进化计算导论

卜晨阳

邮箱：chenyangbu@hfut.edu.cn
计算机科学与技术学院
合肥工业大学

目录：

最优化问题定义及分类

- 最优化问题的求解模型如下公式所示

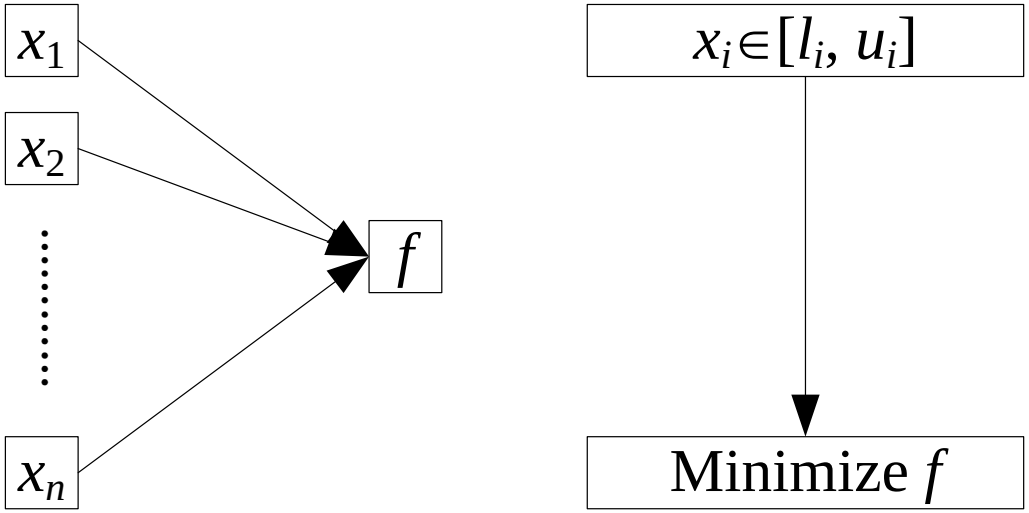
$$\text{Min } f(X),\ X \in D$$

  - 其中D是问题的解空间，X是D中的一个合法解。一般可将X表示为$X = (x_1, x_2, \ldots, x_n)$，表示一组决策变量
  - 最优化问题就是在解空间中寻找一个合法的解X（一组最佳的决策变量），使得X对应的函数映射值$f(X)$最小（最大）

- 最优化问题的分类

分类标志	变量个数	变量性质	约束条件	极值个数	目标个数	函数关系	问题性质	时间变化
类型	单变量	连续	无约束	单峰	单目标	线性	确定性	静态
		离散					随机性	
	多变量	混合	有约束	多峰	多目标	非线性	模糊性	动态

最优化问题定义：



- 例如：

$$f(x) = \sum_{i=1}^{n} x_i^2$$

其中，***n*=30**表示问题空间的维数，$x_i \in$ **[-100,100]**是定义域，这个函数的最小值为**0**

- 这是一个最简单的函数优化问题

最优化问题分类：连续优化问题

- 很多科学实验参数配置和工农业生产实践都需要面临这种类型的最优化问题
  - 例如设计神经网络的过程中，需要确定神经元节点间的网络连接权重，从而使得网络性能达到最优
- 在这种问题中，需要优化的变量的取值是某个连续区间上的值，是一个实数。各个决策变量之间可能是独立的，也可能是相互关联、相互制约的，它们的取值组合构成了问题的一个解
- 由于决策变量是连续值，因此对每个变量进行枚举是不可能的。在这种情况下，必须借助最优化方法对问题进行求解

最优化问题分类：组合优化问题

- 组合优化问题的决策变量是离散取值的
- 很多离散组合优化问题都是从运筹学（**Operations Research，OR**）中演化出来的
- 组合优化其所研究的问题涉及到信息技术、经济管理、工业工程、交通运输、通信网络等众多领域，在科学研究和生产实践中都起着重要的作用
  - 最短路径问题
  - 最小费用流问题
  - 运输问题
  - ……

组合优化问题

- 经典组合优化问题
  - 旅行商问题（Traveling Salesman Problem，TSP）
  - 0-1背包问题（Zero/one Knapsack Problem，ZKP/0-1KP/KP）
  - 加工调度问题(Scheduling Problem，如Flow-shop，Jop-shop)
  - 装箱问题（Bin Packing Problem）
  - 图着色问题（Graph Coloring Problem）
  - 最小生成树问题(MSTP，Minimum Spanning Tree Problem)
  - ……
- 当问题规模*n*比较大时，用枚举方法所需时间太大，我们借助智能优化计算方法，可以在合理的时间内求解得到令人满意的解，从而满足实践的需要

目录：

可求解与难求解问题

现实世界中的问题分类

- 计算机在有限时间内能够求解的(可求解问题)
- 计算机在有限时间内不能求解的(难求解问题)
- 计算机完全不能求解的(不可计算问题)

问题的计算复杂性

计算复杂性是指问题的一种特性，即利用计算机求解问题的难易性或难易程度，其衡量标准：

- 计算所需的步数或指令条数(即时间复杂度)
- 计算所需的存储空间大小(即空间复杂度)

----通常表达为关于问题规模n的一个函数 O(f(n))

可求解与难求解问题

$O(n^3)$与$O(3^n)$的差别

$O(n^3)$	$O(3^n)$
0.2秒	4×10^{28}秒=1015年
注：每秒百万次，n=60	

$O(n!)$与$O(n^3)$的差别

问题规模n	计算量
10	10！
20	20！
100	100！
1000	1000！
10000	10000！

$20! = 1.216\times10^{17}$

$20^3 = 8000$

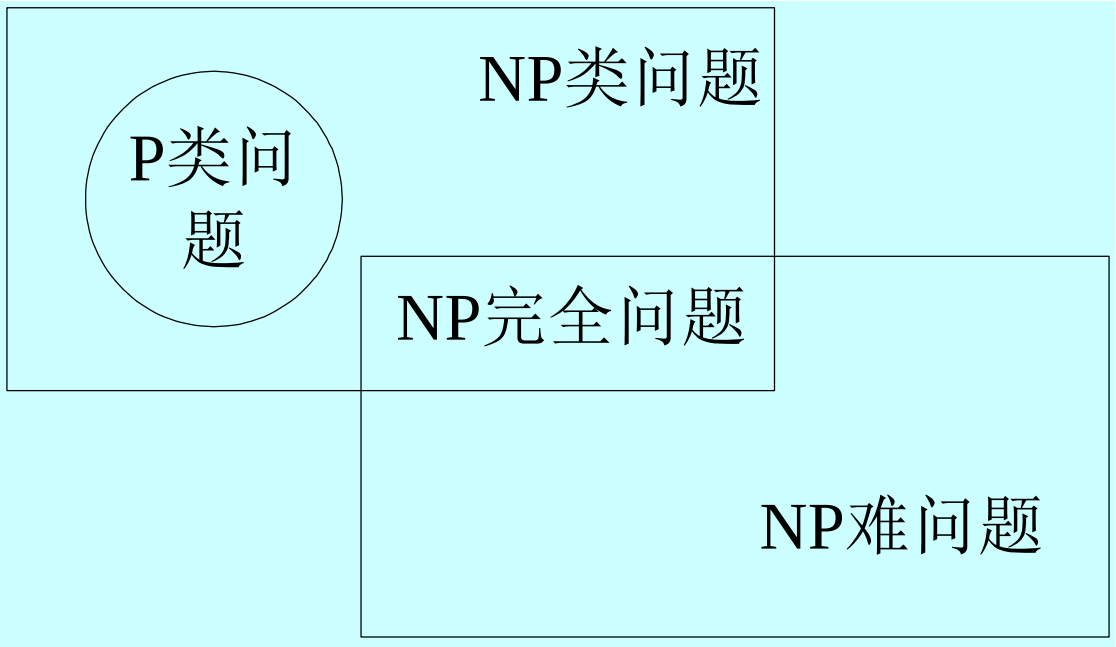
O(1), O(log n), O(n), O(n log n), $O(n^b)$ $O(b^n)$, O(n!)

- 时间复杂度并不是表示一个程序解决问题需要花多少时间，而是当问题规模扩大后，程序需要的时间长度增长得有多快

NP理论

- **P类问题（Polynomial Problem）：**

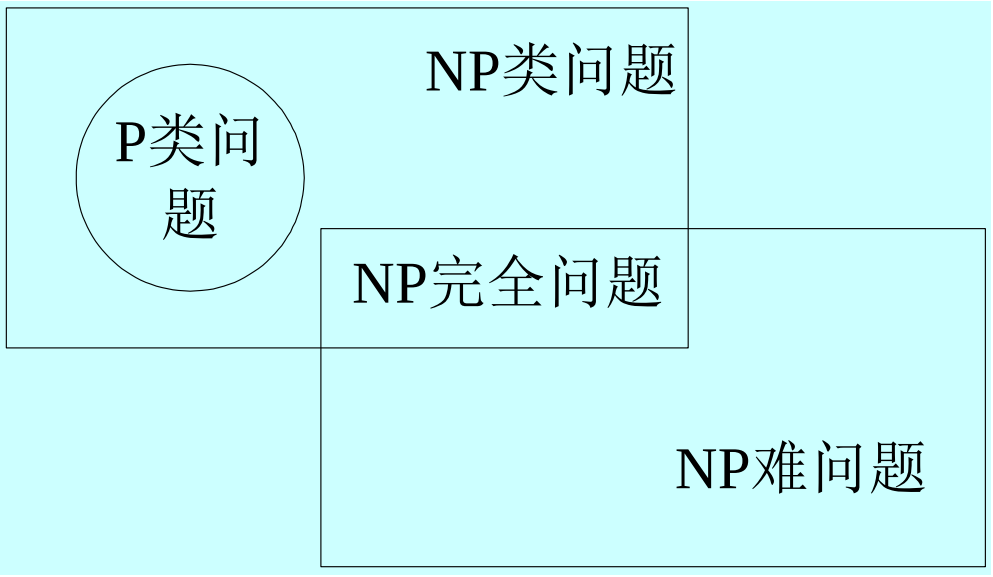
  是指一类能够用确定性算法在多项式时间内求解的判定问题。其实，在非正式的定义中，我们可以把那些在多项式时间内求解的问题当作P类问题。



NP理论

- **NP类问题（Non-deterministic Polynomial Problem）**

NP类问题是指一类可以用不确定性多项式算法求解的判定问题。例如旅行商问题的判定版本就是一个NP类问题。我们虽然还不能找到一个多项式的确定性算法求解最小的周游路线，但是可以在一个多项式时间内对任意生成的一条“路线”判定是否是合法（经过每个城市一次且仅仅一次）。比较P类问题和NP类问题的定义，我们很容易得到一个结论：P⊆NP。



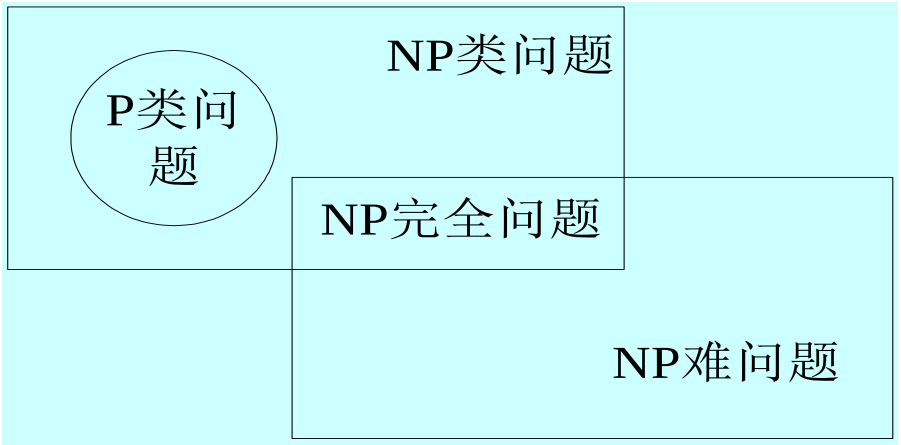
NP理论

- **NP完全问题（NP Complete Problem）**
  - NP中的某些问题的复杂性与整个类的复杂性相关联
  - 这些问题中任何一个如果存在多项式时间的算法,那么所有NP问题都是多项式时间可解的
  - 这些问题被称为NP-完全问题(NPC问题)

我们称一个判定问题*D*是NP完全问题，条件是：

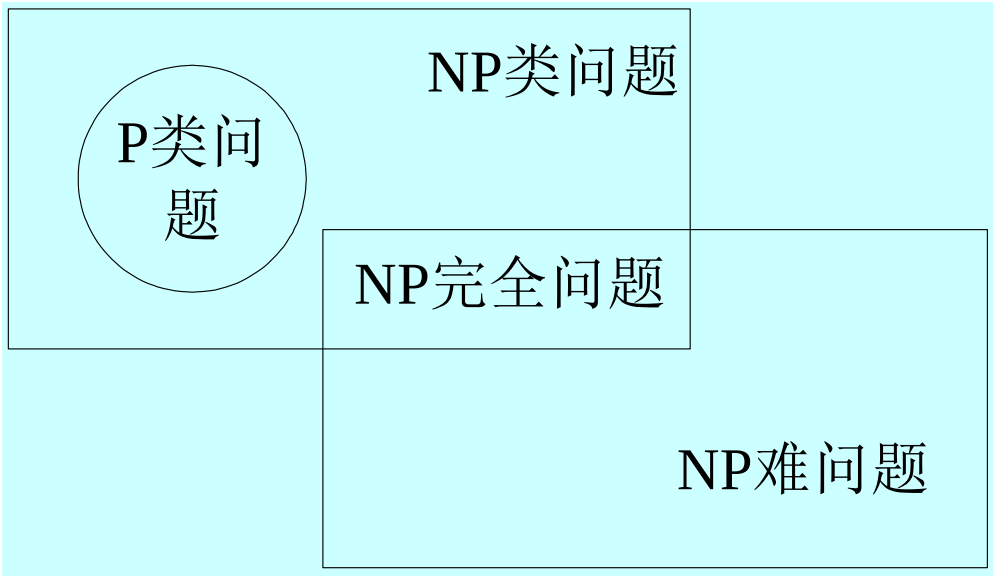
（1）*D*属于NP类；

（2）NP中的任何问题都能够在多项式时间内转化为*D*。



NP理论

- **NP完全问题（NP Complete Problem）**
- 满足条件（2）但不满足条件（1）的问题被称为NP难问题。也就是说，NP难问题不一定是NP类问题，例如图灵停机问题。正式地说，一个NP难问题至少跟NP完全问题一样难，也许更难！例如在某些任意大的棋盘游戏走出必胜的下法，就是一个NP难的问题，这个问题甚至比那些NP完全问题还难。



可求解与难求解问题

P类问题，NP类问题

■**P类问题**： 多项式问题(**P**olynomial Problem)，指计算机可以在有限时间内求解的问题，即：P类问题是可以找出一个呈现$O(n^a)$复杂性算法的问题，其中a为常数。

- 如果一个问题可以找到一个能在多项式的时间里解决它的算法，那么这个问题就属于P问题

■**NP类问题**：非确定性多项式问题(**N**on-deterministic **P**olynomial)。在多项式时间内难于求解但不难判断给定一个解的正确性的问题，即：

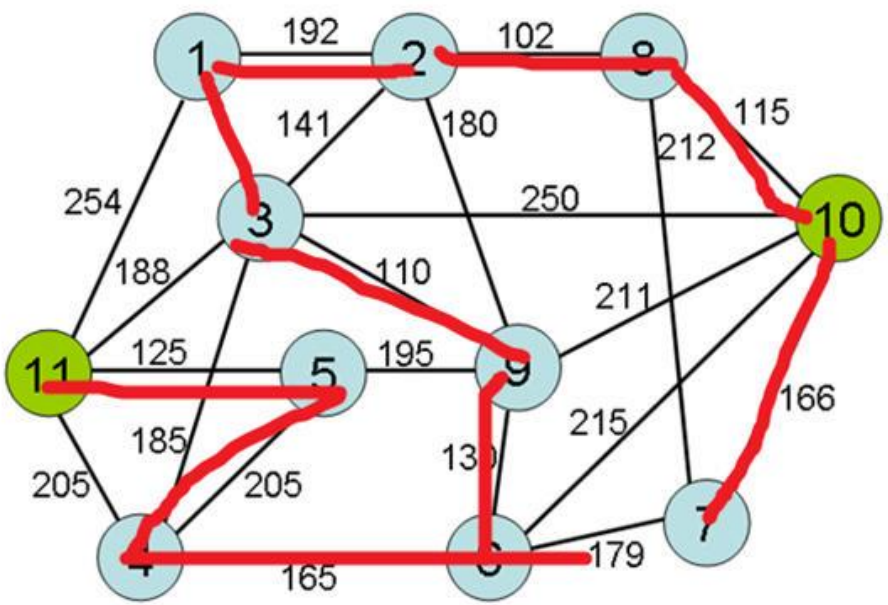
- 在多项式时间内可以由一个算法验证一个解是否正确的问题。

可求解与难求解问题

P类问题，NP类问题举例

■**P类问题**： 是否可以在多项式时间里解决

■**NP类问题**：是否可以在多项式时间内可以验证一个解是否正确



思考：**P = NP** 是否正确？

- 引申概念：

NPC: NP完全问题

NP hard: NP难问题

- 问题：如何有效求解NP问题？
  - 近似算法：进化计算、群体智能等

目录：

- 问题：您了解达尔文的进化论吗？
  - 物竞天择，适者生存。
  - 这是一个优化的过程。

进化算法

- **Charles Darwin**的自然选择学说是生物进化的基础。
  - 同期，**Alfred Wallace**独立地发展了一种相似的学说。

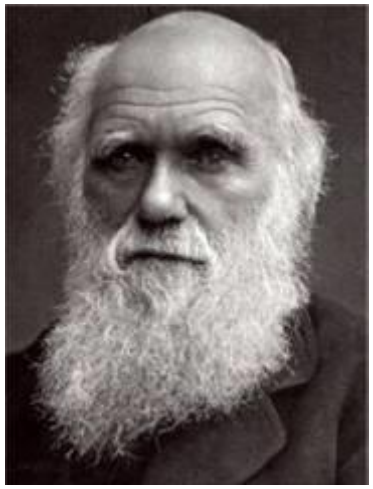
Darwin的进化论可以描述为：

–在一个资源有限的、种群稳定的世界中，每个生物个体都会与其他生物个体为了生存而竞争。拥有优良性状的个体会更加容易获得生存和繁殖的机会，它的性状也更易于传给后代。这些优良性状被下一代继承，经过一段时间便成为种群中的主要性状。

–在幼年生物体的发育过程中，随机事件会导致幼年生物体性状的随机改变。如果新出现的性状有益于生物体，则该生物体获得生存的概率会有所提高。

进化计算（Evolutional Computation）

- 生物种群的生存过程普遍遵循达尔文的物竞天择、适者生存的进化准则。种群中的个体根据对环境的适应能力而被大自然所选择或淘汰。进化过程结果反映在个体结构上，其染色体包含若干基因，体现了个体的外部特性与内部机理间的逻辑关系。生物通过个体间的选择、交叉、变异来适应大自然环境。



- 20世纪60年代以来，如何模仿生物来建立功能强大的算法，进而将它们运用于复杂的优化问题，越来越成为一个研究热点。进化计算正是在这一背景下蕴育而生的。进化计算包括遗传算法 、进化策略 、进化编程和遗传编程，本讲中着重讨论遗传算法。

遗传算法（Genetic Algorithm）

- 遗传算法是模仿生物遗传学（孟德尔，摩尔根）和自然选择机理（达尔文），通过人工方式所构造的一类优化搜索算法，是对生物进化过程进行的一种数学仿真，是进化计算的最重要的形式。
- 遗传算法为那些难以找到传统数学模型的难题指出了一个解决方法。
- 进化计算和遗传算法借鉴了生物科学中的某些知识，这也体现了人工智能这一交叉学科的特点。

- 遗传算法是在生物进化的启示下得到的一种搜索和自适应算法(以字符串表示状态空间)。
- 遗传算法模拟了生物的繁殖、交配和变异现象，从任意一初始种群出发，产生一群新的更适应环境的后代。这样一代一代不断繁殖、进化，最后收敛到一个最适应环境的个体上。
- 遗传算法对于复杂的优化问题无需建模和进行复杂运算，只需要利用遗传算法的算子就能寻找到问题的最优解或满意解。

遗传算法

- **1.遗传算法基本概念**
- **2.遗传操作**
- **3.基本遗传算法**
- **4.遗传算法应用举例**
- **5.高级专题**
  - 一般进化算法
  - 染色体的表示
  - 初始种群
  - 适应度函数
  - 选择
  - 繁殖算子
  - 终止条件
  - 进化计算与经典优化算法

1.遗传算法基本概念

- 个体与种群
  - ①个体
    - 就是模拟生物个体而对问题中的对象（一般就是问题的解）的一种称呼，一个‘个体’也就是搜索空间中的一个点。
  - ②种群(population)
    - 就是模拟生物种群而由若干‘个体’组成的群体，它一般是整个搜索空间的一个很小的子集。

- 适应度与适应度函数
  - ①适应度(fitness)
    - 就是借鉴生物个体对环境的适应程度,而对问题中的个体对象所设计的表征其优劣的一种测度。
  - ②适应度函数(fitness function)
    - 就是问题中的全体个体与其适应度之间的一个对应关系。它一般是一个实值函数。该函数就是遗传算法中指导搜索的评价函数。

- 染色体与基因
  - 染色体（chromosome）就是问题中个体的某种字符串形式的编码表示。字符串中的字符也就称为基因（gene）。
  - 例如：染色体的二进制编码形式

个体		染色体
9	----	1001
（2，5，6）	----	010 101 110

- **遗传算法与生物自然进化的比较**

自然进化	遗传算法
染色体	字符串
基因	字符，特征
等位基因(allele)	特征值
染色体位置(locus)	字符串位置
基因型(genotype)	结构
表型(phenotype)	参数集，译码结构

本章内容

- 1.遗传算法基本概念
- 2.遗传操作
- 3.基本遗传算法
- 4.遗传算法应用举例
- 5.高级专题
  - 一般进化算法
  - 染色体的表示
  - 初始种群
  - 适应度函数
  - 选择
  - 繁殖算子
  - 终止条件
  - 进化计算与经典优化算法

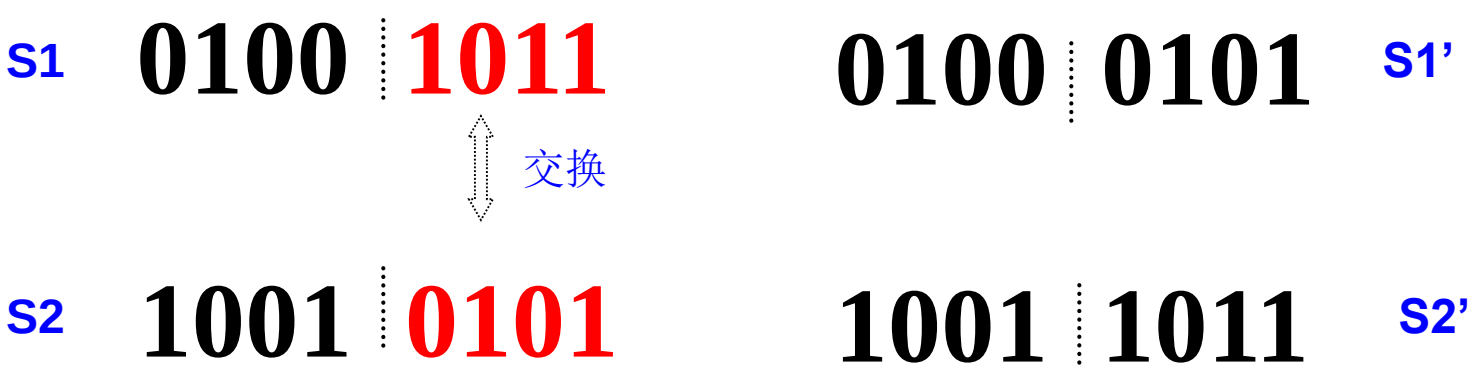
2.遗传操作

- 遗传操作亦称遗传算子(genetic operator)，就是关于染色体的运算。遗传算法中有三种遗传操作:
  - 选择-复制(selection-reproduction)
  - 交叉(crossover，亦称交换、交配或杂交)
  - 变异(mutation，亦称突变)

- 选择-复制
  - 通常做法是：对于一个规模为 N 的种群 S，按每个染色体 $x_i \in S$的选择概率 $P(x_i)$ 所决定的选中机会， 分N次从S中随机选定N个染色体， 并进行复制。
  - 这里的选择概率$P(x_i)$的计算公式为：

$$P(x_i) = \frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)}$$

- 交叉
  - 就是互换两个染色体某些位上的基因。
  - 例如两个染色体S1=01001011，S2=10010101，从第四位开始交换4个基因，如下图所示，得到2个新的染色体S1’=01000101和S2’=10011011，作为S1和S2的子代染色体进入下一代种群。



- 【思考问题】
  - 1)交换前半部分是否可行？
  - 2)选择中间几位交换是否可行？
  - 3)交换位置能不能随机选择？

- 变异
  - 就是改变一条染色体上某个(些)位上的基因。 例如：对二进制数表示的染色体就是将选择基因从0变为1，或从1变为0，其它未被选择的基因不变。例如：S=11001101，选择第三位进行变异，得到下一代染色体S’=11101101，如下图所示：

11001101 ⇨ 11101101

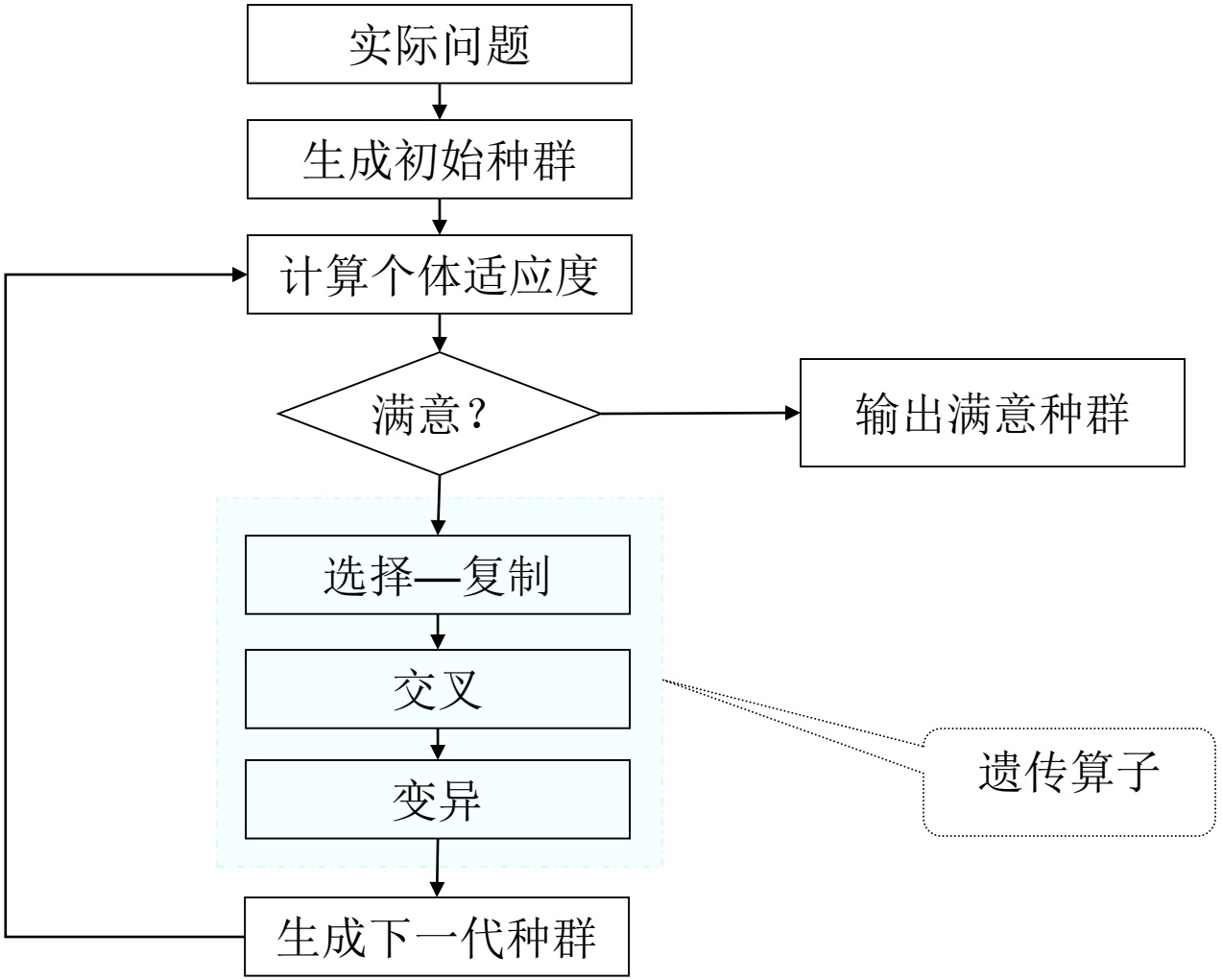
  - 【思考问题】
    - 能否随机选择变异的基因？
    - 能否选择几个基因进行变异？

本章内容

- 1.遗传算法基本概念
- 2.遗传操作
- 3.基本遗传算法
- 4.遗传算法应用举例
- 5.高级专题
  - 一般进化算法
  - 染色体的表示
  - 初始种群
  - 适应度函数
  - 选择
  - 繁殖算子
  - 终止条件
  - 进化计算与经典优化算法

3.基本遗传算法

3.1 基本遗传算法框图



- **3.2算法中的控制参数**

  1.种群规模—个体（染色体）的个数

  2.最大换代数

  3.交叉率(crossover rate)

  就是参加交叉运算的染色体个数占全体染色体总数的比例，记为**Pc**，取值范围一般为**0.4～0.99**。

  4.变异率(mutation rate)

  是指发生变异的基因位数所占全体染色体的基因总位数的比例，记为**Pm**，取值范围一般为**0.0001～0.1**。

- **3.3基本遗传算法步骤**
  - 步1：在搜索空间U上定义一个适应度函数f(x)，给定种群规模N，交叉率Pc和变异率Pm，代数T；
  - 步2：随机产生U中的N个个体s1, s2, …, sN，组成初始种群G={s1, s2, …, sN}，置代数计数器t=1；
  - 步3：计算G中每个个体的适应度f(x)；
  - 步4：若终止条件满足，则取G中适应度最大的个体作为所求结果，算法结束。

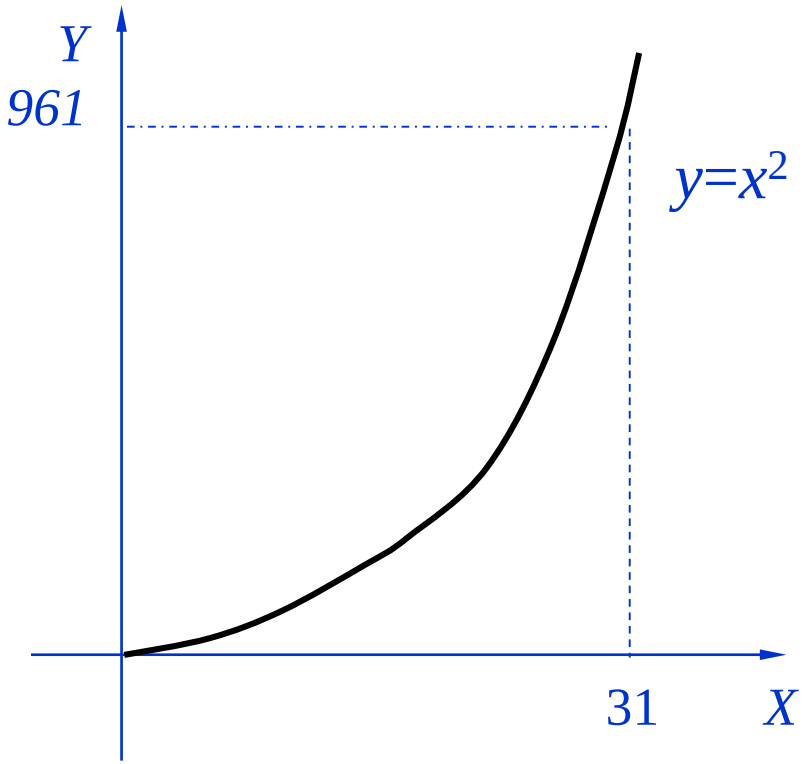
- 步5：按选择概率P(xi)所决定的选中机会，每次从G中随机选定1个个体并将其染色体复制，共做N次，然后将复制所得的N个染色体组成群体Gt1；
- 步6：按交叉率Pc所决定的参加交叉的染色体数c，从Gt1中随机确定c个染色体，配对进行交叉操作，并用产生的新染色体代替原染色体，得群体Gt2；
- 步7：按变异率Pm所决定的变异次数m，从Gt2中随机确定m个染色体，分别进行变异操作，并用产生的新染色体代替原染色体，得群体Gt3；
- 步8：将群体Gt3作为新一代种群，即用Gt3代替G，t = t+1，转步3；

本章内容

- 1.遗传算法基本概念
- 2.遗传操作
- 3.基本遗传算法
- 4.遗传算法应用举例
- 5.高级专题
  - 一般进化算法
  - 染色体的表示
  - 初始种群
  - 适应度函数
  - 选择
  - 繁殖算子
  - 终止条件
  - 进化计算与经典优化算法

4.基本遗传算法应用举例

- **【例1】利用遗传算法求解区间［0,31］上的二次函数$y=x^2$的最大值。**



- 问题分析：
  - 原问题可转化为在区间［0, 31］中搜索能使y取最大值的点x的问题。那么，［0, 31］ 中的点x就是个体, 函数值f(x)恰好就可以作为x的适应度，区间［0, 31］就是一个(解)空间 。这样, 只要能给出个体x的适当染色体编码, 该问题就可以用遗传算法来解决。

- 【解】
  - （1） 设定种群规模，编码染色体，产生初始种群。

    将种群规模设定为4；用5位二进制数编码染色体；取下列个体组成初始种群G:

    s1= 13 (01101)，　s2= 24 (11000)

    s3= 8　(01000)，　s4= 19 (10011)
  - （2） 定义适应度函数,

    取适应度函数：$f(x)=x^2$
  - （3） 计算各代种群中的各个体的适应度, 并对其染色体进行遗传操作,直到适应度最高的个体(即31（11111）)出现为止。

- 首先计算种群G中各个体

s1= 13(01101),　s2= 24(11000)

s3= 8 (01000),　s4= 19(10011)

的适应度f (si) 。容易求得：

$$f(s1) = f(13) = 13^2 = 169$$

$$f(s2) = f(24) = 24^2 = 576$$

$$f(s3) = f(8) = 8^2 = 64$$

$$f(s4) = f(19) = 19^2 = 361$$

- 再计算种群G中各个体的选择概率。根据选择概率计算公式：

$$P(x_i) = \frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)}$$

- 可以求得：

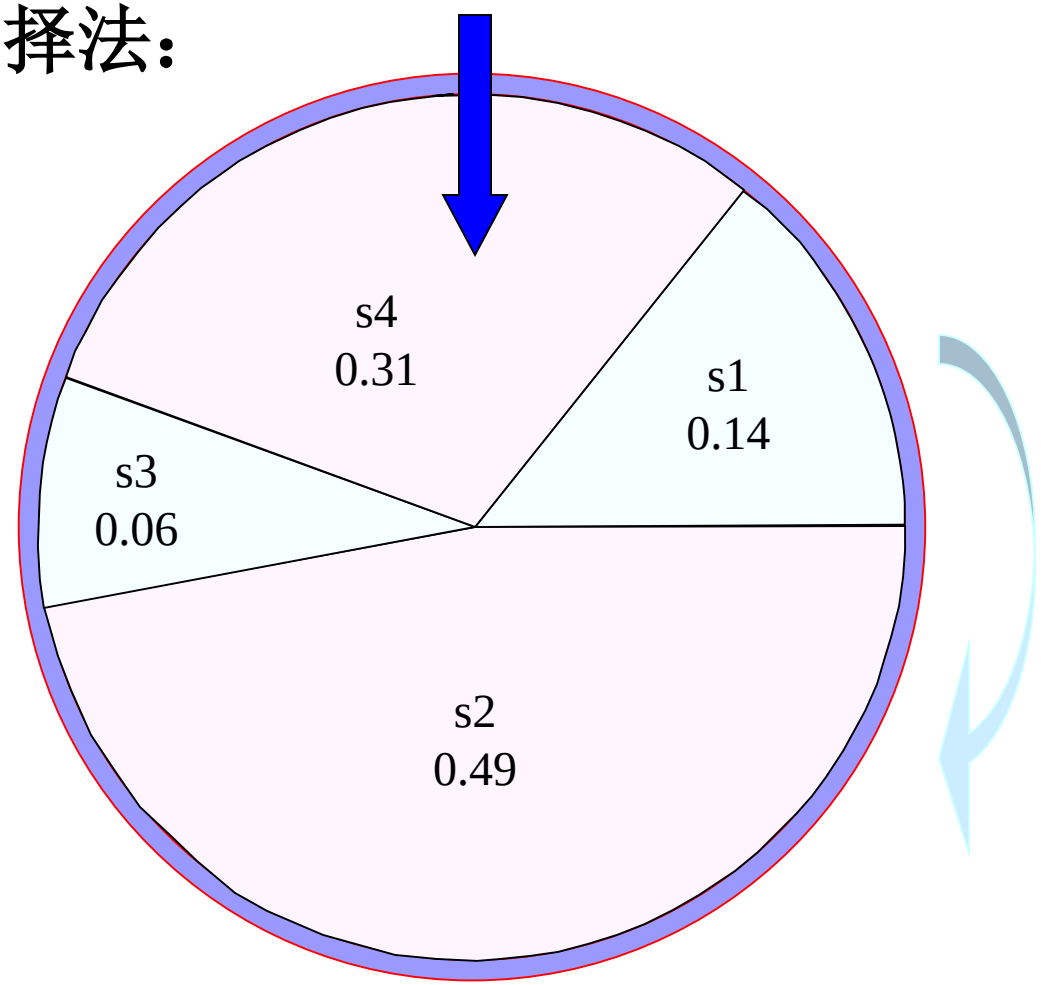
P(s1) = P(13) = 0.14

P(s2) = P(24) = 0.49

P(s3) = P(8) = 0.06

P(s4) = P(19) = 0.31

- 轮盘赌选择法：



轮盘赌选择法示意

- 轮盘赌选择法的子过程模拟:
  - ① 在［0, 1］区间内产生一个均匀分布的随机数r。
  - ② 若 $r \le q_1$，则染色体x_1被选中。
  - ③ 若 $q_{k-1} < r \le q_k$ $(2 \le k \le N)$，则染色体x_k被选中。其中的q_i称为染色体x_i $(i=1, 2, \ldots, n)$的积累概率, 其计算公式为:

$$q_i = \sum_{j=1}^{i} P(x_j)$$

【模拟遗传算法执行过程】

- 第一轮遗传操作：
- 选择-复制：
  - 设从区间［0, 1］中产生4个随机数如下：

$$r1 = 0.450126, \quad r2 = 0.110347$$

$$r3 = 0.572496, \quad r4 = 0.98503$$

- 轮盘赌法选择的结果

染色体	适应度	选择概率	累积概率	选中次数
s1=01101	169	0.14	0.14	1
s2=11000	576	0.49	0.63	2
s3=01000	64	0.06	0.69	0
s4=10011	361	0.31	1.00	1

- 于是，经复制得群体：
  - s1’=11000（24），s2’=01101（13）
  - s3’=11000（24），s4’=10011（19）

- 交叉：
  - 设交叉率pc=100%，即S1中的全体染色体都参加交叉运算。
  - 设s1’与s2’配对，s3’与s4’配对。分别交换后两位基因，得新染色体：

    s1’’=11001（25），s2’’=01100（12）

    s3’’=11011（27），s4’’=10000（16）

- 变异：
  - 设变异率pm=0.001。
  - 这样，群体Gt1中共有

    5×4×0.001=0.02位基因可以变异。
  - 0.02位显然不足1位，所以本轮遗传操作不做变异。
  - 于是，得到第二代种群的Gt3，即G2：

    s1=11001（25）, s2=01100（12）

    s3=11011（27）, s4=10000（16）

- 第二代种群的G2

染色体	适应度	选择概率	累积概率	选中次数
s1=11001	625	0.36	0.36	1
s2=01100	144	0.08	0.44	0
s3=11011	729	0.41	0.85	2
s4=10000	256	0.15	1.00	1

- 第二轮遗传操作：
  - 假设这一轮选择-复制操作中，种群G2中的4个染色体都被选中，则得到群体：

    s1’=11001（25），s2’= 01100（12）

    s3’=11011（27），s4’= 10000（16）
  - 做交叉运算，让s1’与s2’，s3’与s4’ 分别交换后三位基因，得：

    s1’’ =11100（28）, s2’’ = 01001（9）

    s3’’ =11000（24）, s4’’ = 10011（19）
  - 这一轮仍然不会发生变异。
  - 于是，得第三代种群G3：

    s1=11100（28）, s2=01001（9）

    s3=11000（24）, s4=10011（19）

- 第三代种群G 3

染色体	适应度	选择概率	累积概率	选中次数
s1=11100	784	0.44	0.44	2
s2=01001	81	0.04	0.48	0
s3=11000	576	0.32	0.80	1
s4=10011	361	0.20	1.00	1

- 第三轮遗传操作：
  - 设这一轮的选择-复制结果为：

    s1’=11100（28），s2’=11100（28）

    s3’=11000（24），s4’=10011（19）
  - 做交叉运算，让s1’与s4’，s2’与s3’ 分别交换后两位基因，得：

    s1’’=11111（31）, s2’’=11100（28）

    s3’’=11000（24）, s4’’=10000（16）
  - 这一轮仍然不会发生变异。
  - 于是，得第四代种群G4：

    s1=11111（31），s2=11100（28）

    s3=11000（24），s4=10000（16）

- 显然，在这一代种群中已经出现了适应度最高的染色体s1=11111。于是，遗传操作终止，将染色体“11111”作为最终结果输出。
- 然后，将染色体“11111”解码，即得所求的最优解：31。
- 将31代入函数$y=x^2$中，即得原问题的解，即函数$y=x^2$的最大值为961。

- 同学们可能疑问上面例子太简单了，用GA求解是简单问题复杂化！下面我们再来看一个例子：
- 【例2】已知函数

$$f(x) = x \cdot \sin(10\pi \cdot x) + 1.0$$

当x∈[-1,2]时，求函数的最大值。要求求解结果精确到6位小数。

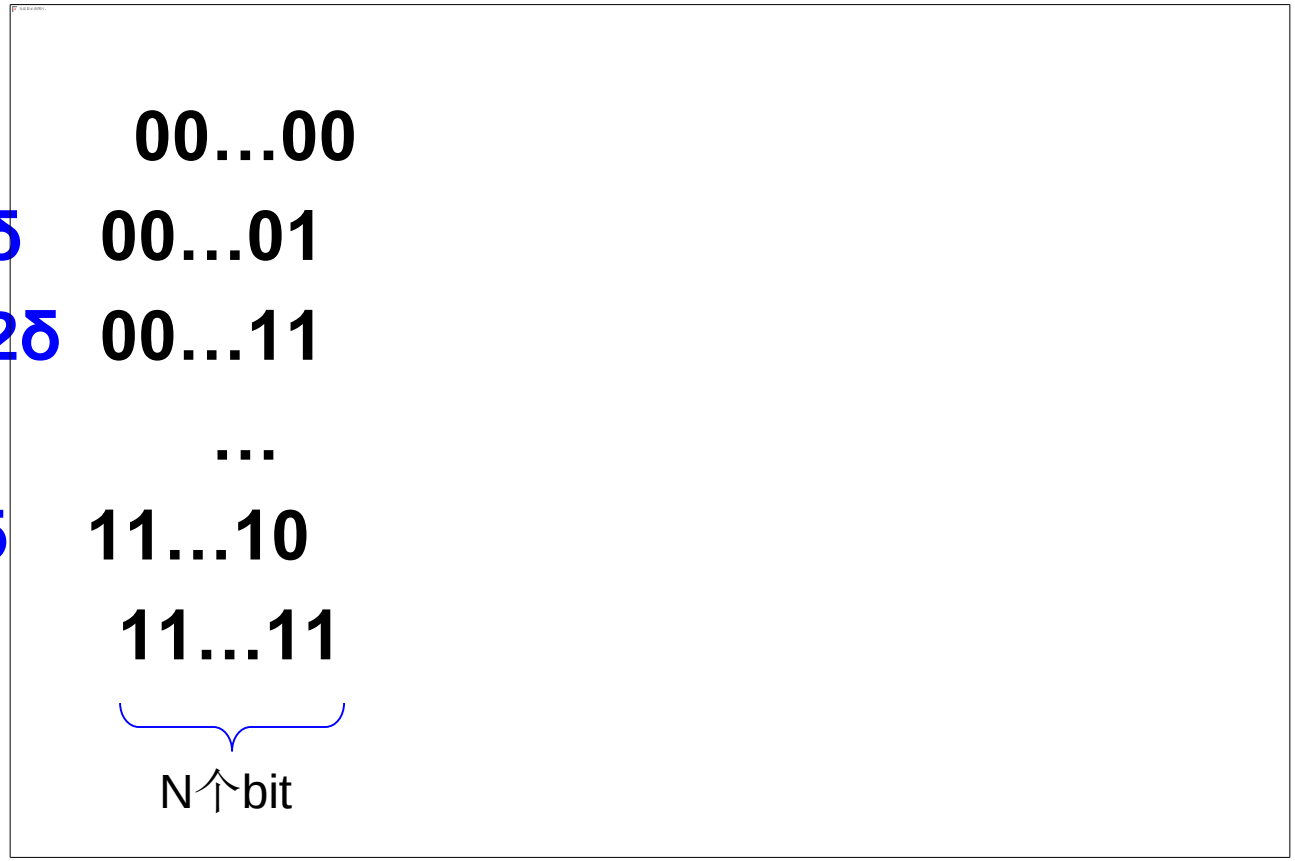
- 高数解法
  - **f’(x)=0**，得：

    sin(10πx)+10πxcos(10πx)=0
  - 变换得：

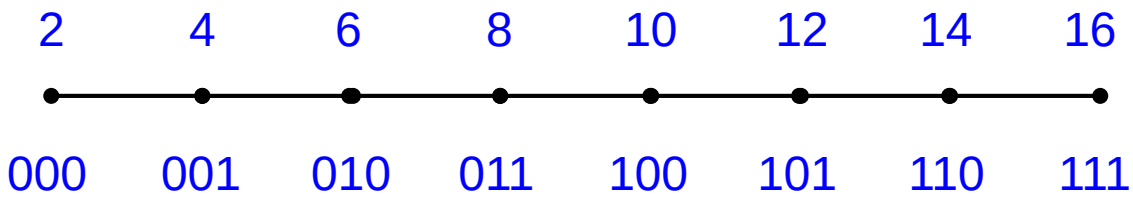
    tan(10πx)=-10πx
  - 可见问题有很多解。
- 更复杂的函数，甚至没法通过解析方法给出问题的解。

- 连续空间的二进制编码
  - 空间范围：[a, b]，其中a<b，
  - 二进制编码长度N，可以表示2^N个二进制数。将这2^N个二进制数对应到[a, b]上的2^N个点上，相当于将区间[a, b]等分为2^N-1个段，每段的步长为：

$$\delta = \frac{b-a}{2^N - 1}$$


a 00…00
a+δ 00…01
a+2δ 00…11
…
b-δ 11…10
b 11…11
N个bit

- 例：[2, 16]，N=3
  - 步长$\delta=(16-2)/(2^3-1)=2$
  - 二进制编码与[2, 16]区间点的对应关系如下：



- 连续空间的二进制解码
  - 连续区间[a, b]，码长N，给定一个二进制码，对应[a, b]区间的十进制数为：

    $$x = a + (\text{二进制码转为十进制数}) \times \delta$$

  - 例前例中，给定二进制码：101，在区间[2, 16]上对应的十进制数为：

    $$x = 2 + 5 \times 2 = 12$$

  - 由前面的图示也可以看出此结果。

- “例2”的二进制编码
  - 因为要精确到小数点后6位，所以步长应为：

    $\delta \leq 0.000001$
  - 根据步长的计算公式，计算二进制编码长度N：

$$N = \log_2\left(\frac{b-a}{\delta}+1\right)$$

  - 结果为 N=22，即二进制码长为22位。
  - 实际步长$\delta=(2-(-1))/(2^{22}-1)=7.152559\times10^{-7}$

- “例2”的二进制解码
  - 给定一个22位二进制编码：
  - 10,0010,1110,1101,0100,0111
  - 对应十进制数为：0.637197
- “例2”的种群规模：50
- 交叉率：Pc=0.25
- 变异率：Pm=0.01
- 经150代更新，结果为：
  - S_{max}=11,1100,1101,0001,0000,0101
  - X_{max}=1.850773
  - $f(X_{max})$=2.850227

- 【例3】遗传算法解不定方程：a+2b+3c+4d=30，其中a、b、c、d皆为正整数。
  - 【分析】由题目可见a、b、c、d取值的范围为：

    1<= a、b、c、d<=30
  - 染色体（个体）编码：十进制编码

    使用a、b、c、d的十进制取值组合，例：

    (14, 9, 2, 8)
  - 设计适应度函数：

$$f(x)=\frac{1}{|a+2b+3c+4d-30|}$$

- 种群规模5
- 随机产生初始种群

染色体编号	染色体（个体） (a,b,c,d)
s1	(1,28,15,3)
s2	(14,9,2,4)
s3	(13,5,7,3)
s4	(23,8,16,19)
s5	(9,13,5,2)

- **TSP问题的GA求解**
  - 已知n个城市的距离，推销员反问每个城市一次且仅一次，回到出发的城市，旅行距离最短。
  - 城市可用编号1—n表示。
  - 如何用GA求解呢？

- 编码：字符编码
  - 比如10个城市，1，3，5，7，9，2，4，8，10，6为一种走法，即一个染色体（个体）。
- 适应度函数
  - 路径距离的倒数，即：

$$f(k)=\frac{1}{\sum d_{ij}}$$

  - 其中，k为第k个染色体，d_{ij}为城市i和j之间的距离。

- 初始种群的产生
  - 在1-n之间的一种组合即是一个个体；
  - 可用不重复随机数产生。
- 选择—复制操作
  - 轮盘赌法

- 交叉
  - 多种方法
  - 也可沿用前面的方法，但要处理不合理的城市（重复、缺少）。
  - 最简单的方法：选择单个父体，除了出发城市，其余部分选择一个点进行循环，得到一个新的染色体。
  - 例：对路径1，3，5，7，9，2，4，8，10，6，选择后面4个城市循环，则新的路径变为：

    1，3，5，7，9，2，4，8，10，6

    1，4，8，10，6，3，5，7，9，2

- 变异
  - 假定对路径的第i个位置的城市m进行变异，在1-n之间产生一个随机数k，将m放到路径的最后，将k替换原来的m，这样k城市是重复的，删除那个重复的k城市。
  - 例： 对1，3，5，7，9，2，4，8，10，6
  - 假定在第六个位置变异，这里城市为2，
  - 假定产生的随机数为5，则序列变为：

    1，3，5，7，9，5，4，8，10，6，2
  - 删除蓝色的重复的5即可，变异后的染色体为：

    1，3，7，9，5，4，8，10，6，2

本章内容

- 1.遗传算法基本概念
- 2.遗传操作
- 3.基本遗传算法
- 4.遗传算法应用举例
- 5.高级专题
  - 一般进化算法
  - 染色体的表示
  - 初始种群
  - 适应度函数
  - 选择
  - 繁殖算子
  - 终止条件
  - 进化计算与经典优化算法

一般进化算法

- 对随机选择的一些个体执行基于自然选择的进化过程，可以看成是在染色体值空间的一个搜索过程。
  - 这些个体组成了“种群（population）”。
  - 进化算法就是对给定的问题求最优解的随机搜索方法。

- 进化搜索主要受到以下几个部分的影响。
  - 编码（encoding）：与染色体一样，对问题的候选解进行编码。
  - 适应度函数（a fitness function）：用于求适应度的函数，表征个体的生存能力。
  - 初始化（initialization）：种群的初始化。
  - 选择（selection）：选择算子。
  - 繁殖（reproduction）：繁殖算子。

一般进化算法

1. 令代数计数器t=0；
2. 创建和初始化n_x维的群体C(0)，包含n_s个个体；
3. **while** 终止条件不为真 **do**
4. 评估每个个体$x_i(t)$的适应度值$f(x_i(t))$；
5. 执行繁殖算子，来产生后代；
6. 选择新群体C(t+1)；
7. 进入下一代，即t=t+1；
8. **end**

进化算法中，每一次迭代，都被称做一代（Generation）。

一般进化算法

- **Darwin理论被抽象成如下的算法：**
  - 自然选择发生在繁殖算子中，“最好的”父代有更多的机会被选择来产生后代，形成新的种群。
  - 随机改变通过变异（**Mutation**）算子来实现。

各种进化算法

- 进化算法各部分的不同实现，形成了不同的进化计算方法。

- **遗传算法（Genetic Algorithms: GAs）**
  - 以基因进化为模型。
- **遗传编程（Genetic Programming: GP）**
  - 以遗传算法为模型，但个体为程序（表示为树）。
- **进化规划（Evolutionary Programming: EP）**
  - 来源于对进化中自适应行为的模拟（如表现型进化，**phenotypic evolution**）。

各种进化算法

- 进化策略（Evolution Strategies: ESs）
  - 对进化过程中的控制变量进行建模，即进化的进化。
- 差分进化（Differential Evolution: DE）
  - 类似于遗传算法，不同之处在于其所使用的繁殖机制。
- 文化算法（Cultural Evolution: CE）
  - 对种群文化的进化、以及文化如何影响个体基因和表现型的进化的进行建模。
- 协同进化（Co-Evolution: CoE）
  - 模拟初始“愚蠢的”个体如何通过合作或竞争来获取必要性状以求得生成的演化过程。

本章内容

- 1.遗传算法基本概念
- 2.遗传操作
- 3.基本遗传算法
- 4.遗传算法应用举例
- 5.高级专题
  - 一般进化算法
  - 染色体的表示
  - 初始种群
  - 适应度函数
  - 选择
  - 繁殖算子
  - 终止条件
  - 进化计算与经典优化算法

染色体的表示

- 一些进化算法（包括遗传算法）使用实数表示方式。
- 尽管进化规划最早是采用有限状态机表示，但如今主要用实值表示，即每个向量元素为实数。
- 进化策略和差分进化也使用实数表示方式。
- **Michalewicz**认为，原始的浮点表示方式能带来更高的精度、更快的求解速度，性能超越了等效的二进制表示方式。
- 其他表示方案：整数表示、排列表示、有限状态机表示、树表示、混合整数表示，等等。

本章内容

- 1.遗传算法基本概念
- 2.遗传操作
- 3.基本遗传算法
- 4.遗传算法应用举例
- 5.高级专题
  - 一般进化算法
  - 染色体的表示
  - 初始种群
  - 适应度函数
  - 选择
  - 繁殖算子
  - 终止条件
  - 进化计算与经典优化算法

初始种群

- 进化算法是一种基于种群的随机搜索算法。
  - 每个进化算法都维护一个由候选解组成的种群。
  - 应用进化算法解决优化问题的第一步是产生一个初始种群。

- 产生初始种群的标准方法是在可行域中产生随机值，并分配给每个染色体的每个基因。
  - 随机选择的目标是确保初始种群为整个搜索空间的均匀表示。
  - 如果初始种群没有覆盖搜索空间，则有可能使得未覆盖区域被搜索过程所忽略。

初始种群

- 初始种群的大小会影响计算复杂性和空间探索能力。

- 大量的个体能增加多样性，进而提高种群的空间搜索能力。
  - 然而，个体越多，每代的计算量越大。
  - 当每代的计算时间增加时，可能只需要较少的代数就能找到可以接受的解；

- 如果种群的个体数量较小，则只能探索空间中的较小部分。
  - 虽然每代的计算时间减少了，但进化算法需要更多的代数才能收敛得到一个与大种群相当的结果。
  - 在使用小个体数量种群时，如果增加变异频率，则可迫使进化算法探索更多的搜索空间。

本章内容

- 1.遗传算法基本概念
- 2.遗传操作
- 3.基本遗传算法
- 4.遗传算法应用举例
- 5.高级专题
  - 一般进化算法
  - 染色体的表示
  - 初始种群
  - 适应度函数
  - 选择
  - 繁殖算子
  - 终止条件
  - 进化计算与经典优化算法

适应度函数

- 在达尔文式的进化模型里，拥有最好性状的个体获得生存和繁殖的机会更大。

- 在进化算法中，为确定一个个体的生存能力，通常用一个数学函数来表征染色体所表示的解有多好。

- 所谓适应度函数，将把一个染色体映射成一个标量值：

$$f:\ \Gamma^{n_x} \rightarrow \mathrm{R}$$

式中，Γ代表n_x维染色体的数据类型。

适应度函数f可以用目标函数Ψ来表示，因为目标函数描述了最优化问题。

适应度函数

- 目标函数所期待的输入数据表示形式不一定与染色体的表示形式一致。
  - 如果不一致，适应度函数可以为：

$$f:\ S_C \xrightarrow{\Phi} S_X \xrightarrow{\Psi} \mathrm{R} \xrightarrow{\gamma} \mathrm{R}_+$$

式中，S_C为目标函数的搜索空间，Φ、Ψ、γ分别代表染色体编码函数、目标函数、和缩放函数。

缩放函数是可选的，用于在比例选择中以保证适应度函数值是正数。

为方便起见，以下假定：

$$S_C = S_X,\ f = \Psi$$

适应度函数

- 通常，适应度是适应程度的绝对度量。
  - 用染色体表示的解可以直接用目标函数评估。

- 对于某些应用，如博弈学习，是不可能找到绝对的适应度函数的。
  - 但是，可以找到一个相对适应度评估方法来确定一个个体相对于种群中其他个体的能力。

适应度函数

- 设计适应度函数时，要注意到优化问题的不同类型。
  - 无约束优化：适应度函数即为目标函数。
  - 带约束的优化：进化算法的适应度函数要包含两个目标，一个是原始的目标函数，另一个是约束惩罚函数。
  - 多目标优化：可以通过带权重的聚集方法来解决，相应的适应度函数为全部子目标的加权和。或者，采用基于Pareto的优化算法。
  - 动态和噪音问题：在此类问题中，解的函数值随着时间改变而改变。动态适应度函数依赖于时间，而噪音函数往往会添加高斯噪音成分。

本章内容

- 1.遗传算法基本概念
- 2.遗传操作
- 3.基本遗传算法
- 4.遗传算法应用举例
- 5.高级专题
  - 一般进化算法
  - 染色体的表示
  - 初始种群
  - 适应度函数
  - 选择
  - 繁殖算子
  - 终止条件
  - 进化计算与经典优化算法

选择算子

- 选择是进化算法中的主要算子之一，与达尔文式的适者生存的概念直接相关。

- 选择算子的主要目标是获得更好的解，通过进化算法中的两个步骤来完成：新种群的选择和繁殖。

- 新种群的选择：在每一代的结束，一个由候选解构成的新种群会被选择作为下一代种群。新种群可用仅从后代中选择，或是同时从后代与父代中选择。选择算子的目标是确保优良个体能存活到下一代。
- 繁殖：后代通过交叉或变异算子生成。
  - 在交叉中，优良个体应该有更多的机会去繁殖，以确保后代包含最好个体的基因。
  - 在变异中，选择机制关注“劣势”个体。目的是引入更好的特征（即“性状”），以增加劣势个体的生存能力。

选择压力

- 选择算子以它们的选择压力（**selective pressure**）为特征，选择压力又叫接管时间（**takeover time**），它与生成一个均匀分布的种群所需要的时间相关。

- 选择压力定义：选择算子单独反复作用使得最优解占整个种群的速度。
  - 高选择压力的算子会比低选择压力的算子能更快地降低种群的多样性，但同时会导致过早收敛到次优解。
  - 高选择压力会限制种群的探索能力。

选择算子

- 随机选择
- 比例选择
- 锦标赛选择
- 排序选择
- 玻尔兹曼选择
- (μ^+, λ)选择
- 精英选择
- 名人堂
- ……

本章内容

- 1.遗传算法基本概念
- 2.遗传操作
- 3.基本遗传算法
- 4.遗传算法应用举例
- 5.高级专题
  - 一般进化算法
  - 染色体的表示
  - 初始种群
  - 适应度函数
  - 选择
  - 繁殖算子
  - 终止条件
  - 进化计算与经典优化算法

繁殖算子

- 繁殖是从选择的父代中，应用交叉（crossover）和/或变异算子（mutation）生成子代的过程。

- 交叉是通过组合随机选择的两个或多个父代个体的基因物质生成新个体的过程。
  - 如果选择关注的是最具适应能力的个体，则选择压力可能会降低新种群的多样性而引起过早收敛。

繁殖算子

- 变异是随机改变染色体基因值的过程。
  - 主要目的是使种群引入新的基因物质，提高基因的多样性。
- 应用变异算子时，应该注意不要破坏高适应度个体的优良基因。
  - 为此，变异通常以较低的概率进行。
  - 或者，变异概率也可与个体的适应度成反比，即适应度越低的个体具有越高的变异概率。
- 为了提高前期迭代的探索能力，变异概率可以初始为较大值，并随着时间逐渐减少直到最后一代。

繁殖算子

- 繁殖可以使用替换机制，即当且仅当新产生的子代个体的适应度优于它的父代个体时，才进行替换。

- 由于交叉和变异算子与表达方式和EC框架有关，有很多不同的实现方法。

本章内容

- 1.遗传算法基本概念
- 2.遗传操作
- 3.基本遗传算法
- 4.遗传算法应用举例
- 5.高级专题
  - 一般进化算法
  - 染色体的表示
  - 初始种群
  - 适应度函数
  - 选择
  - 繁殖算子
  - 终止条件
  - 进化计算与经典优化算法

终止条件

- 在进化算法中，进化算子不断迭代执行，直到满足终止条件。

- 最简单的终止条件就是限制进化算法执行的代数或调用适应度函数的次数。
  - 这个限制不能太小，否则进化算法不会有充足的时间去探索未知空间。

- 确定种群是否收敛的标准也常用作终止条件。
  - 收敛可以严格定义为种群变得停滞不动的时候，即种群中没有基因或表现特征改变时。

终止条件

- 确定种群是否收敛的标准也常用作终止条件。

  - 当连续几代内都没有提高时终止。例如，监视最优个体的适应度，如果在一个给定大小的时间窗内没有显著更新，则进化算法可以终止。相反，如果此条件不满足，则使用其他机制以增加多样性来获得更大的探索空间，如增加变异概率和变异次数。

  - 当种群没有改变时终止。在连续几代内，基因信息的平均改变量太小，则进化算法可以终止。

终止条件

- 确定种群是否收敛的标准也常用作终止条件。

  - 当得到一个可接受的解时终止。若$f(x^*)$表示目标函数的最优值，则当最优个体x_i满足$f(x_i)\le|f(x^*)-\varepsilon|$时，表示一个可接受的解已找到。 ε为错误门限，如果过大，则找到的可接受解较差；如果过小，则进化算法很难终止（如果没有施加时间限制）。

  - 当目标函数斜率接近0时终止。例如，

$$f'(t)<\varepsilon,\quad f'(t)=\frac{f\left(\mathbf{x}_{best}(t)\right)-f\left(\mathbf{x}_{best}(t-1)\right)}{f\left(\mathbf{x}_{best}(t)\right)}$$

其中$\mathbf{x}_{best}(t)$表示第t代的最佳个体。

本章内容

- 1.遗传算法基本概念
- 2.遗传操作
- 3.基本遗传算法
- 4.遗传算法应用举例
- 5.高级专题
  - 一般进化算法
  - 染色体的表示
  - 初始种群
  - 适应度函数
  - 选择
  - 繁殖算子
  - 终止条件
  - 进化计算与经典优化算法

进化计算与经典优化算法

- 经典优化算法已经在线性、二次型、强凸型、单模及其它某些特定问题上取得了成功。

- 进化算法则对不连续、不可微、多模和带噪问题更为有效。

- 进化算法与经典优化算法（**classical optimization：CO**）的主要不同在于搜索过程及搜索过程中使用的信息。

进化计算与经典优化算法

- 搜索过程
  - 经典优化算法使用确定的规则从搜索空间中的一点移动到另外一点。进化计算则使用概率变换规则。
  - 进化计算还对搜索空间进行并行搜索，而经典优化算法则使用顺序搜索。
  - 进化算法从不同的初始点开始搜索，这样能并行搜索大量空间。经典优化算法只能从一个点，连续地向最优点调整和移动。
- 搜索过程中使用的信息
  - 经典优化算法使用搜索空间的偏导数信息（通常为一阶或二阶）来启发搜索路径。另一方面，进化计算不使用偏导数信息，它使用个体的适应度来指导搜索。

本章小结

- 一般进化算法的框架
  - 染色体的表示、适应度函数
  - 初始种群
  - 选择、繁殖算子
  - 终止条件
- 进化计算与经典优化算法

谢谢大家！！！