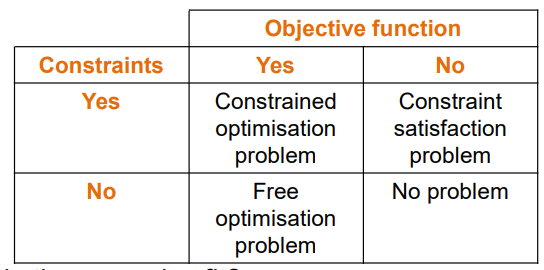
1. **优化vs约束满足**



**二、NP问题**

**1. P 类问题（简单题）**

* **特点**：计算机能快速解决的问题，时间增长不会太夸张（比如像算加法一样，数据翻倍，时间只多一点点）。
* **例子**：排序一组数字、找最短路线。
* **总结**：能高效搞定的问题。

**2. NP 类问题（验证题）**

* **特点**：计算机可能找不到快速解法，但如果有人告诉你一个答案，你能快速检查这个答案对不对。
* **例子**：数独游戏（填数字很难，但检查对不对很简单）、旅行商问题（验证某条路线是不是最短的）。
* **总结**：找答案难，验答案易。

**3. NP完全问题（终极难题）**

* **特点**：NP问题里最难的“大BOSS”，所有NP问题都能转化成它来解决。如果能解决一个NP完全问题，就能解决所有NP问题。
* **例子**：逻辑电路设计（SAT问题）、拼图游戏（所有碎片必须完美匹配）。
* **总结**：NP里的天花板，解决它≈解决所有NP问题。

**4. NP难问题（变态题）**

* **特点**：比NP完全问题还要难，甚至可能连验证答案都不容易！它们不一定是NP问题，但难度不低于NP完全问题。
* **例子**：旅行商问题的优化版（不光要验证路线，还要找绝对最短的路线）。
* **总结**：难到爆炸，验证答案都可能要很久。

**关系总结**

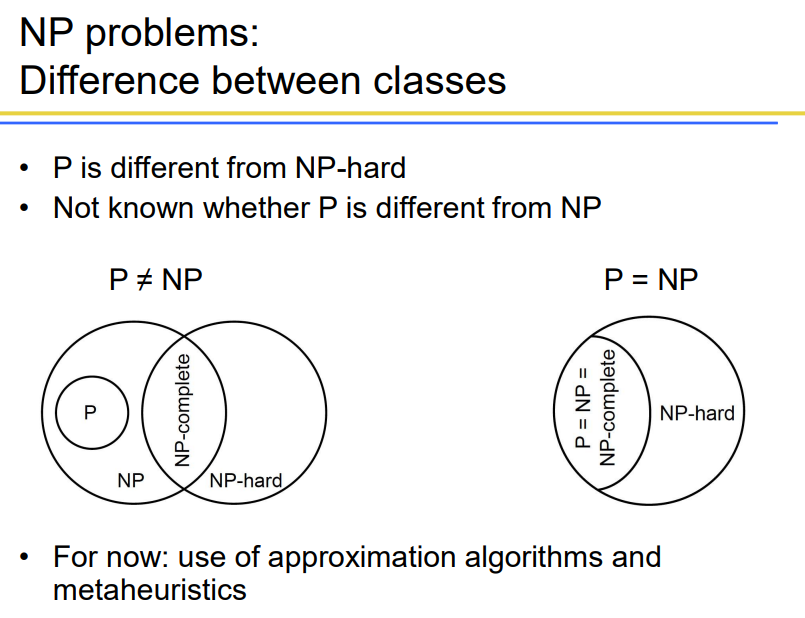
* **P ⊆ NP**：所有简单题（P）都属于验证题（NP）。
* **NP完全**：是NP问题里最难的一批，也是NP和NP难的交集。
* **NP难**：可能不是NP问题，但比NP完全还难。

**通俗比喻**：

* **P**：你能1分钟解开一道数学题。
* **NP**：别人给你答案，你1分钟能看懂答案对不对，但自己解可能要1年。
* **NP完全**：这类题是所有“NP题”的母题，能解它就能解所有NP题。
* **NP难**：解这种题可能需要1万年，甚至看不懂答案对不对。

**关键区别**：

* NP完全问题属于NP（能快速验证答案），NP难问题不一定能快速验证答案。



**三、主进化算法组件：**

**1.表示**

编码：表型=>基因型（不一定一一对应）

解码：基因型=>表型（必须一一对应）

比如：用二进制代码表示整数值。

* **基因型空间**（编码）：如 10010
* **解码后**：对应整数 40

**编码**就像把现实中的解翻译成代码（比如把25℃写成二进制），但不同代码可能对应同一个解（比如 10010 和 01001 都对应25℃）。  
**解码**则是反过来，必须确保每个代码只能对应唯一解（比如 10010 必须解码成25℃，不能是其他值）。

**2.评估（适应度）函数**

* 为每个表型分配一个实数值的适应度，作为选择的基础。
  + 适应度值区分度越高越好（比如分数差异越大，选择越容易）。

**3.种群**

* 选择作用于种群层面（比如选父母），变异作用于个体层面（比如基因变化）。

种群是允许重复的个体集合，是进化的基本单位

**4.选择机制**

适应度高的个人容易被选中

**5.变异算子和重组算子mutation and recombination**

* **分类**：根据输入数量（arity）分为两种：
  + **单输入（arity=1）**：变异算子（比如把一个解稍微改动一下）。
  + **多输入（arity>1）**：重组算子（比如把两个解“混合”生成新解）。
    - 最常见的重组是双输入（称为“交叉”，像父母基因传给子女）。
    - 输入多于2的情况理论上存在，但实践中很少用。
* **争议**：过去有人争论“重组”和“变异”哪个更重要，现在多数进化算法同时用两者。
* **为什么同时用变异和重组**？
  + 重组能结合不同解的优势，但可能陷入局部最优。
  + 变异能跳出局部最优，增加多样性。

**进化算法的主要类型（历史分类）：**

* **遗传算法（GA）**：用二进制字符串表示解（如“1010”代表开关配置）。
* **进化策略（ES）**：用实值向量表示解（如[1.2, 3.4]代表参数组合）。
* **进化规划（EP）**：用有限状态机表示解（如自动控制系统的状态转换）。
* **遗传编程（GP）**：用LISP语法树表示解（如自动生成程序代码）。

**What is an Evolutionary Algorithm?** 标题：什么是进化算法？

* Scheme of an EA进化算法的方案
* Main EA components: 主要进化算法组件：
  + Representation / evaluation / population表示 / 评估 / 种群
  + Parent selection / survivor selection父代选择 / 幸存者选择
  + Recombination / mutation重组 / 变异
* Examples: eight-queens problem示例：八皇后问题
* Typical EA behaviour典型的进化算法行为
  + EAs and global optimisation EA和全局优化
  + EC and neighbourhood search EC和邻域搜索

SGA steps：

Encoding——Initial population——Decoding——Computing fitness proportions——Crossover——Mutation

全局优化就像在迷宫中找出口，目标是找到最短路径（最优解）。

* **确定性方法**：像用地图一步步排除错误路线，最终保证找到出口，但耗时极长。
* **启发式方法**：像随机尝试几条路线，可能很快找到出口，但无法确定是否最短。  
  两种态度反映了理论完美主义（必须找到绝对最优解）与现实实用主义（快速找到可用解）的权衡。

Hill-Climbers: – Problems often exhibit many local optima – Often very quick to identify good solutions

* **EA的特点：**
  + **使用种群（多解并行搜索）。**
  + **使用随机搜索算子（如变异、重组）。**

**Characterization of EA:**

**Use of populations (parallel search with multiple solutions).**

**Use of stochastic search operators (e.g., mutation, recombination).**

* **EA的改进：**
  + **派一群人去不同位置同时探索（种群多样性）。**
  + **随机交换位置信息（重组）或随机改变路线（变异），避免集体困在局部山峰。**
* **EA的“邻域”：不依赖固定邻域结构，而是通过种群和随机操作探索更大范围。**

**Global: Through population and recombination, the “neighborhood” of EA can cover multiple regions of the solution space and even enable long-distance exploration.** **全局性：通过种群和重组，EA的“邻域”可覆盖解空间的多个区域，甚至实现远距离探索。**

**突变：Pm is called the mutation rate，Typically between 1/pop\_size and 1/ chromosome\_length，Mutation can cause variable effect (use gray coding)**

1. **point crossover：More likely to keep together genes that are near each other – Can never keep together genes from opposite ends of string– This is known as Positional Bias**

**Initialization的fitness一般不高，所以点会最低。**

**交叉还是变异？**

* **交叉的独特作用：只能结合两个父代的信息。**
* **变异的独特作用：只能引入全新的信息（比如新基因）。**
* **等位基因频率：交叉不会改变种群中某个基因的频率（比如种群中50%是0，交叉后还是50%）。**
* **关键结论：想找到最优解，往往需要一次“幸运”的变异。**
* 交叉是探索性的，像在两个父代的区域之间“跳远”。
* 变异是利用性的，像在父代附近“随机溜达”。

1. **变异顺序：  
   先调整“步长”（𝜎→𝜎′*σ*→*σ*′），再用新步长生成新解（𝑥→𝑥′*x*→*x*′）。**
2. **数学本质：**
   * **步长变异：*σ*′=*σ*⋅*e(τ*⋅*N*(0,1))⋅随机数*σ*′=*σ*⋅*eτ*⋅随机数（类似“按当前步长比例调整”）。**
   * **解参数变异：*x*′=*x*+*N*(0,*σ*′)，扰动幅度由 𝜎′*σ*′ 控制。**
3. **核心逻辑：**
   * **好的 𝜎′*σ*′ 能生成好的 𝑥′*x*′，因此两者需绑定评估。**
   * **顺序颠倒会导致步长与解脱节，失去自适应性。**

**示例：  
假设当前 𝜎=1*σ*=1，𝑥=5*x*=5。**

1. **变异 𝜎*σ*：𝜎′=1⋅𝑒0.1⋅0.3≈1.03*σ*′=1⋅*e*0.1⋅0.3≈1.03（略微增大）。**
2. **用 𝜎′*σ*′ 变异 𝑥*x*：𝑥′=5+𝑁(0,1.03)*x*′=5+*N*(0,1.03)，若随机数为 0.5，则 𝑥′=5.515*x*′=5.515。**
3. **若 𝑥′=5.515*x*′=5.515 的适应度更高，说明 𝜎′=1.03*σ*′=1.03 是更优的步长，会被保留到下一代。**

**实际意义：**

* **自动适应不同阶段：**
  + **初期步子大，快速探索全局；后期步子小，精细调整。**
* **不用手动调参：**
  + **传统算法需要人为设定步长，但自适应变异让算法自己学会“该用多大步子”。**