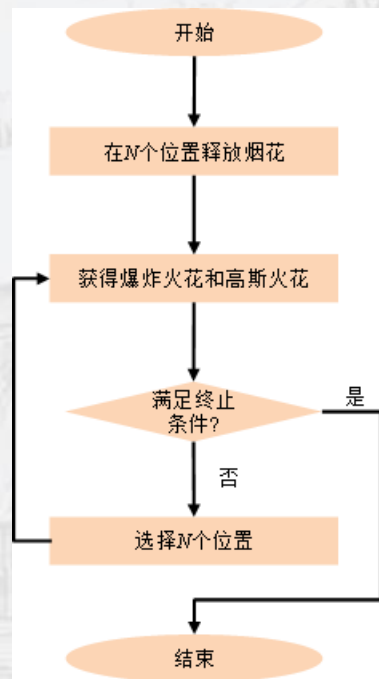
烟花算法 (Fireworks Algorithm, FWA) 由北大教授谭营在2010年提出, 是受到烟花在夜空中爆炸产生火花并照亮周围区域这一自然现象的启发。

对于一个最优化问题, 尤其自变量是连续空间的最优化问题, 需要在整个解空间内迅速有效地寻找到最优解, 在烟花算法中, 烟花被看作为解空间中一个可行解, 那么烟花爆炸产生一定数量火花的过程即为其搜索邻域的过程。烟花算法中烟花之间进行的资源交互 (爆炸火花数目和爆炸半径) 使得烟花算法成为一种新型的群体智能算法。



算法包括以下几个步骤:

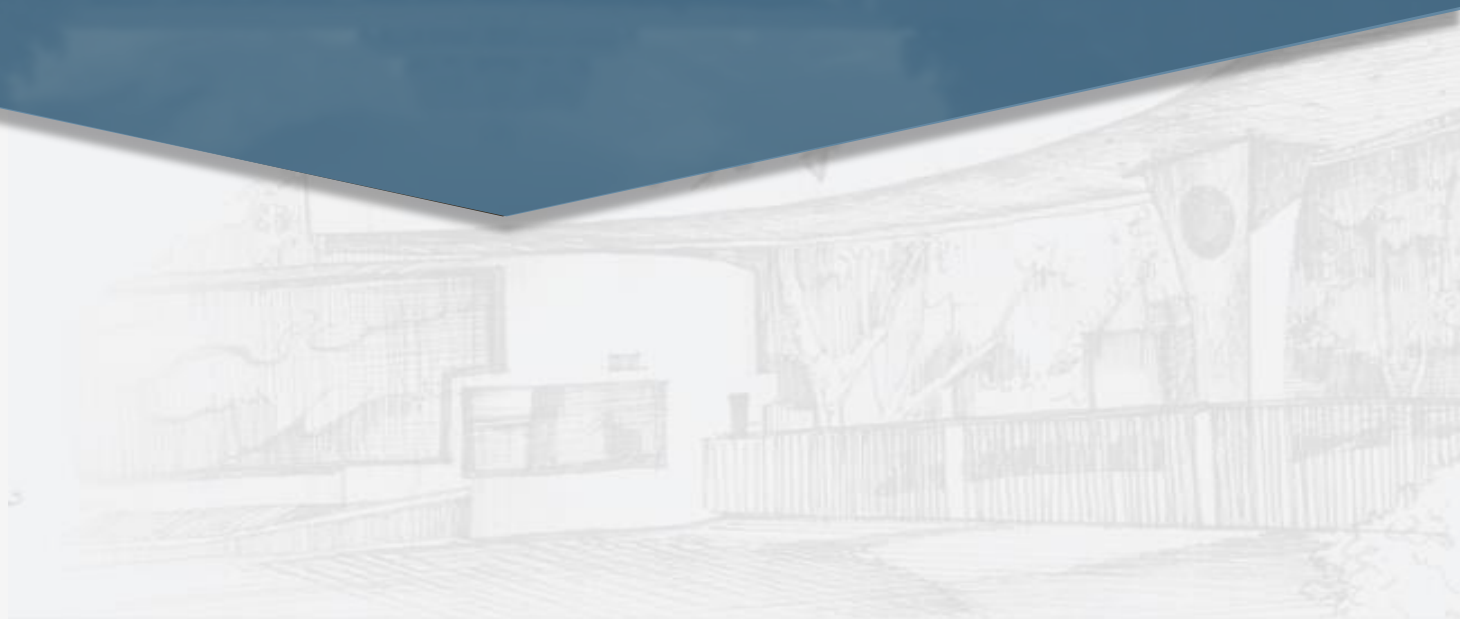
- ① 在可行解空间中随机产生一定数量的烟花, 每个烟花代表解空间中的一个可行解。
- ② 根据优化目标函数计算每个烟花的适应度值, 并据此确定烟花质量的好坏, 以在不同爆炸半径下产生不同数量的火花。算法使用两种形式的火花——爆炸火花和高斯变异火花。其中爆炸火花负责对烟花邻近区域的搜索, 适应度值好的烟花在较小的邻近区域内产生较多的火花, 反之, 适应度值差的烟花在较大的邻近区域内产生较少的火花。相对于爆炸火花, 高斯火花的引入增强了种群的多样性。
- ③ 判定是否满足终止条件。如满足则停止, 否则在爆炸火花、高斯变异火花和烟花中选择一定数量个体作为烟花进入下一代迭代。





西安交通大学
XI'AN JIAOTONG UNIVERSITY

3.5 其它优化算法





其它优化算法



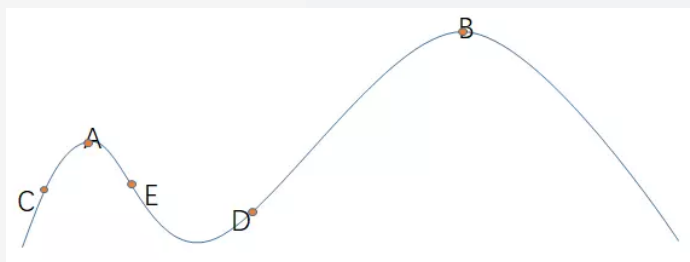
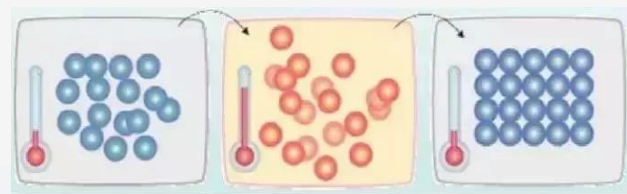
西安交通大学
XI'AN JIAOTONG UNIVERSITY

模拟退火算法

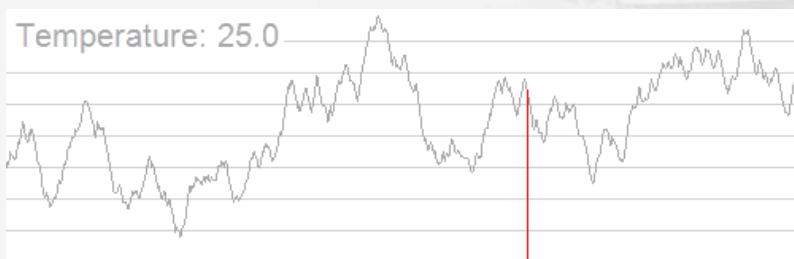
模拟退火算法（Simulated annealing, SA）是一种基于蒙特卡罗思想设计的，常用于在较大的解空间中搜索近似全局最优解的优化算法。

算法源于固体退火原理，将固体加温至充分高，再使其以足够慢的速度冷却，用原子或晶格空位移动来释放内部残留应力，通过这些原子排列重组过程来消除材料中的差排。

加温时，固体内部粒子随温度升高变为无序状，内能增大，而缓慢冷却时粒子趋于有序，在每个温度都达到平衡态，按照物理规律，最终在常温时达到基态，内能减为最小。



介绍SA前必要先介绍爬山算法，它是一种简单的贪心搜索算法，该算法每次从当前解的临近解空间中选择一个最优解作为当前解，直到达到一个局部最优解。**爬山算法实现很简单，其主要缺点是会陷入局部最优解，而不一定能搜索到全局最优解。**如图所示：假设C点为当前解，爬山算法搜索到A点这个局部最优解就会停止搜索，因为在A点无论向那个方向小幅度移动都不能得到更优的解。



SA从某较高初温出发，伴随温度不断下降，结合一定概率突跳特性在解空间中随机寻找目标函数全局最优解，即在局部最优解能概率性地跳出并最终趋于全局最优。**“一定的概率”的计算参考了金属冶炼的退火过程**，这也是模拟退火算法名称的由来。

将温度 T 当作控制参数，目标函数值 f 视为内能 E ，而固体在某温度 T 时的一个状态对应一个解，然后算法试图随着 T 的降低，使目标函数(内能)也逐渐降低，直至趋于全局最小值（退火中低温时的最低能量状态），就像金属退火过程一样。



天牛须搜索

天牛须搜索（Beetle antennae search, BAS）算法于2017年被提出，它是一种模仿天牛觅食行为的生物启发式算法，通过**模拟天牛触须的检测功能和天牛的随机游走机制**，实现了一种类似天牛觅食过程的寻优机制。

在天牛觅食过程中，食物气味吸引天牛向着食物前进。天牛通过两只触角对气味进行感知，且根据食物距离两只触角的远近不同，**两只触角所感知的气味浓度也有所差异**。当食物处于天牛左侧时，左侧触角感知的气味浓度强于右侧触角感知的气味浓度，天牛根据两只触角所**感知的浓度差，向着浓度强的一侧随机前进**。通过一次次迭代，最终找到食物的位置。



算法的流程：

- ① **初始化搜索天牛**：在允许范围内随机设置该天牛的初始质心位置 x^0 、两须之间的距离 d^0 以及左须到右须的矢量方向 $\vec{b} = \frac{rands(k,1)}{|rands(k,1)|}$ 。
- ② **计算左须和右须的位置**： $x_l^t = x^t + \frac{1}{2}d^t \cdot \vec{b}$ ， $x_r^t = x^t - \frac{1}{2}d^t \cdot \vec{b}$ 其中， d^t 表示天牛两须之间的距离，与步长之间有如下关系： $\delta^t = c_2 \cdot d^t$ ， δ^t 表示第 t 次迭代的步长， c_2 为一个常数。
- ③ **计算天牛的适应度函数**：分别计算左须和右须对应的目标函数。
- ④ **更新天牛的质心位置**： $x^{t+1} = x^t + \delta^t \cdot \vec{b} \cdot \text{sign}(f(x_r^t) - f(x_l^t))$ 如果 $f_l < f_r$ ，则天牛向左转；如果 $f_l > f_r$ ，则天牛向右转。
- ⑤ **更新步长**： $\delta^{t+1} = c_1 \delta^t$ $d^{t+1} = \frac{\delta^{t+1}}{c_2} = \frac{c_1 \delta^t}{c_2}$ 其中， c_1 为常数，这种更新方法可以保证在初始情况下步长较长，随着迭代的发生步长逐渐减小。
- ⑥ **检查终止条件**：如果未达到设定条件（预设误差或者迭代的次数），则返回第②步。



果蝇优化算法

果蝇优化算法（Fruit Fly Optimization Algorithm, FOA）于2011年由台湾亚东技术学院潘文超受**果蝇觅食行为**启发而提出，该算法优点在于计算过程简单、易于编码实现和理解等。

果蝇是生物学研究中最重要模式生物之一，20世纪初Morgan选择黑腹果蝇作为研究对象，通过简单的杂交及子代表型计数的方法，**建立了遗传的染色体理论**，奠定经典遗传学的基础并开创利用果蝇作为模式生物的先河。

果蝇在感知方面优于其他物种，尤其在嗅觉和视觉方面。果蝇嗅觉器官能发现空气中漂浮的各种气味，甚至能闻到40公里外的食物。然后，当它接近食物位置，也可以用它灵敏的视觉找到食物和同伴聚集的位置，并朝那个方向飞行。



算法步骤：

- ① **初始化**：首先随机初始化果蝇种群位置；
- ② **食物搜索**：通过嗅觉给出果蝇寻找食物的随机方向和距离；
- ③ **计算味道浓度判定值**：由于无法得知食物位置，先估计与原点的距离，再计算味道浓度判定值；
- ④ **适应度评估**：将味道浓度判定值代入味道浓度判定函数或称为适应度函数（fitness function），用来求出果蝇个体位置的味道浓度；
- ⑤ **确定最优个体**：找出该果蝇群体中味道浓度最低的果蝇(最优个体)；
- ⑥ **飞行**：记录并保留最佳味道浓度值与其x、y坐标，此时果蝇群体利用视觉向该位置飞去；
- ⑦ **循环**：进入迭代寻优，并判断味道浓度是否优于前一迭代味道浓度。



食肉植物算法

食肉植物算法（carnivorous plant algorithm, CPA）是由马来西亚的Ong Kok Meng，于2020年受**食肉植物如何适应恶劣环境**（比如捕食昆虫和传粉繁殖）的启发而提出的。

大多数植物是动物的直接食物来源，但是自然界中也有例外，食肉植物就是其中一种。它们通常**生长在缺乏营养的恶劣环境中**，例如，湿地、沼泽和以水丰富为特征的泥泞海岸。为了获取生长和繁殖所需的氮和磷等额外的营养，食肉植物用它们**分泌的酶吸引、诱捕和吃掉**苍蝇、蝴蝶、蜥蜴和老鼠等动物。

CPA从随机一组解开始，然后这些**解被分为食肉植物和猎物**，再按生长和繁殖过程分组，进行适应度值的更新，最后将所有解合并。



算法步骤：

- ① **初始化**：即在湿地中随机初始化 n 个食肉植物和 m 个猎物个体的位置；
- ② **分类和分组**：个体按照其适应度升序排序（考虑最小化问题），排在最前面的 n 个解作为食肉植物，剩余 m 个解是猎物（默认猎物是多余食肉植物的）。分组是为了减少食肉植物生长过程中出现大量劣质猎物的可能性，这对提高食肉植物的生存能力至关重要。
- ③ **成长（探索）**：由于土壤营养不良，食肉植物会吸引、诱捕和消化猎物来生长。这种植物的香味能引诱猎物，但猎物也能偶尔成功地从食肉植物的魔爪中逃脱，所以这里引入了一个吸引率。每一种群都随机选择一个猎物，如果吸引率高于随机生成的数字，食肉植物就会捕获猎物并消化来生长。
- ④ **繁殖（利用）**：食肉植物吸收猎物的营养，并利用这些营养生长和繁殖。在繁殖方面，只有排名第一的食肉植物，即种群中最好的解，才允许繁殖。这是为了确保只关注最优解，从而避免对其他解进行不必要的利用，节省计算成本。
- ⑤ **适应度更新和合并**：将新生成的食肉植物和猎物与先前的种群进行合并，就得到了一个新种群，且按照适应度值升序排序，选择排名前 n 的个体作为新的候选解。这种精英选择策略保证了选择更优的解用于下一代的繁殖。
- ⑥ **停止准则**：重复整个分类、分组、生长和繁殖的过程，直到到达终止停止准则。



西安交通大学
XI'AN JIAOTONG UNIVERSITY

谢谢大家

