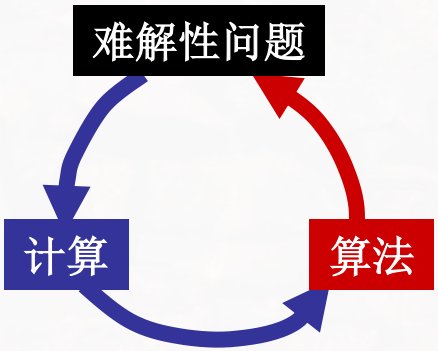
第9讲 怎样研究算法：遗传算法示例

什么是可求解与难求解问题?

难解性问题的基本求解思路是什么?

如何类比自然界生物求解问题的思想，设计计算类问题的求解算法?



可求解与难求解问题

----计算复杂性

----P类问题、NP类问题和NPC类问题

----NPC类问题的求解思路

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))

$O(n^3)$与$O(3^n)$的差别，$O(n!)$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!

$20! = 1.216 \times 10^{17}$

$20^3 = 8000$

$O(n^3)$	$O(3^n)$
0.2秒	4×10^{28}秒 =1015年

注：每秒百万次，n=60，1015年相当于10亿台计算机计算一百万年

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

P类问题，NP类问题，NPC类问题

■**P类问题**： 多项式问题(Polynomial 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。

■**NP类问题**：非确定性多项式问题(Non-deterministic Polynomial)。有些问题，其答案是无法直接计算得到的，只能通过间接的猜算或试算来得到结果，这就是非确定性问题(Non-deterministic)。虽然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非确定性问题，就是NP类问题。

■**NPC问题**：完全非确定性多项式问题(NP-Complete)。如果NP问题的所有可能答案都可以在多项式时间内进行正确与否的验算的话就叫做完全非确定性多项式问题，即NP-Complete问题。

问：加密算法应该设计成一个什么问题呢？

NPC类问题求解

TSP问题的遍历算法

穷举法或称**遍历法**：对解空间中的每一个可能解进行验证，直到所有的解都被验证是否正确，便能得到精确的结果---**精确解**

可能是O(n!)或O(a^n)

TSP问题的贪心算法

近似解求解算法---近似解

应该是O(n^a)

仿生学算法

Δ = | 近似解 - 精确解 |
满意解： Δ充分小时的近似解

进化算法，遗传算法，蚁群算法，蜂群算法，…

遗传算法的缘起
--生物学中的遗传与进化

生物学中的遗传和进化

关于生物遗传进化的基本观点

(i) 生物的所有遗传信息都包含在其染色体中，染色体决定了生物的性状；

(ii) 染色体是由基因及其有规律的排列所构成的，遗传和进化过程发生在染色体上；

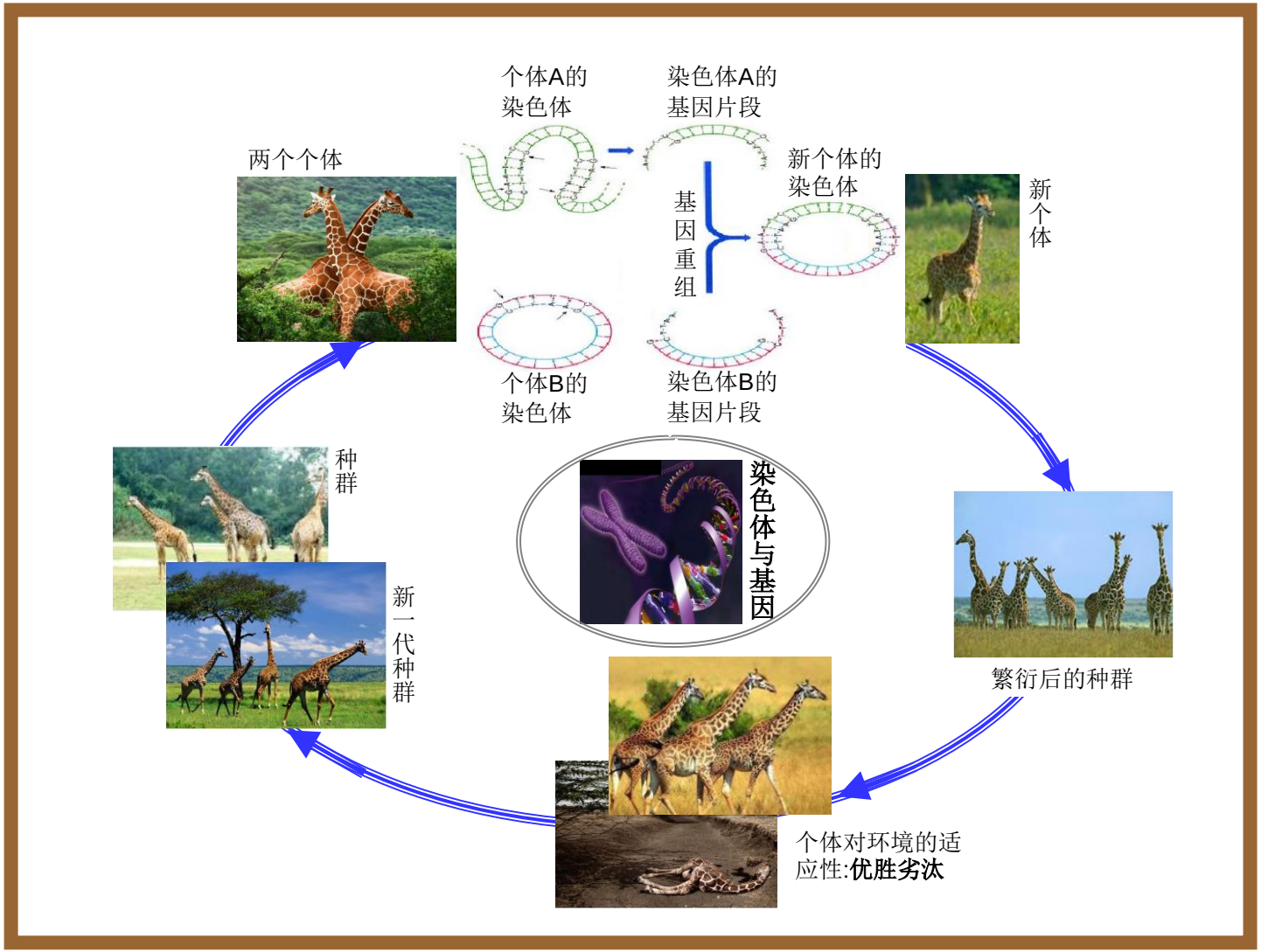
(iii) 生物的繁殖过程是由其基因的复制过程来完成的；

(iv) 通过同源染色体之间的交叉或染色体的变异会产生新的物种，使生物呈现新的性状。

(v) 对环境适应性好的基因或染色体经常比适应性差的基因或染色体有更多的机会遗传到下一代。

遗传与进化　　优胜劣汰

生物学中的遗传和进化


个体A的
染色体
染色体A的
基因片段
两个个体
新个体的
染色体
基因重组
新个体
个体B的
染色体
染色体B的
基因片段
染色体与基因
种群
新一代种群
繁衍后的种群
个体对环境的适应性:优胜劣汰

生物学中的遗传和进化

基本概念

- 种群(Population) vs. 个体(Individual) vs. 染色体(chromosome)
- 染色体(chromosome) vs.基因(gene)
- 基因型(Genotype) vs.表现型(Phenotype)
- 个体的适应度(Fitness)
- 选择(Selection)
- 复制((Reproduction)
- 交配/杂交(Crossover)
- 突变(Mutation)

计算学科的遗传算法

一个简单的遗传算法应用示例

求多项式函数的最小值：

$$\text{Min } F(X) = X^2 - 19X + 20, \text{ 其中} X=1,\ldots,64\text{之间的整数。}$$

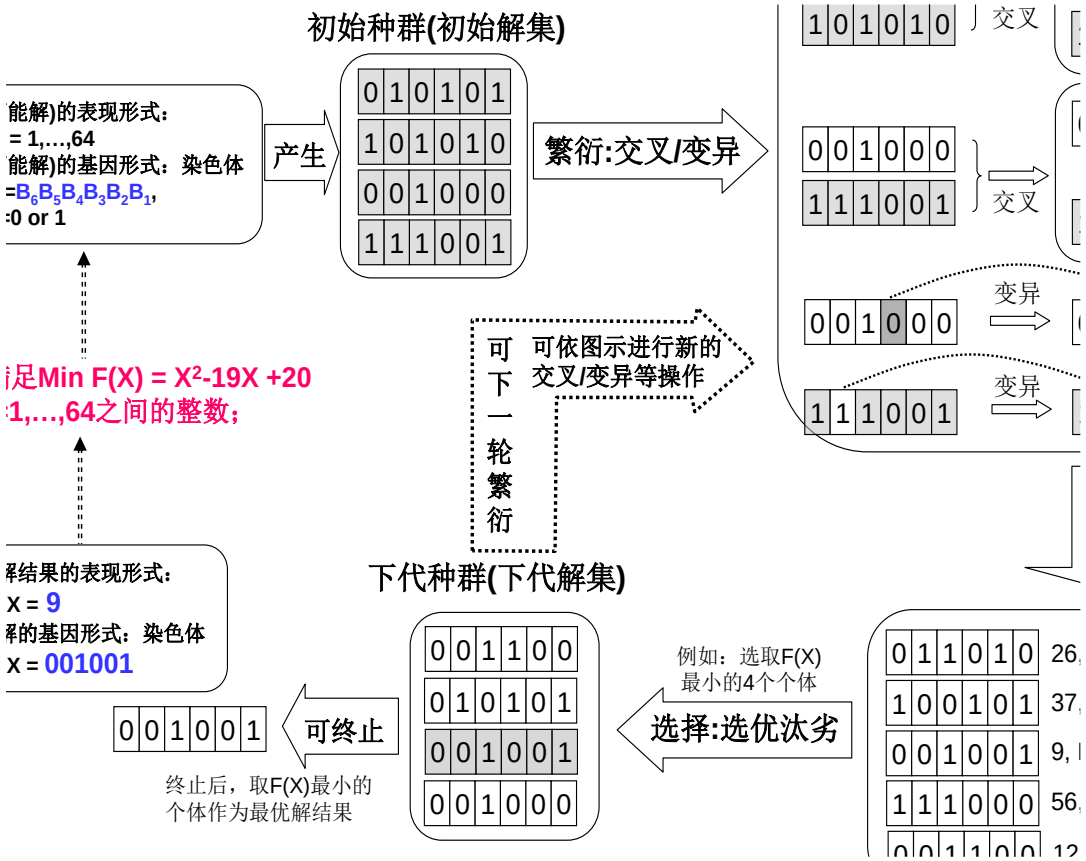
注：此题不难求出精确解，其精确解为X=9或者X=10。

如何用遗传算法进行求解?

生物学中的概念与计算学科中的概念映射

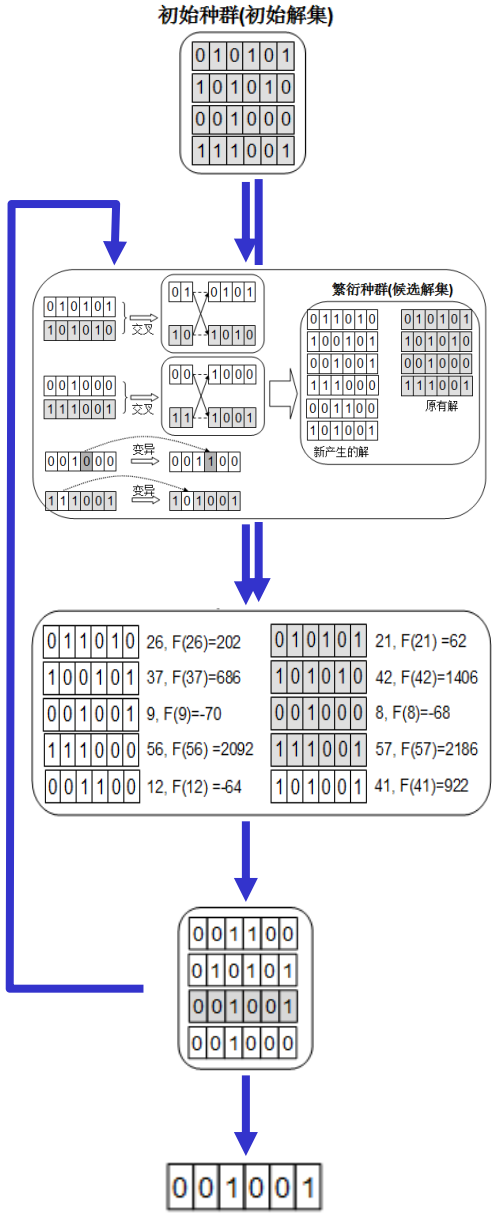
生物学中的概念	计算学科中的概念	解释
种群	**解集**/种群	若干可能解的集合
个体	**解**/个体	一个可能解的表现型，本例中即是十进制的X
染色体	**编码解**/染色体	一个可能解的基因型，本例即是X的二进制编码。$X = b_6b_5b_4b_3b_2b_1$，其中b_i=0或1。
基因	**基因**	解编码中的若干位的b_i
适应度	**适应度**	一个可能解接近最优解的一个度量，本例直接用F(X)作为其适应度的度量，其值越小越接近最优解
选择	选择	指从种群(解集)中依据适应度按某种条件选择某些个个体(可能解)
复制	复制	将一个解从一个解集复制到另一个解集
交配/杂交	交叉	即交配/杂交，是新可能解的一种形成方法，即是对两个可能解的编码通过交换某些编码位而形成两个新的可能解的遗传操作
突变	变异	也是新可能解的一种形成方法，它是通过随机地改变一个可能解的编码的某些片段(或基因)而使一个可能解变为一新的可能解的遗传操作

遗传算法求解过程的模拟



遗传算法基本框架及其设计要点

```
begin    /* 遗传算法 */
         t ← 0;     /* 进化的种群代数 */
         生成初始种群P(t);
         计算初始种群P(t)中每个个体的适应值;
         while（不满足终止条件） do
/* 利用下述操作生成新个体，并选择更优个体组成新种群 */
                  (1)通过复制、交叉或变异操作重组种群P(t)中
                  的个体，产生新个体，形成候选种群C(t);
/*注意此处C(t)并未包含P(t)中的个体 */
                  (2)计算C(t)中每个个体的适应值;
                  (3)根据适应值从C(t)和P(t)中选择更优的个体
                  组成新种群P(t+1);
                  (4) t ← t+1;
         end while
         选择P(t)中最优个体为所求的解;
end begin
```



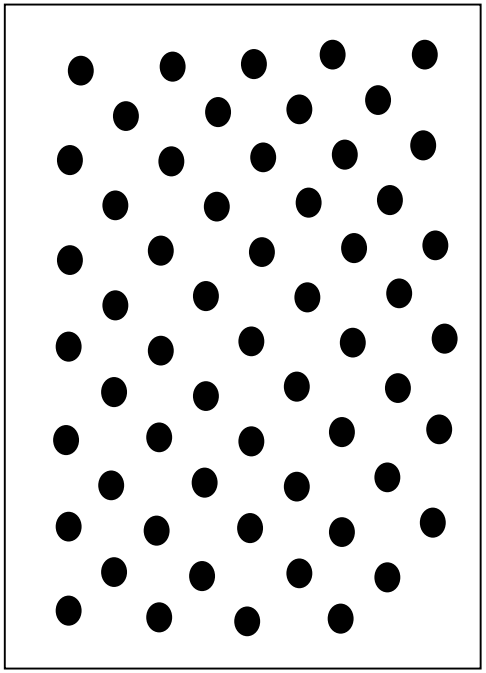
遗传算法
为什么可以求解NPC问题

遗传算法的基本思想

■NPC问题理论上可通过枚举-验证的遍历算法来实现

穷举法或称**遍历法**：对解空间中的每一个可能解进行验证，直到所有的解都被验证是否正确，便能得到精确的结果---**精确解**

可能是O(n!)或O(a^n)



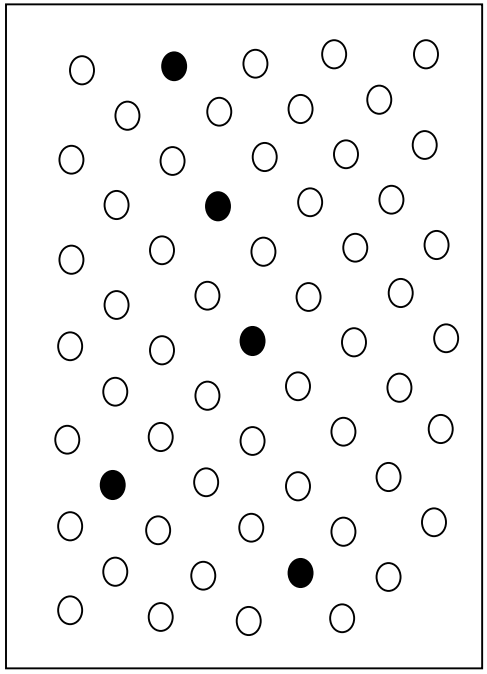
(a)穷举，遍历
搜索所有，可找到精确解

遗传算法的基本思想

■不求精确解，而求近似解-满意解，可采取随机搜索方法

随机搜索法：在解空间中随机选择一些可能解进行验证，求出所选择可能解(子解空间)中的最优解---**近似解**

- 基于概率论：随机选择
- 子解空间越大，求得满意解的可能性越大，但耗时也会加长
- 求出的近似解并不能保证是满意解



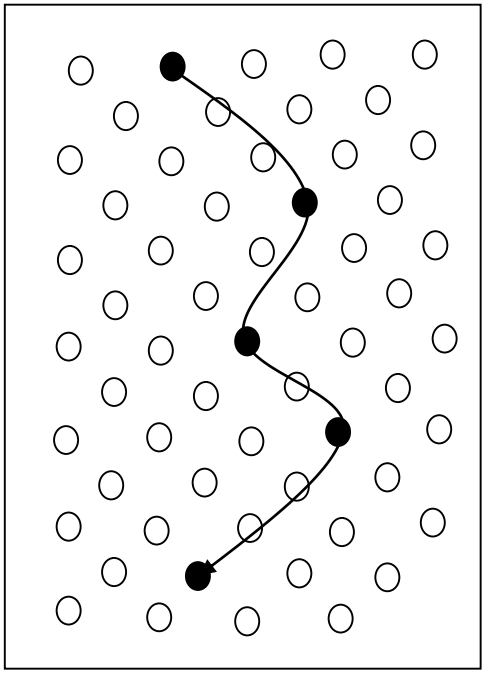
(b)完全随机搜索
随机点之间完全没有联系

遗传算法的基本思想

■为提高近似解的质量，可采取导向性随机搜索方法

导向性随机搜索法：对随机点的选取进行导向(导向到接近最优解的方向或路径)

- 基于概率论：随机选择
- 随机选择的可能解与前一可能解相比，更偏向于满意解
- 万一初始解就很差怎么办?



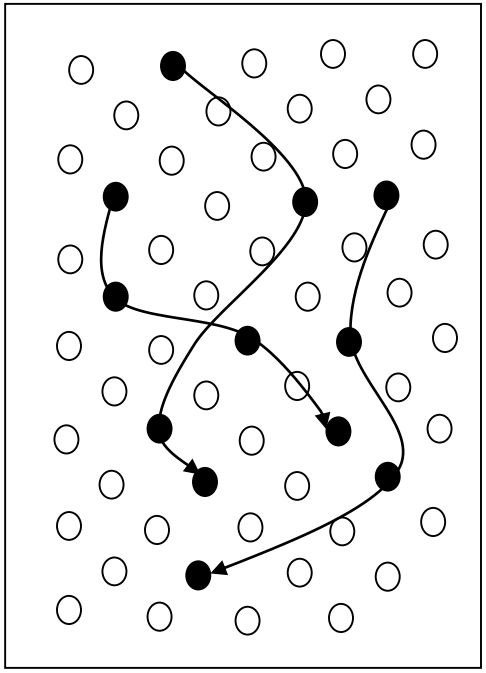
(c)导向性随机搜索
随机点之间形成一路径

遗传算法的基本思想

■为进一步提高近似解的质量，可采取导向性群(体)随机搜索方法

导向性群(体)随机搜索法：通时对多个随机点的选取进行导向(导向到接近最优解的方向或路径)，多条搜索路径

- 基于概率论：随机选择
- 多条路径下的最优，总比一条路径的最优要更优一些。
- 遗传算法就是这样一种导向性群随机搜索算法。
- 同一时刻多条路径上的解集合即为一个种群。多次选择，即多代进化。



(d)导向性群(体)随机搜索
多随机点同步进行导向性搜索

遗传算法的适用条件

■对于NP问题，当没有其他更好的算法可以使用时，可以考虑选择遗传算法。

■遗传算法的适用条件：

(1)已知“解空间”，即可能解的表现型和基因型

(2)关于可能解的“适应度”函数的计算方法(适应度用于判断一个可能解接近精确解的程度或方向)。

遗传算法提供了一种求解复杂系统问题的通用框架。

怎样用遗传算法求解具体的应用问题(I)

----问题理解与分析建模

----面向应用的遗传算法设计要点

----可能解编码方案的多样性

----交叉/变异规则的多样性与随机性

----遗传算法设计的其他方面

----仍需要思考的

现实世界的几个具体问题分析

例1：“会议室”租用问题
例2：“航班机组成员”选择问题
例3：“软件测试用例”选择问题

约束矩阵

会议室/机组成员/测试用例/讲座/航班/功能	x_1	x_2	x_3	x_4	x_5	x_6
T_1	0	0	0	1	1	0
T_2	0	1	1	0	0	1
T_3	1	0	0	0	1	0
T_4	0	1	0	1	0	1
T_5	0	1	1	0	0	0
T_6	1	0	0	0	1	0
费用 C_j	500	250	250	400	600	400

a_{ij}

$<x_1,x_2,\cdots,x_n>$

$$\min \quad z(x)=\sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i=1,2,\ldots,m$$

$$x_j \in \{0,1\}, \quad j=1,2,\ldots,n$$

其实它们是同一个问题：
一维的集覆盖问题

例4 “课程表”优化问题

- 有8 门课程需要安排教室，记为L_i，i=1,…,8；
- 有6 个教室可被使用，记为R_j，j=1,…,6；
- 要求每门课只能安排在一个教室，而每个教室最多只能安排两门课。
- 教室与课程班之间的约束矩阵A如下表给出，其中a_{ij}=1表示该课程班可以安排在该教室，a_{ij}=0则不可安排。
- **费用为教室容纳人数/课程班人数，即$C_{ij}=K_j/L_i$。**

教室(最大人数) / 课程(选课人数)	R1 (100 人)	R2 (50 人)	R3(多媒体) (50 人)	R4 (80 人)	R5(多媒体) (100 人)	R6(多媒体) (80 人)
L_1(85 人)	1	0	0	0	1	0
L_2(40 人)	1	1	1	1	1	1
L_3(95 人)	1	0	0	0	1	0
L_4(60 人)	1	0	0	1	1	1
L_5(45 人)	1	1	1	1	1	1
L_6(90 人)	1	0	0	0	1	0
L_7(76 人)	1	0	0	1	1	1
L_8(56 人)	1	0	0	1	1	1
K	100	50	70	80	120	100
$C_{ij} = K_j / L_i$						

约束矩阵

例1，例2，例3

一维集覆盖问题

可能解是一维向量 $<x_1,x_2,\cdots,x_n>$

使每一行都被选出的列覆盖，被哪一列或几列覆盖不重要，要满足约束矩阵

$$\min \quad z(x)=\sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i=1,2,\ldots,m$$

$$x_j \in \{0,1\}, \quad j=1,2,\ldots,n$$

例4

二维集覆盖问题

可能解是二维矩阵

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

使每一行都确定安排在某一列上，使被选中的行-列覆盖所有的行，要满足约束矩阵

$$\min f(x)=\sum_{i=1}^{8}\sum_{j=1}^{6} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{j=1}^{6} a_{ij} x_{ij} = 1, \quad for\ every\ i,\ i=1,\ldots,8$$

$$\text{s.t.} \quad \sum_{i=1}^{8} a_{ij} x_{ij} <= 2, \quad for\ every\ j,\ j=1,\ldots,6$$

$$x_{ij} \in \{0,1\};\ i=1,\ldots,8;\ j=1,\ldots,6$$

怎样用遗传算法求解具体的应用问题(II)

----问题理解与分析建模

----面向应用的遗传算法设计要点

----可能解编码方案的多样性

----交叉/变异规则的多样性与随机性

----遗传算法设计的其他方面

----仍需要思考的

遗传算法设计要点

NPC求解:

- 产生一个或一批可能解
- 判断可能解是否是问题的解

- 可能解的形式是怎样的？
  - 解的表现型和基因型
- 怎样产生待判定的可能解？
  - 交叉、变异
  - 随机(概率)
- 产生多少个待判定可能解？
  - 解集的规模
  - 进化的代数
  - 随机(概率)
- 怎样判定一个解是否是所求的解？
  - 适应度
  - 选择(标准)
  - 满意解

一组关于解的概念需要区分

$\langle x_1, x_2, \cdots, x_n \rangle$

$$\min \quad z(x) = \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i = 1, 2, \ldots, m$$

$$x_j \in \{0, 1\}, \quad j = 1, 2, \ldots, n$$

假设一个问题的解的形式为*x*，由*x*的取值空间或定义域给定的任何一个*x*值被称为**可能解**

而一个问题通常有很多个关于可能解的约束，即不是任何一个*x*值都满足约束，我们可将满足问题约束的那些*x*值称为**可行解**

而由一个算法在任何一组可行解中求出的最优解被称为是**近似解**

而符合用户期望的近似解被称为是**满意解**

所有可行解中的最优解是问题的**最优解**。

“可能解集合”⊃“可行解集合”⊃“近似解集合”⊃“满意解集合”⊃“最优解集合”

遗传算法设计要点

```
begin    /* 遗传算法 */
         t ← 0;     /* 进化的种群代数 */
         生成初始种群P(t);
         计算初始种群P(t)中每个个体的适应值;
         while（不满足终止条件） do
/* 利用下述操作生成新个体，并选择更优个体组成新种群 */
                  通过复制、交叉或变异操作重组种群P(t)中
                  的个体，产生新个体，形成候选种群C(t);
/*注意此处C(t)并未包含P(t)中的个体 */
                  计算C(t)中每个个体的适应值;
                  根据适应值从C(t)和P(t)中选择更优的个体
                  组成新种群P(t+1);
                  t ← t+1;
         end while
         选择P(t)中最优个体为所求的解;
end begin
```

例4问题的解的形式(表现型)与编码(基因型)

可能解的一般形式

教室 →

课程 ↓

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix}$$

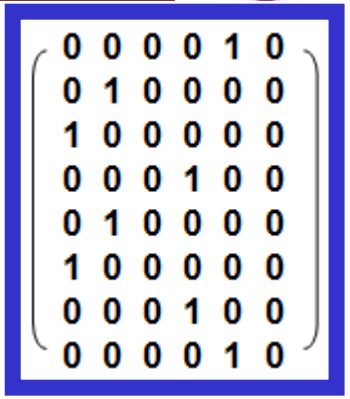
X_{ij}=1课程i被安排在教室j; =0课程未被安排在教室j

可行解1

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

可行解2

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



例4问题的**解的形式(表现型)与编码(基因型)**

针对具体问题，可以有不同的编码方案

行优先编码: 课程分段编码,一门课程相关的划为一段，有多少课程就有多少段

0 0 0 0 1 0	0 1 0 0 0 0	1 0 0 0 0 0	0 0 0 1 0 0	0 1 0 0 0 0	1 0 0 0 0 0	0 0 0 1 0 0	0 0 0 0 1 0
1 0 0 0 0 0	0 0 1 0 0 0	0 0 0 0 1 0	0 0 0 1 0 0	0 1 0 0 0 0	1 0 0 0 0 0	0 0 0 1 0 0	0 1 0 0 0 0

列优先编码: 教室分段编码, 一个教室相关的划为一段，有多少教室就有多少段

0 0 1 0 0 1 0 0	0 1 0 0 1 0 0 0	0 0 0 0 0 0 0 0	0 0 0 1 0 0 1 0	1 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0		
1 0 0 0 0 1 0 0	0 0 0 0 1 0 0 1	0 1 0 0 0 0 0 0	0 0 0 1 0 0 1 0	0 0 1 0 0 0 0 0	0 0 0 0 0 0 0 0		

不同的编码方案，交叉、变异的规则也可能不同

怎样用遗传算法求解具体的应用问题(III)

----问题理解与分析建模

----面向应用的遗传算法设计要点

----可能解编码方案的多样性

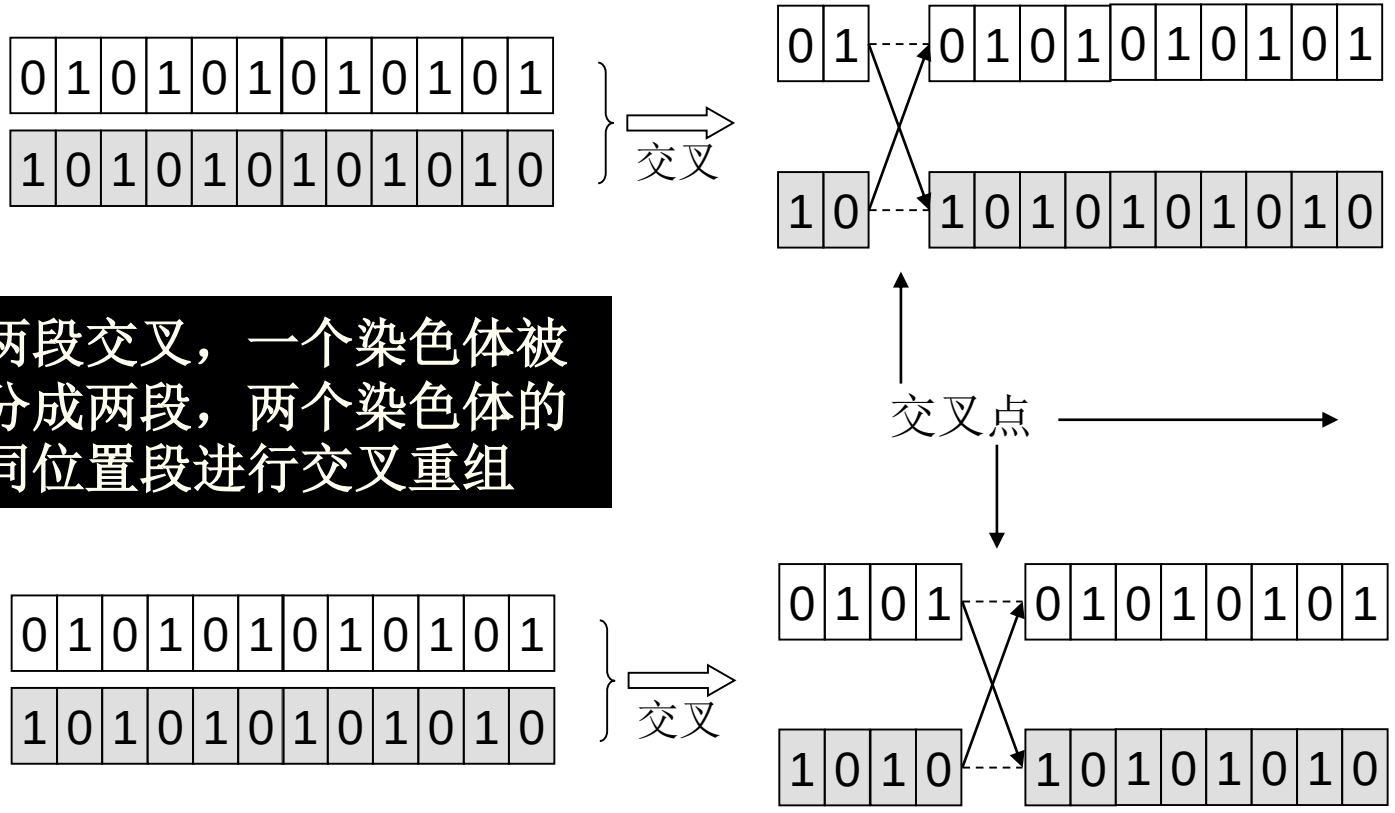
----交叉/变异规则的多样性与随机性

----遗传算法设计的其他方面

----仍需要思考的

遗传算法的交叉规则设计

产生新的待判定的可能解 ← 交叉 → 两段交叉

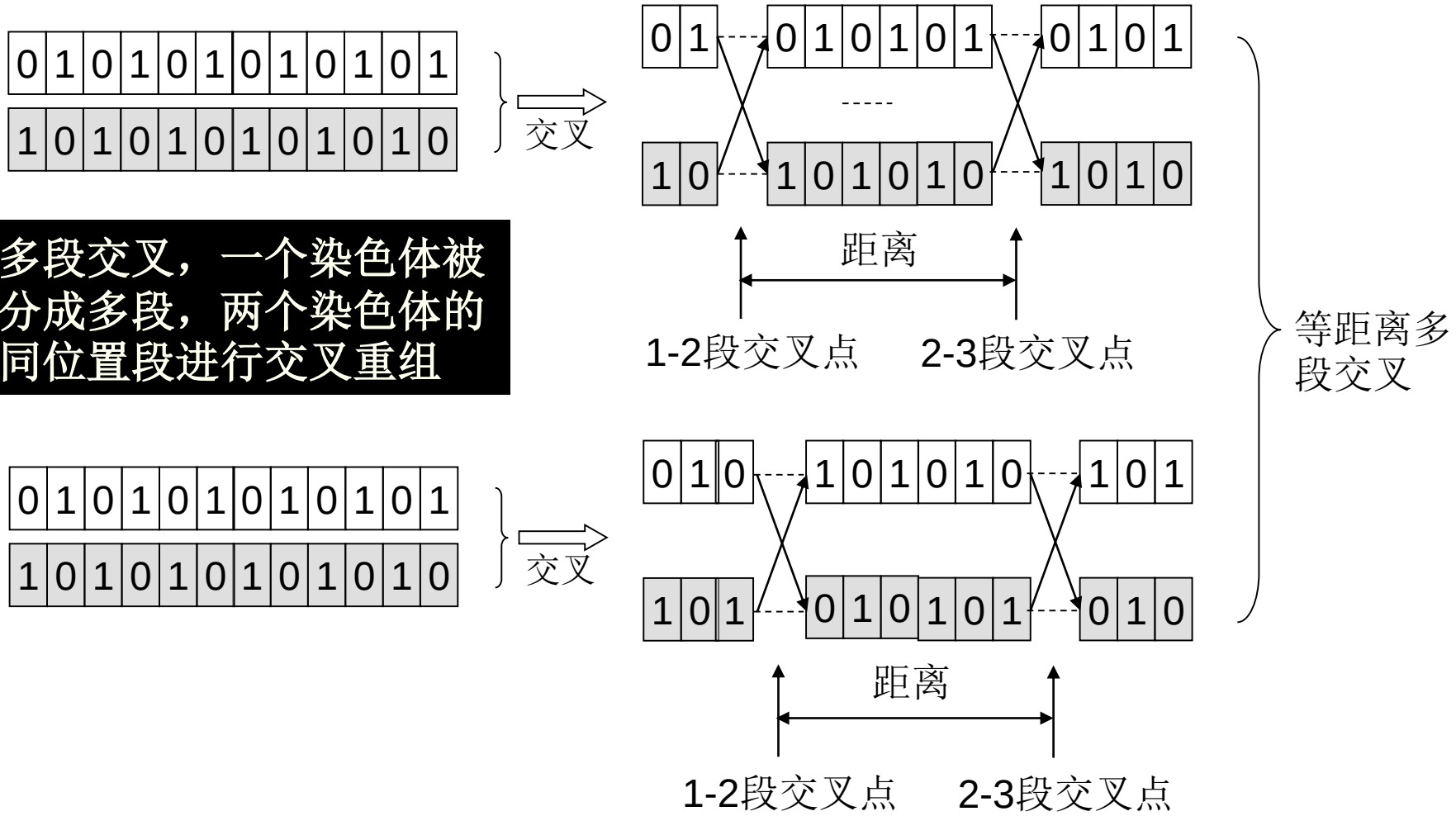


两段交叉，一个染色体被分成两段，两个染色体的同位置段进行交叉重组

是否只有这一种交叉方案呢?

遗传算法的交叉规则设计

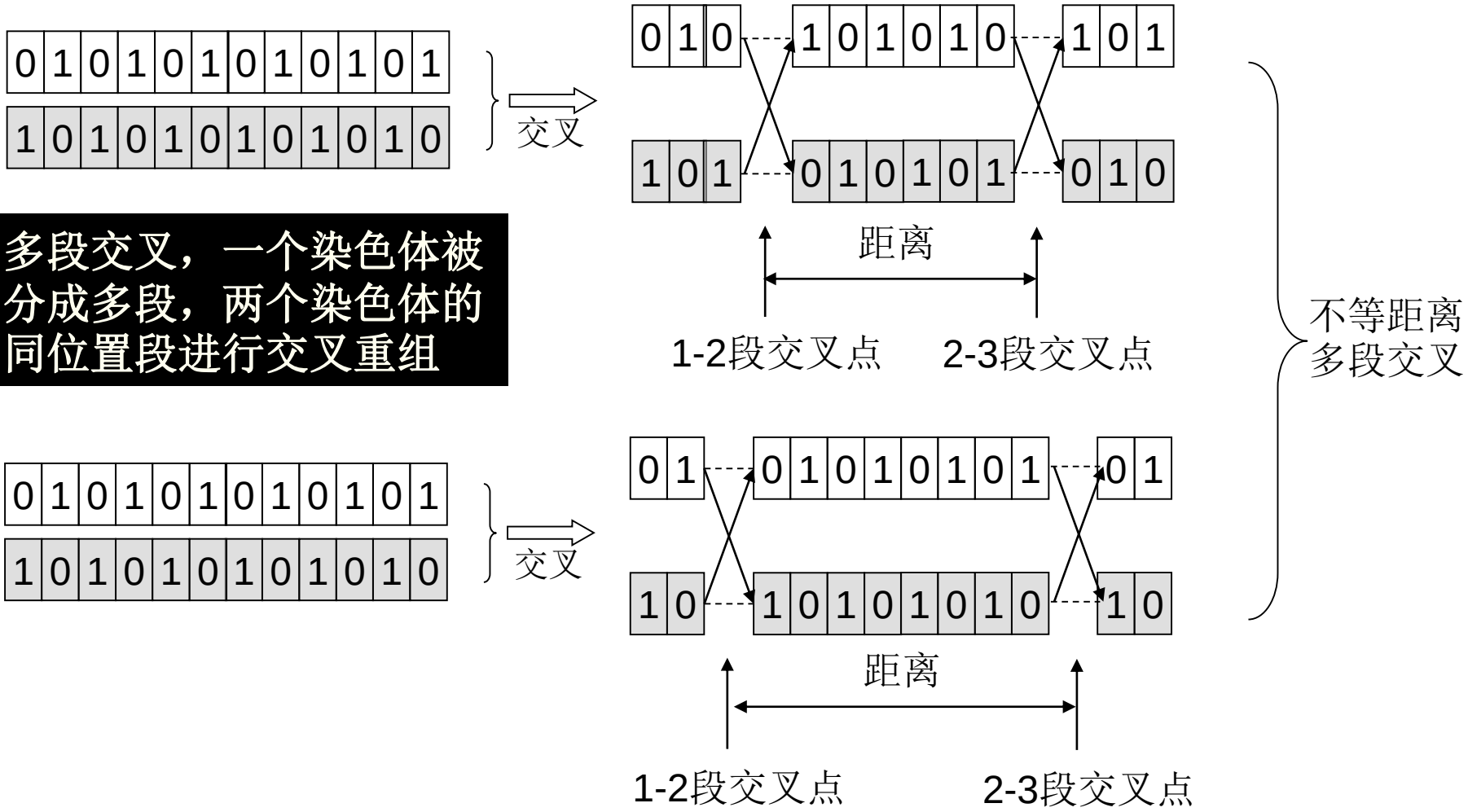
产生新的待判定的可能解 ← 交叉 → 等距离多段交叉



多段交叉，一个染色体被分成多段，两个染色体的同位置段进行交叉重组

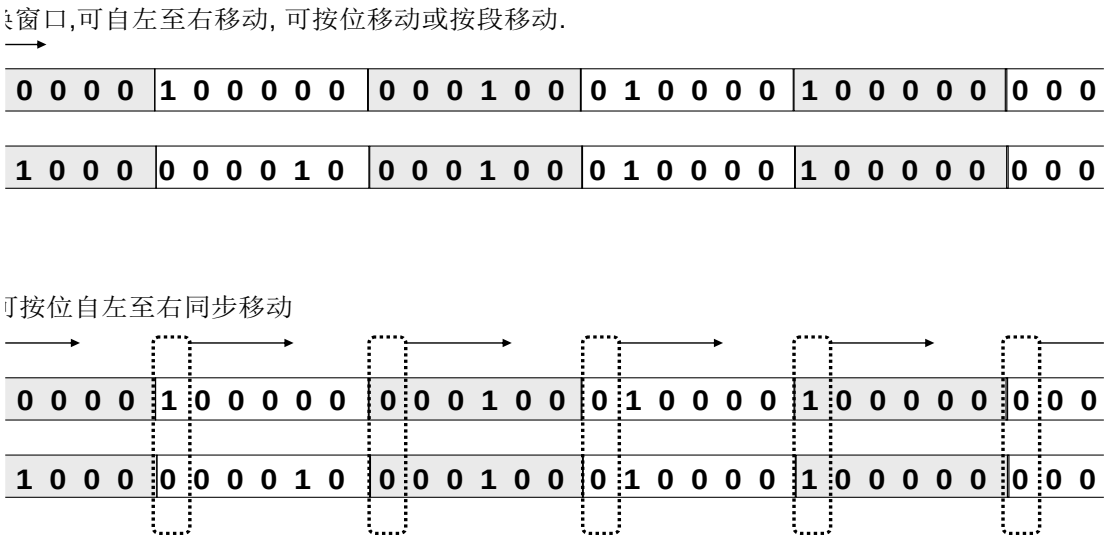
遗传算法的交叉规则设计

产生新的待判定的可能解 ← 交叉 → 不等距离多段交叉

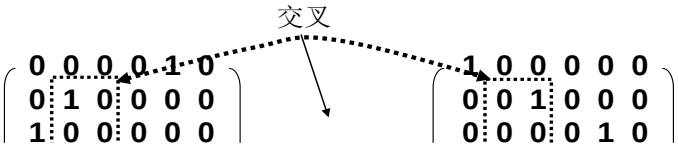


多段交叉，一个染色体被分成多段，两个染色体的同位置段进行交叉重组

结合具体问题的解的编码方式 ➔ 交叉：点交叉、行交叉、列交叉、块交叉



行交叉是指对两个染色体同位置的两个课程段基因进行交叉重组
列交叉是指对不同段的相同位置的两个染色体的基因进行交换



块交叉则是对两个染色体的任何位置的两个块基因进行交换
点交叉则是对两个染色体按某一概率交换其位基因

交叉与随机

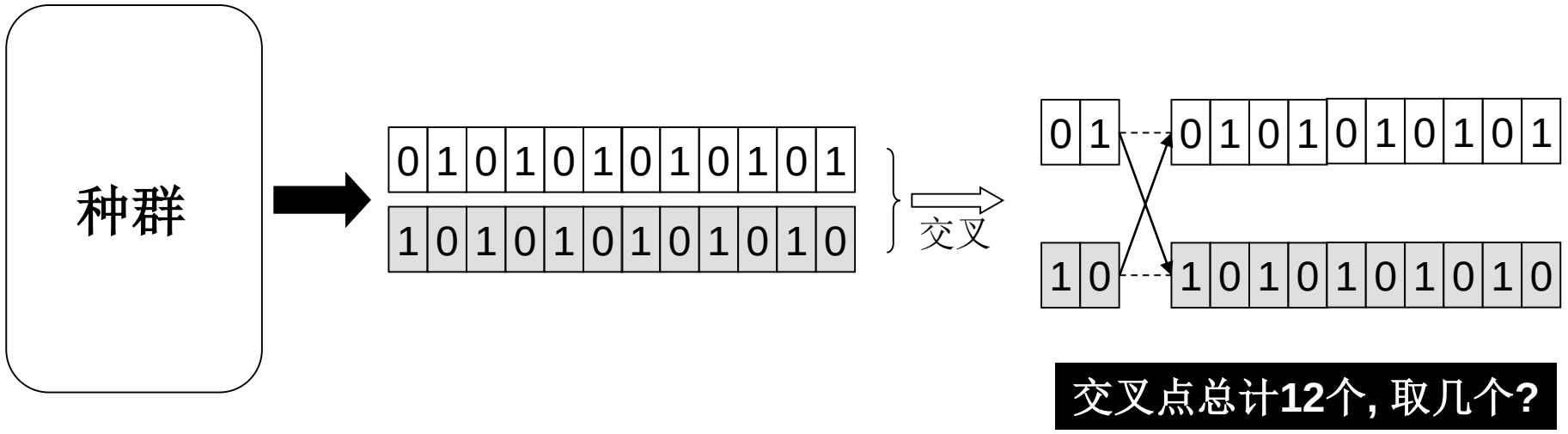
交叉操作本质上是一种组合，其组合所形成的可能解空间依然是庞大的，难以确定性的遍历每个组合。

基于概率的随机处理方法也是遗传算法的核心处理机制

交叉概率：在一个群体中，个体被选择出进行交叉的概率

交叉点的选择：可通过随机方式产生

子空间中的待判定解的选择：可通过随机方式产生



交叉点总计12个，取几个?

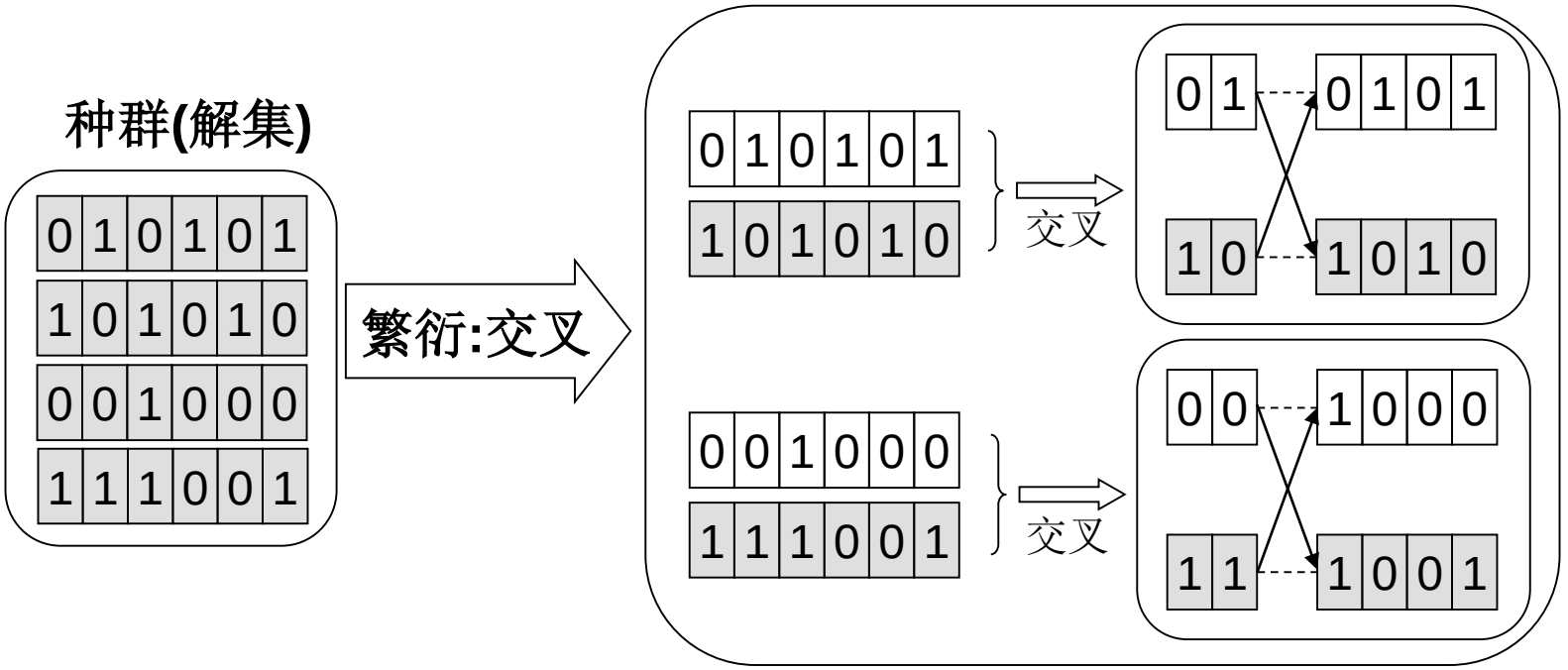
交叉与随机

(1)种群中个体的配对分组问题，即哪两个个体进行配对交叉;

(2)两段交叉中交叉点位置的选择;

(3)多段交叉中的交叉点距离的变化与不变，即两个或多个交叉点间等距离和不等距离的多段选择问题;

(4)结合问题解编码的交叉与随机策略;



交叉组合产生新可能解的方式是变化多样的

交叉与随机

交叉策略按前述的各种策略进行，每一种策略都是一子解空间。
随机可依据问题选择不同的概率模型，进行处理。

概率模型的不同、交叉策略选择的不同构成了丰富多彩的遗传算法

交叉操作I：点交叉

设P_1和P_2表示两个n位的父代染色体，$f(P_1)$和$f(P_2)$分别表示两个父代的适应值，C表示子代染色体。点交叉操作如下：

step1: i=1

step2: 如果$P_1[i]= P_2[i]$，则$C[i]= P_1[i]= P_2[i]$

step3: 如果$P_1[i]\neq P_2[i]$，则

step3.1: 以概率$p= f(P_1)/(f(P_1)+f(P_2))$ 让$C[i]:=P_1[i]$

step3.2: 以概率1-p让$C[i]:= P_2[i]$

step4: 如果i =n，停止；否则i=i+1，返回step2

交叉与随机

交叉操作II：行交叉

设P_1和P_2表示两个k*h位的父代染色体，其中k为每一段的位数，h为染色体的分段数目，$f(P_1)$和$f(P_2)$分别表示两个父代的适应度值，C表示子代染色体。

step1: 产生一个0至h-1的随机数x；

step2: 如果P_1[x*k+i]=P_2[x*k+i] for all i=1,…,k，则不产生后代；

step3: 如果P_1[x*k+i]≠P_2[x*k+i] for any i=1,…,k，则以概率p= $f(P_1) / (f(P_1) + f(P_2))$ 让P_1[x*k+i]与P_2[x*k+i] for i=1,…,k交换；

II注：两个染色体的x段如相同，则不交换，否则以概率p进行交换。

交叉与随机

交叉操作III：列交叉

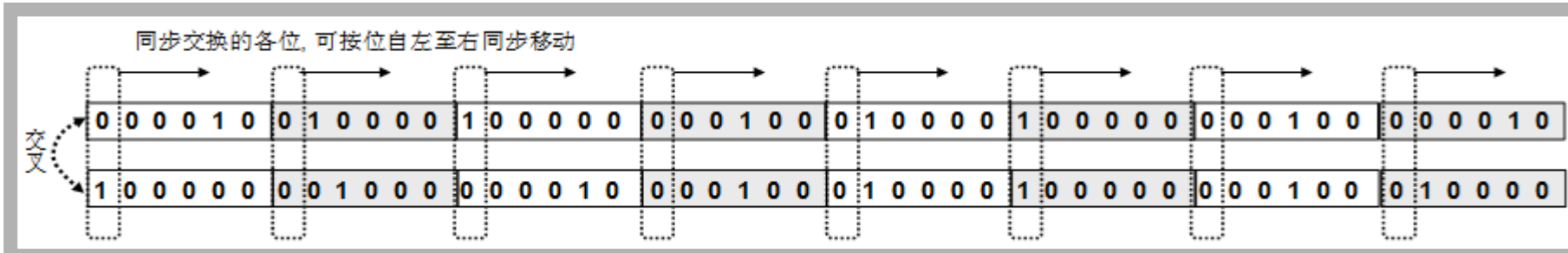
设P_1和P_2表示两个k*h位的父代染色体，其中k为每一段的位数，h为染色体的分段数目，$f(P_1)$和$f(P_2)$分别表示两个父代的适应度值，C表示子代染色体。

step1: 产生一个0至k-1的随机数x；

step2: 如果$P_1[i*k+x]=P_2[i*k+x]$ for all i=1,…,h，则不产生后代；

step3: 如果$P_1[i*k+x]\neq P_2[i*k+x]$ for any i=1,…,h，则以概率$f(P_1)/(f(P_1)+ f(P_2))$让$P_1[i*k+x]$与$P_2[i*k+x]$交换 for all i=1,…h；

//注：两个染色体的各段的x位如都相同，则不交换，否则以概率p进行交换。



交叉与随机

交叉策略比较：

“点交叉”将覆盖**更大的**可能**解空间**，产生更多种新可能解，增加获得最优解的机会；

“行交叉”、“列交叉”大大地**压缩**了可能解的**解空间**，产生的可能解都是可行解(编码方案将约束条件考虑进去了)；

策略的选择需要折中：选择搜索更加广泛的解空间呢，还是选择压缩搜索空间呢？需要折中。

不同的交叉策略对算法的求解质量和收敛速度等方面是有影响的，没有一种设计能够面面俱到，因此如何在算法不同方面性能之间权衡，也是遗传算法设计过程的关键所在，体现了一定程度的技术性和艺术性。

怎样用遗传算法求解具体的应用问题(IV)

----问题理解与分析建模

----面向应用的遗传算法设计要点

----可能解编码方案的多样性

----交叉/变异规则的多样性与随机性

----遗传算法设计的其他方面

----仍需要思考的

遗传算法设计的其他问题

变异操作是对群体中的某些个体染色体的某些基因进行突变处理

- **变异概率**(PM, probability of mutation)，控制算法中变异操作的使用频率。
- **变异操作的基本步骤**：

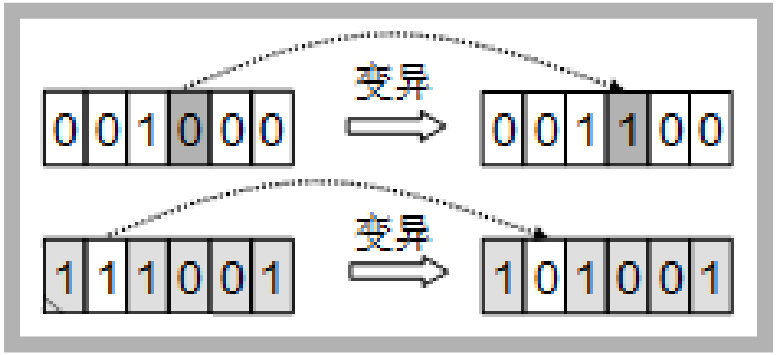
  a)对种群中所有个体以事先设定的变异概率判断是否进行变异；

  b)对进行变异的个体随机选择变异位置进行变异。

- **引入变异操作的目的**：

一是使遗传算法具有**局部的随机搜索能力**。

二是使遗传算法**可维持群体多样性**。



遗传算法设计的其他问题

适应度函数的选择

主要考察其是否能度量一个可能解接近最优解的程度和方向。

初始种群中的个体通常是随机产生的。

初始种群的规模与个体解，可依据问题解空间的分布特性来选择。

终止条件通常有以下几种：

(1)进化次数限制----进化到指定的代数即可终止算法；

(2)计算耗费的资源限制（如计算时间、计算占用的内存等）**--**当达到一定的资源占用量时可终止算法，如当产生超过一定数量的不重复可行解后即可终止；

(3)某一个个体已经满足最优值的条件，即最优值已经找到；

(4)适应度已经达到饱和，继续进化不会产生适应度更好的个体；

(5)人为干预；

(6)以上两种或更多种的组合。

思考1：遗传算法的收敛速度和解的质量有什么关系呢？

解的质量，可以使用“近似率”来衡量，所谓近似率是指算法求得的解与问题最优解的近似程度。

收敛速度，是指对于具有迭代特征的近似算法，在迭代多少次后能够使得结果稳定(通俗来讲，即结果不再随进一步迭代而发生变化或发生极小的可以被忽略的变化)，它从一定程度反映了算法求解的“快慢”

“在执行相同次数的迭代后，近似率高的算法更好”

“在达到期望的满意解的前提下，迭代次数越少越好”

算法的比较：当不同算法均应用多次后，求得满意解次数越多的算法越好！

思考2：遗传算法各项参数对算法收敛速度和解的质量有什么影响？各次迭代的种群规模大小，尤其是初始种群规模大小对算法的性能有什么影响呢？

■不同的交叉变异规则反映了算法对可能解空间的覆盖范围，覆盖范围越大则获得最优解的概率也越大。

■此外，如变异率、交叉率等，对算法的性能又有什么影响呢？

思考3：染色体编码中什么是“遗传基因”，怎样发现种群的遗传基因，怎样使遗传基因被遗传被继承？

一般，染色体基因就是一个二进制位。一个种群的优质个体的编码中：

■“值相同位”越多的是否就是遗传基因呢？

■“0、1组合相同的片段”越多是否就是遗传基因呢？例如

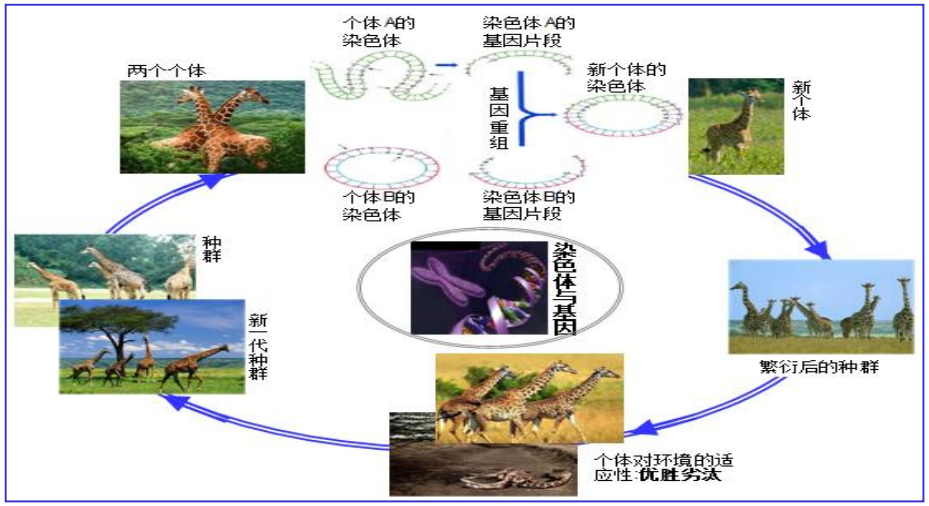
{00101011, 00101001, 10100101, 01100100}

自左而右第3-4位“10”有4个相同，它是否是遗传基因呢？

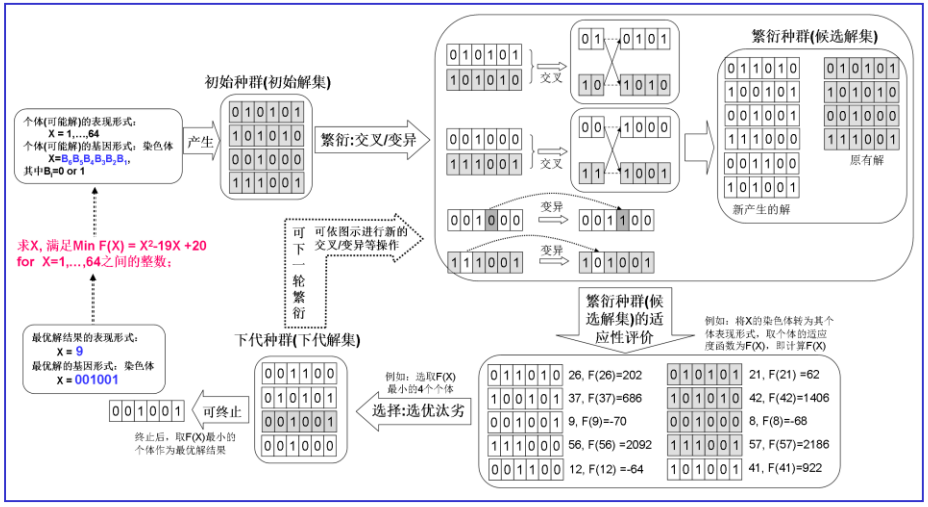
■发现了遗传基因后，在交叉变异重组时又怎样使其不受破坏呢？

本讲小结

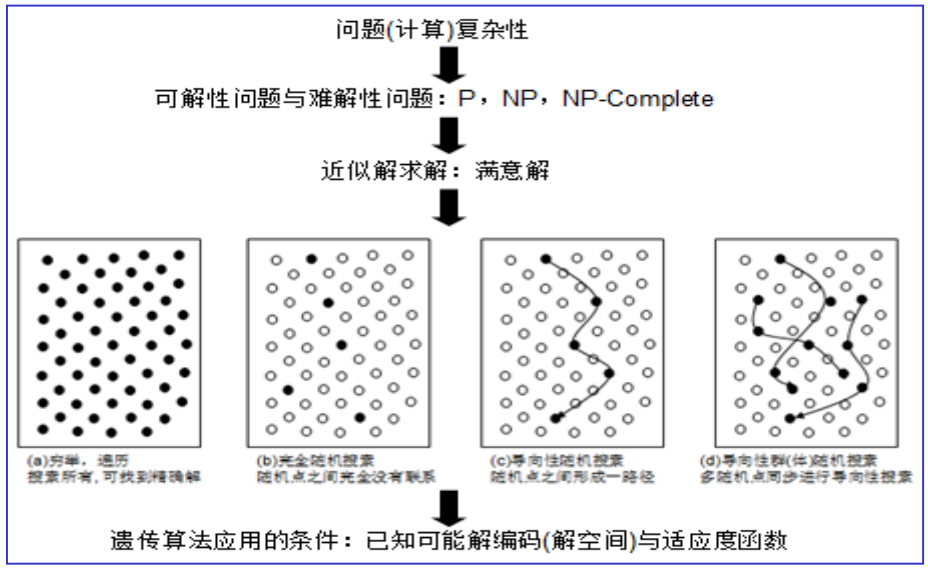
社会/自然现象(遗传与进化)



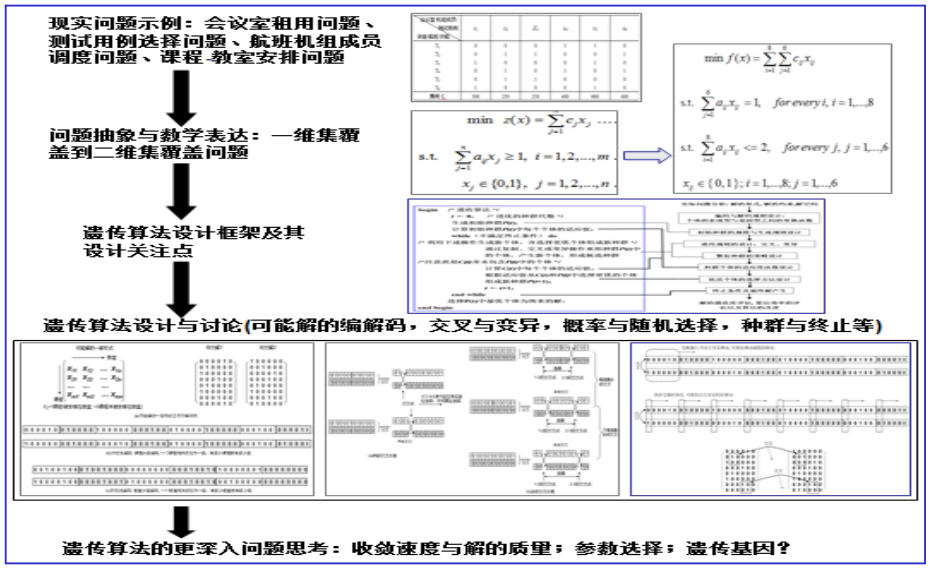
计算学科的(遗传)算法



(遗传)算法的本质



(遗传)算法的应用



本讲小结

社会/自然现象(遗传与进化)

计算学科的(遗传)算法

社会/自然/生活

宽度学习

计算与计算系统(浅层次)

深度学习

计算与计算系统(深层次)

本质

应用

由遗传算法的学习，学会算法的研究，更重要的是养成科学研究良好的习惯

(遗传)算法的本质

(遗传)算法的应用

第9讲 怎样研究算法：遗传算法示例

Questions & Discussion?

2015