中国科学技术大学
University of Science and Technology of China

人工智能讲义

优化问题

March 9, 2022

Outline

优化问题描述

优化问题 $\min f(x)$

$$s.t.$$

$$h(x) \leq 0$$

说明

- $f(x)$ 目标函数
- $h(x) < 0$，约束条件，限制 x 的取值范围，可以没有约束条件

优化问题形式化为搜索问题

搜索问题是通用模型: 优化问题可建模为搜索问题

形式化方法 1:

- 状态: 一个解当成一个状态，解集构成状态空间
- 后继函数: 无
- 目标状态: 由目标函数的约束形式定义（最小化/最大化）
- 初始状态: 任意指定
- 路径耗散: 计算目标函数一次，代价为 1
- 解: 终态

形式化方法 2:

- 状态: 解由多个部分构成时，一个部分解或完整解是一个状态（参考 8 皇后问题的形式化方法 2）
- 后继函数: 无
- 目标状态: 由目标函数的约束形式定义（最小化/最大化）
- 初始状态: 任意指定
- 路径耗散: 计算目标函数一次，代价为 1
- 解: 完整解/终态

部分解和完全解

- 若 x 是一个向量，或多个部分构成
- x 所有分量都赋值了，则称 x 是完全解，否则称 x 是部分解

优化搜索问题

当作搜索问题的优化问题，特点：

- 优化搜索问题的解，不需要返回从初态开始的路径，返回终态即可。（初态不唯一）；
- 这降低了对解的要求；
- 后继函数在搜索问题中可以在问题建模时，考虑求解算法的便利和性能而特别设计；

例子：形式化方法 1

数值优化问题：$\min\ f(x) = ax^2 + bx + c, a > 0$

- 状态：用 x 来标识，任意一个解代表一个状态，可以实数编码或二进制编码 x
- 后继函数：
  - 若采用爬山法来求解问题，邻域就是后继函数；
  - 若用梯度下降法来求解问题，任意 x 的后继由其梯度信息和步长参数确定；
- 目标状态：x 的取值使得目标函数 f 取最小值；
- 初态：任意指定；
- 路径耗散：每评估/计算一个状态的目标状态值，路径耗散为 1。
- 解：使得 f 最小的 x

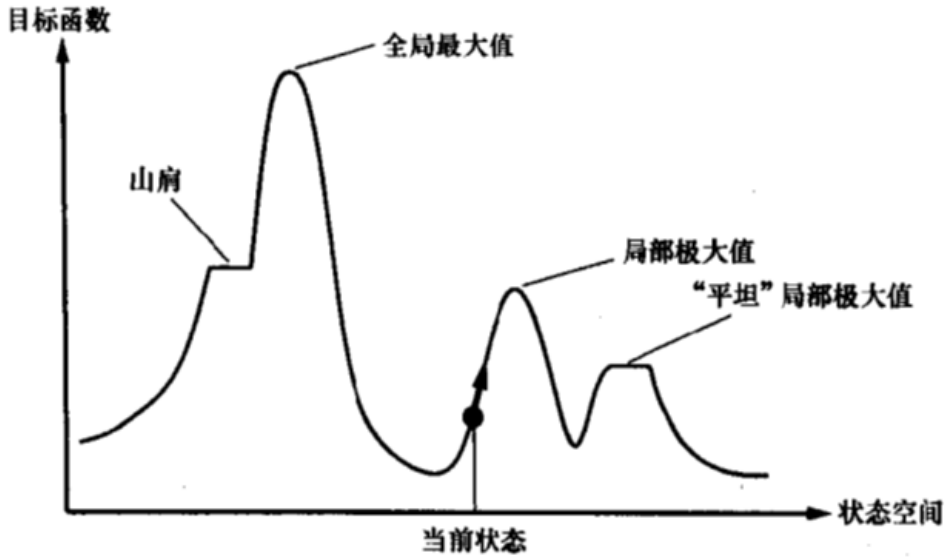
例子：形式化方法 2

组合优化问题：MaxSAT

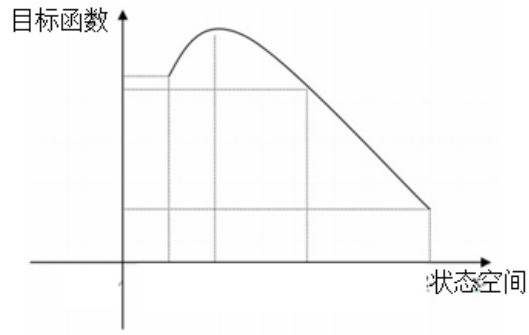
n 个布尔变量 X，m 个定义在 X 上的布尔表达式，求 X 的一个赋值，使得能让 m 个布尔表达式尽可能多地被满足 (为真)。

- 状态：任何一种 0 个或 $k \leq n$ 个布尔变量被赋值的情况为一个状态；
- 后继函数：如采用爬山法，定义两个解的 n 个布尔变量中只有一个布尔变量赋值不一样时，二者互为后继；
- 初态：任意指定；
- 目标状态：满足让最多布尔表达式为真的赋值方法；
- 路径耗散：每评估一次所有的布尔表达式，代价为 1.

优化问题：地貌图



编码？



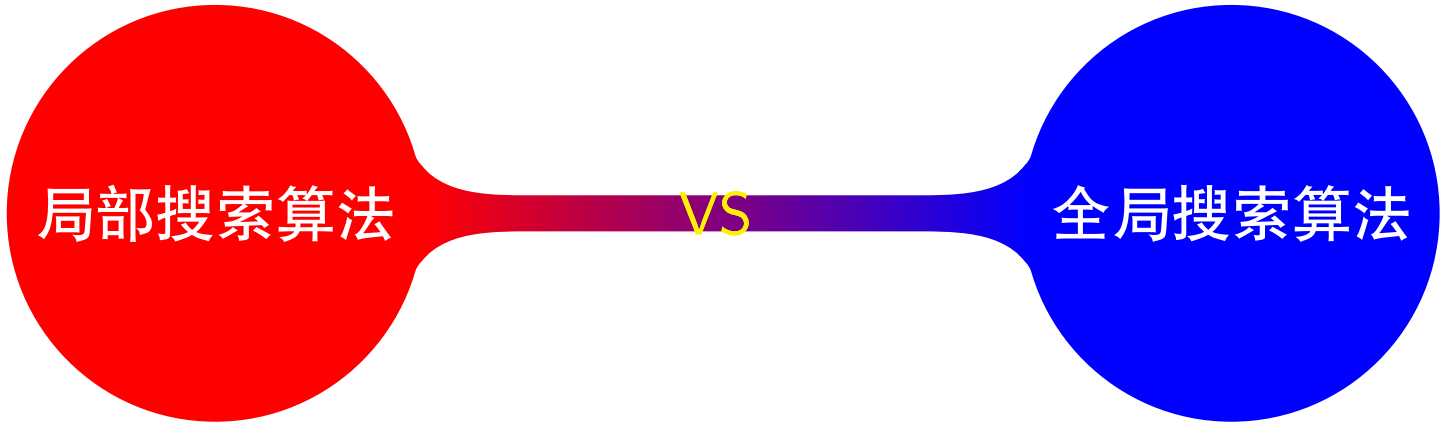
Landscape: $L = (S, f, N)$，地貌图

- 状态空间 S，编码
- 目标函数 f
- 邻居算子 N，搜索中的后继函数

建模：状态空间的编码和后继函数的设计相关、重要！

理解 Landscape首先给定一个所有状态构成的集合 S，S 是集合，包含元素之间是无序的；设计一个邻居算子/后继函数，在状态之间建立关系，形成状态图，如左图的横坐标系统，是一个“状态图”；在状态图形成的空间基础上，增加一维对每个状态赋予一个目标函数值 f，这样就得到了 Landscape!

全局和局部搜索算法



后继状态是状态空间的子集
状态图不是完全图
边权值为概率，表示状态间转移概率

后继状态为整个状态空间
状态图为完全图
边权值为 1

典型代表算法

局部搜索算法

- 爬山法、梯度下降法、禁忌搜索、局部剪枝搜索、模拟退火

全局搜索算法

- 模拟退火、遗传算法、蚁群算法、粒子群算法、差分算法、EDA(估计分布算法)、Memetic 算法

梯度下降法

梯度的概念

- n 元实值函数 $g(x)$ 在 n 维空间中变化速度最快的方向:$\nabla_x g \triangleq (\frac{\partial g}{\partial x_1}, \frac{\partial g}{\partial x_2}, \ldots, \frac{\partial g}{\partial x_n})^t$
- 站在山上，各个方向中，有个方向是最陡峭的方向

算法思想

- 想象一个人被随机扔在群山中，现在他想最快地下山到最低的山谷;
- 群山很大，视线范围无法覆盖所有区域;
- 假设下山有各个方向都可以选；如何最快下山？找最陡峭的方向(负梯度方向)！梯度方向上山最快。
- 你下山时，步幅迈多大？如果你是巨人，一步最大可以跨越两座山头？合适恰当的步幅设置很重要!
- 梯度下降也称最速下降法。

梯度下降法

第一步：初始化，设置算法相关参数/超参数 主要包括两个参数（参数具体使用参看第三步）：

- 算法停止准则 T，当执行 T 次循环时，算法终止；
- 下降步长/步幅序列，$\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_T]$。该序列一般为递减。

第二步：初始化，初态 X_0

- 初态 X_0 可设置为某个特殊值
- 也可以用随机值

第三步：循环迭代

- $for\ s = 1, 2, ..., T$
  - $X_{s+1} = X_s - \lambda_s \nabla_X g(X_s)$, 其中
  - $\nabla_X g = (\frac{\partial g}{\partial x_1}, \frac{\partial g}{\partial x_2}, \ldots, \frac{\partial g}{\partial x_n})^t$ and $X = (x_1, x_2, \ldots, x_n)^t$
- $endfor$

梯度下降法

评述

- 不能确保获得全局最小解
  - 该优化问题面临的是“群山”
  - 梯度下降可能落入一个不是最低的山谷，获得一个极小值/局部最优解，无法确保获得全局最小解
  - 若将视线范围视为“邻域”，局部最小值是指邻域内最小，也即是视线范围内没有更低的山谷存在
- 算法停止条件和步长序列的设置需要人工经验

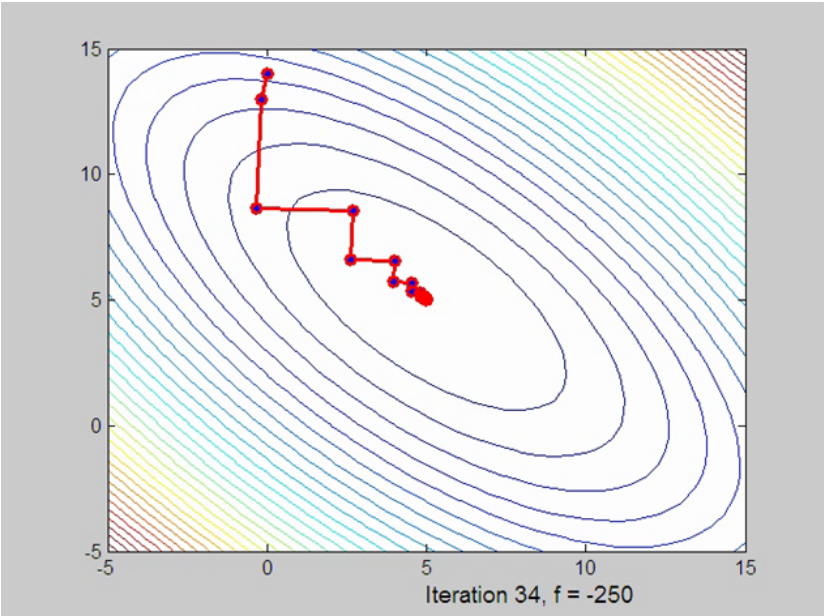
改进算法？

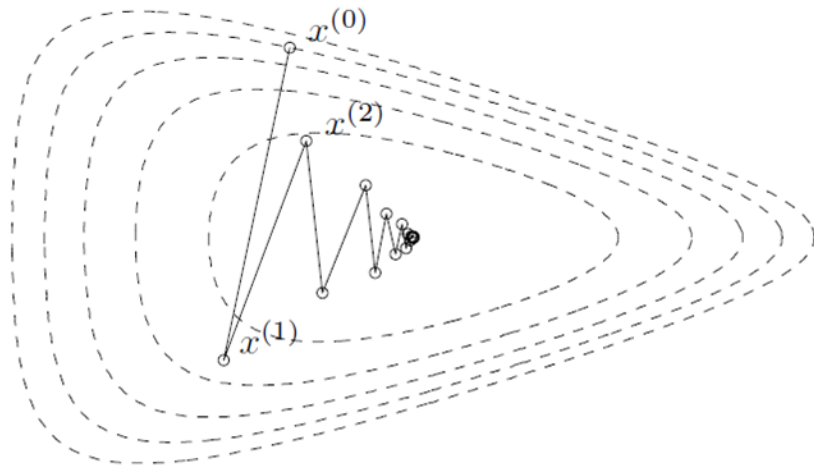
梯度下降法的改进

直线搜索：不知道如何设置步长，就用各种步长来尝试

- 不知道步长 λ 的如何取值时，妥协的做法
- $\lambda^* = \arg_\lambda \min g(X - \lambda v)$，其中 v 为梯度方向 $\nabla_x g \triangleq (\frac{\partial g}{\partial x_1}, \frac{\partial g}{\partial x_2}, \ldots, \frac{\partial g}{\partial x_n})^t$ 或其它方向。
- λ 的取值范围一般设置为一个集合 $\{\ldots, -2c, -c, c, 2c, \ldots\}$

梯度下降的例子





- 每个点的梯度方向垂直等值线；
- 步长每次都是最优的的 λ^*，如左图；
- 每次前进的方向都是垂直于上次前进方向；
- zig-zagging/锯齿现象，折线方向走向极小值点，如左图。
- 右图每一步步长不保证最优。

梯度下降法的改进

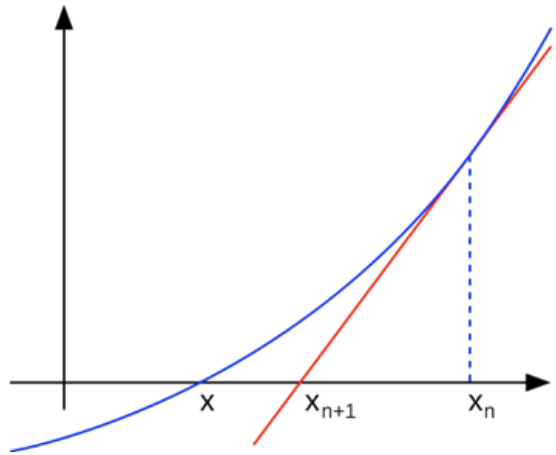
随机梯度下降引入的原因

- 计算梯度信息的时间代价如果很大，怎么办？
- 算法的每一步，即每一次循环，更新一次 W，都需要完全扫描一次训练数据集，即：
  $W_{s+1} = W_s - \lambda_s \nabla_W (\frac{\sum_{(x,y)\in D_{train}} (W_s \cdot \phi(x) - y)^2}{|D_{train}|}) =$
  $W_s - \lambda_s \frac{\sum_{(x,y)\in D_{train}} 2(W_s \cdot \phi(x) - y)\phi(x)}{|D_{train}|}$
- 当训练数据集很大时，包括千万/上亿的行时，算法的时间开销巨大。
- 数据挖掘工程实践中常考虑这类问题。此时训练数据集太大，无法一次载入内存，涉及内外存数据交换，时间开销更大，因此需要对算法进行改进。

随机梯度下降法

- 计算梯度方向时，只用了一个随机样本的信息，替代经典梯度下降算法要计算所有训练数据集对梯度方向的贡献。
- 即：$W_{s+1} = W_s - \lambda_s \frac{\sum_{(x,y)\in D_{train}} 2(W_s \cdot \phi(x) - y)\phi(x)}{|D_{train}|} \Longrightarrow W_{s+1} =$
  $W_s - \lambda_s \nabla_W ((W_s \cdot \phi(x_i) - y_i)^2)$
- 时间性能得到提升，不需要每次循环都载入一次训练数据集；付出的代价是更多的循环次数。

牛顿法



寻找函数 $g(X)$ 的根 X^*，使得 $g(X^*) = 0$

- 找 $g(X)$ 在 X_n 处的切线，以切线和 X 轴的交点为新点 X_{n+1}
- $\nabla g(X_n) = \frac{g(X_n)-0}{X_{n+1}-X_n} \Rightarrow X_{n+1} = X_n - \frac{g(X_n)}{\nabla g(X_n)}$

问题：

- 算法能快速对可导函数求根
- 但是和优化问题有何相关之处？
- 离散的组合优化问题怎么办？

爬山法

爬山法: 最小化目标函数 $g(X)$

Input: 邻居/邻域算子: $N(X)$; 目标函数 $g(X)$
Output: X_c: 代表解的终态
随机生成初态, 记为 X_c /* 初始化 */;
while T **do**
    在 $N(X_c)$ 中找到子集 $N'(X_c) = \{Y | g(Y) \leq g(X_c)\}$;
    if $empty(N'(X_c)$ **then**
        return X_c
    end
    ;
    依据策略 s 选择 $X' \in N'(X_c)$;
    $X_c \longleftarrow X'$;
end

评述: 爬山法找到的是局部最优解

- 非常简单、高效的算法，内存需求少，两个关键因素: 初始解和邻居算子 $N(\cdot)$ 的定义
- 策略 s，常见的有:
  - best-found: 在 $N'(X_c)$ 中选择 $g(\cdot)$ 最小的
  - first-found: 随机在 $N'(X_c)$ 中选择一个，之所以叫 first-found，因为实际实现上述算法时，不需要求出整个 $N'(X_c)$，在对 X_c 的邻域遍历时，遇到第一个更好的邻居时，本次循环就结束;
  - Random Walk: 在 $N(X_C)$ 而非 $N'(X_c)$ 上随机选择一个当成下次循环迭代的 X_c，一种随机采样

爬山法：改进

理解：与梯度下降的相关性

- Best-found：梯度下降的离散化版本；
- first-found：随机梯度下降；

爬山法的改进算法

- 用不同的随机初始值重启算法，对于上百万个皇后问题，不到一分钟，随机初值重启的爬山法能求解出
- 局部最优的存在是和邻域结构的定义相关，故就有“变”邻域结构结构的局部搜索算法。比如迭代爬山法，到达局部最优解后，调整邻居算子，实现“扰动”，然后再回到局部搜索。
- 模拟退火 (= 爬山法 + 随机行走)

爬山法改进：局部剪枝搜索

LBS/局部剪枝搜索: 爬山法的并行改进

- 初始时刻，开始 k 个随机种子/初始值/初态，准备执行爬山法
- 接下来的步骤，是循环迭代过程，可以有不同的策略：
  - 策略一：假设每个解有 l 个后继，每一步会在所有 lk 个后继中选择最优的 k 作为当前的 k 个解
  - 策略二：所有 lk 个后继对应一个被选择的概率分布，例如以每个后继的目标值在所有后继目标值中的占比作为该后继被选择的概率（最大化问题），然后依次概率分布，随机选择 k 个出来

模拟退火算法

模拟退火思想:

- 一定程度上解决爬山法容易陷入局部最优的困境
- 小概率允许“坏”的移动，而不是像爬山法一样，每次都选最好的移动
- 分类上，介于全局搜索和局部搜索算法之间

模拟退火: 最小化目标函数 $g(X)$

Input: 邻居/邻域算子: $N(X)$; 目标函数 $g(X)$
Output: X_c: 代表解的终态
随机生成初态, 记为 X_c /* 初始化 */;
while T **do**
  随机选择 $X' \in N(X_c)$;
  if $g(X') \leq g(X_c)$ **then**
    $X_c \longleftarrow X'$
  end
  以概率 p: $X_c \longleftarrow X'$;
end

模拟退火算法的参数

概率 p

- 概率 p 的设计思想来自冶金学:
- 开始时刻，温度较高，有较大概率移动到“坏”解，随着搜索的进行，温度降低，移动到“坏”解得概率指数下降;
- 公式: $p = e^{-\Delta g/T}$，其中 $\Delta g = g(X') - g(X_c)$ 是解和它的邻居的目标函数之差

温度 T

- 温度 T 的下降过程，称为 $schedule$,
- 可以证明，T 下降得足够慢，算法找到最优解的概率逼近 1

Tabu 搜索/禁忌搜索

Tabu 搜索

- 简单、带记忆的局部搜索算法
- 不是真正意义的“原子算法”，是一种策略，可以添加到各种局部搜素算法中

Tabu 搜索算法核心

- 一个长度有限的（不妨设为 k）最近历史解列表 $Tabu-list$（用队列实现）
- 在搜索过程中，发现的新解如果出现在 $Tabu-list$，直接放弃
- $Tabu-list$ 能一定程度上防止在局部最优不动了

Tabu 搜索算法问题

- $Tabu-list$ 的长度 k 多少最好？

$(1+1)-EA$

从局部搜索到全局搜索：爬山法 $\Longrightarrow$ $(1+1)-EA$

Input: 目标函数 $g(X)$
Output: X_c: 代表解的终态/最优解
随机生成初态/初始解，记为 X_c /* 初始化 */;
while T **do**
    在状态空间中随机选择一个解为 X_c 的后继状态 X' /* 爬山法是从邻居集合中选择一个后继 */;
    if $g(X') \leq g(X_c)$ **then**
        $X_c \longleftarrow X'$
    end
    ;
end

评述：$(1+1)-Evolutionary\ Algorithm$

- 算法中当前解的后继，仍可以认为是邻居算法产生的；
- 此时的邻居算子能够有一定概率一步访问到其它任何一个解，这是一个特殊的邻居算子，称为全局邻居算子
- 可能不同解之间的跳转概率不一样，一般 $(1+1)-EA$ 会“系统地”设置这样一个概率分布，比如依据 X_c, X' 的差异大小，设置相应的跳转概率

$(1+1)-EA$ 的并行化

类似于 $LS \to LBS$，λ 个 $(1+1)-EA$ 并行执行

- 当前解集中的每个解产生 μ 个后继解，在所有的 $\lambda\mu$ 个解中选择 λ 个作为下次循环的当前解集
- 选择的策略可以是：（这些算法称为：$(\lambda, \lambda\mu)-ES$）
  - 最好的 λ 个；
  - 按解质量的设置“生存概率”，以此选择 λ 个解等；
  - 随机选择 k 个解，选择其中最好的一个，重复 λ 次，锦标赛选择；
- 若选择下次循环的当前解是在本次循环和其后继，共 $\lambda+\lambda\mu$ 个解中进行，那么算法称为 $(\lambda+\lambda\mu)-ES$

EA：后继函数设计

EA: 后继函数，产生的后继，全局范围内/整个解集范围内

- 从当前解集的一个解产生一个/多个后继，可类似于生物界的无性繁殖
- 从当前解集的两个或多个解产生一个/多个后继，可类似于生物界的有性繁殖

EA: 后继函数的典型例子

- 在当前 k 个解中，随机选择 2 个，产生“后继/孩子”，这称为“交叉”算子/crossover，一般性的演化算法/进化算法
- 在当前 k 个解中，选择 3 个 a, b, c, 计算后继为 $x = a + \zeta(b - c)$，得到差分演化算法
- 在当前 k 个解中，全部选择，用来估计一个模型/分布，该模型我们选择用来描述整个解集，解集获得的过程我们称为从该模型中进行“采样”，这就是估计分布算法，EDA

EA：后继函数设计

EA: 更多后继函数的典型例子

- 在当前 k 个解中寻找后继时，考虑 k 个解的搜索历史，找到每个解历史最佳解和 k 个历史最佳中的最佳解，以这些解为基础，设计“速度函数”，获得 k 个解的后继；该算法称为“粒子群算法”
- 在当前 k 个解中，每个解的后继是以该解为初始解，执行局部搜索（例如爬山法），得到局部极值解，以此局部极值解为后继；该算法称为“memetic 算法”

EA：状态空间设计

EA: 状态空间的设计/编码 $\rightarrow$ 遗传算法

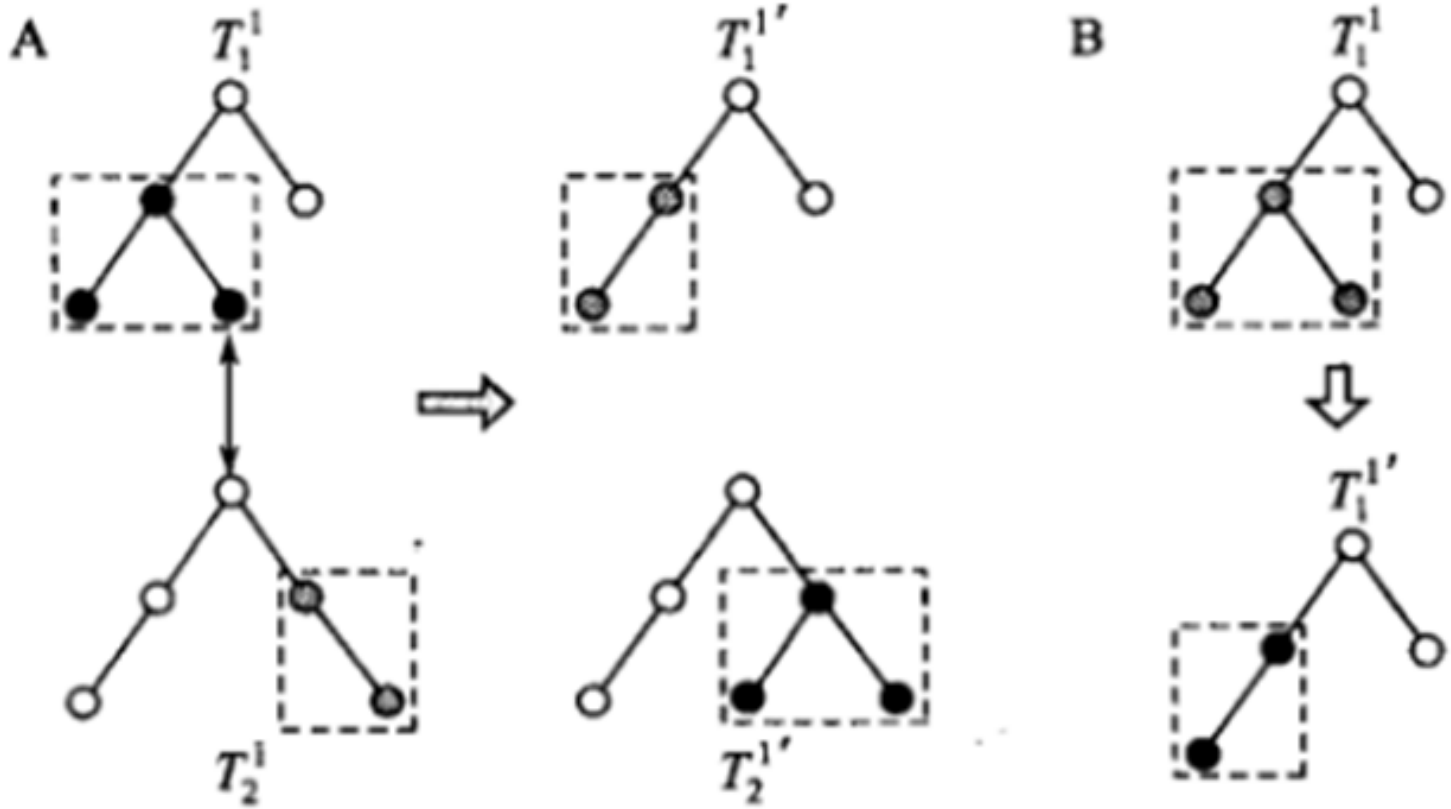
- 解/状态的表示：二进制串表示一个解
- 交叉算子：两个二进制串的混杂/分解
- 变异算子：二进制串的某些位置的概率“翻转”$(0 \rightarrow 1, 1 \rightarrow 0)$
- 选择算子：在“后继”或者“后继 + 当前解集”中用某种方式选择出“新的”当前解集。

后继函数的设计和状态空间的设计密切相关，上述二进制串的状态表示，得到“遗传算法”。

EA：状态空间设计

EA: 状态空间的设计/编码 $\rightarrow$ 遗传规划

- 解/状态的表示：树型结构表示一个解
- 寻找一个计算机程序/计算方法，达到目的



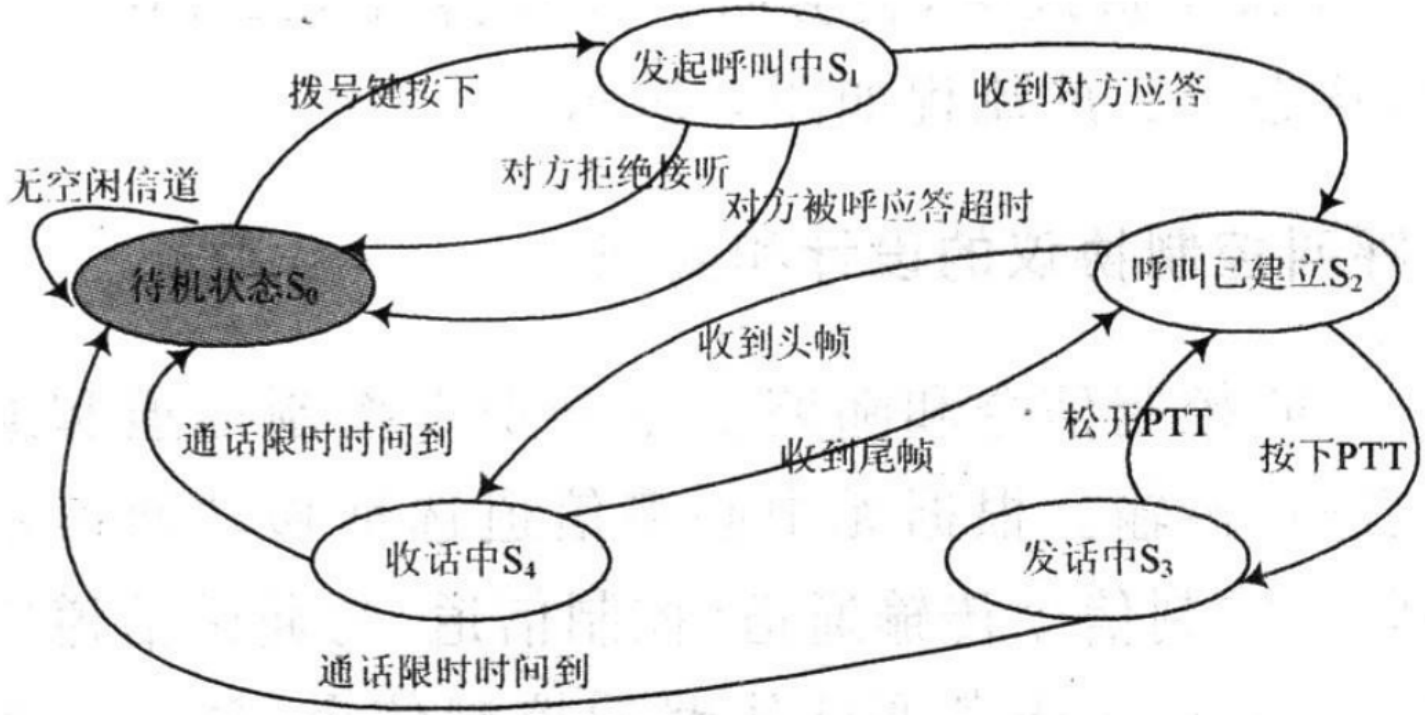
重新审视搜索问题

当采用形式化方法 2 来定义搜索问题的时候
任一状态可以用一个 $n-D$ 的向量来描述，故

- 经典搜索问题的解是从初态向量到终态向量构成的一个“向量链”，如 $MaxSAT$ 问题，路径是从没有任何布尔变量被赋值，到所有布尔变量被赋值，长度为 $n+1$ 的向量链；
- 后继函数的设计，一般取 $n-D$ 中一维的值变化得到后继，例如 8-皇后问题
- 构造性算法：$n-D$ 的取值，一个一个地确定，得到节点指数增加的搜索树
- 路径：表示部分解和完整解

我们称这是构造性算法（解的各个部分/分量逐步被构造出来）

状态图 → 有限状态机



- 每个状态如何保存？前面讨论状态图的搜索时，我们没考虑；
- 有限状态机（FSM) 是对上述简单状态图的扩展，引入时序、输出、输入、代价和
- 状态转移等概念，得到更具体的状态图；如图例子；
- 每个状态用一个向量来描述，一个输入导致状态转移到其它状态，同时会输出，会付出一定代价；在连续的输入作用下，状态会连续的转移；

优化搜索问题的构造性算法

部分解和完整解

- 把有限状态机看成是状态图，在其上定义新的搜索问题：寻找遍历所有节点的路径，使得路径的某个属性达到最小（类似于 TPS 问题）
- 在有限状态机上执行 $A*$ 算法，$g(N)$ 是部分解的路径耗散，对解的缺失（未确定）部分进行估计（heuristic）

优化问题求解一般都针对完整解进行评价

- 能不能在优化问题求解时，考虑部分解？
- 对构成解的“成分”进行分析？
- 解的“成分”，包括成分的“取值”和成分构成解时的“次序”（考虑解是路径时，次序重要）
- 对采用形式化方法 2 的优化搜索问题，设计算法？

蚁群算法

动机

- 解的哪一个成分更重要？哪一个成分的哪个取值更重要？
- 所谓“重要”即更小的代价获得更多的“收益”，heuristic?

蚁群算法思想

- 分布式的策略来估计
- 让 n 只蚂蚁在有限状态机上爬行，并留下“信息素”，来标记爬行历史；该过程迭代重复
- 在评价好的“状态转移”边上，将会有跟多的蚂蚁在上面爬行，留下信息素；