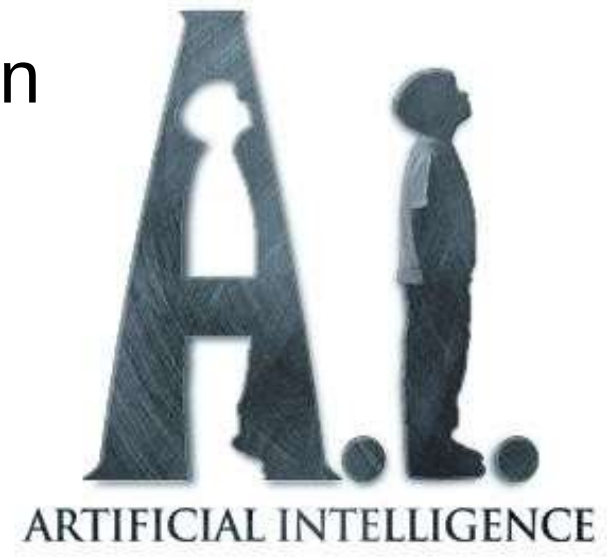
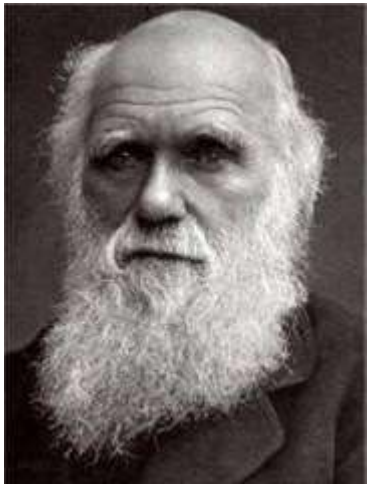
Ch.4 Computational Intelligence
第四章　智能计算 II

4.3 Evolutionary Computation
演化计算



进化计算

- 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(***Natural selection***)
- 演化过程结果反映在**个体结构**上
- 生物通过个体间的选择(***selection***)、交叉(***crossover***)、变异(***mutation***)来适应大自然环境。
- 演化算法/进化计算(*Evolutionary Algorithms*)
  - **遗传算法 GA**
  - 进化策略 ES
  - 进化编程 EP
  - 遗传编程 GP



Charles Darwin

进化计算

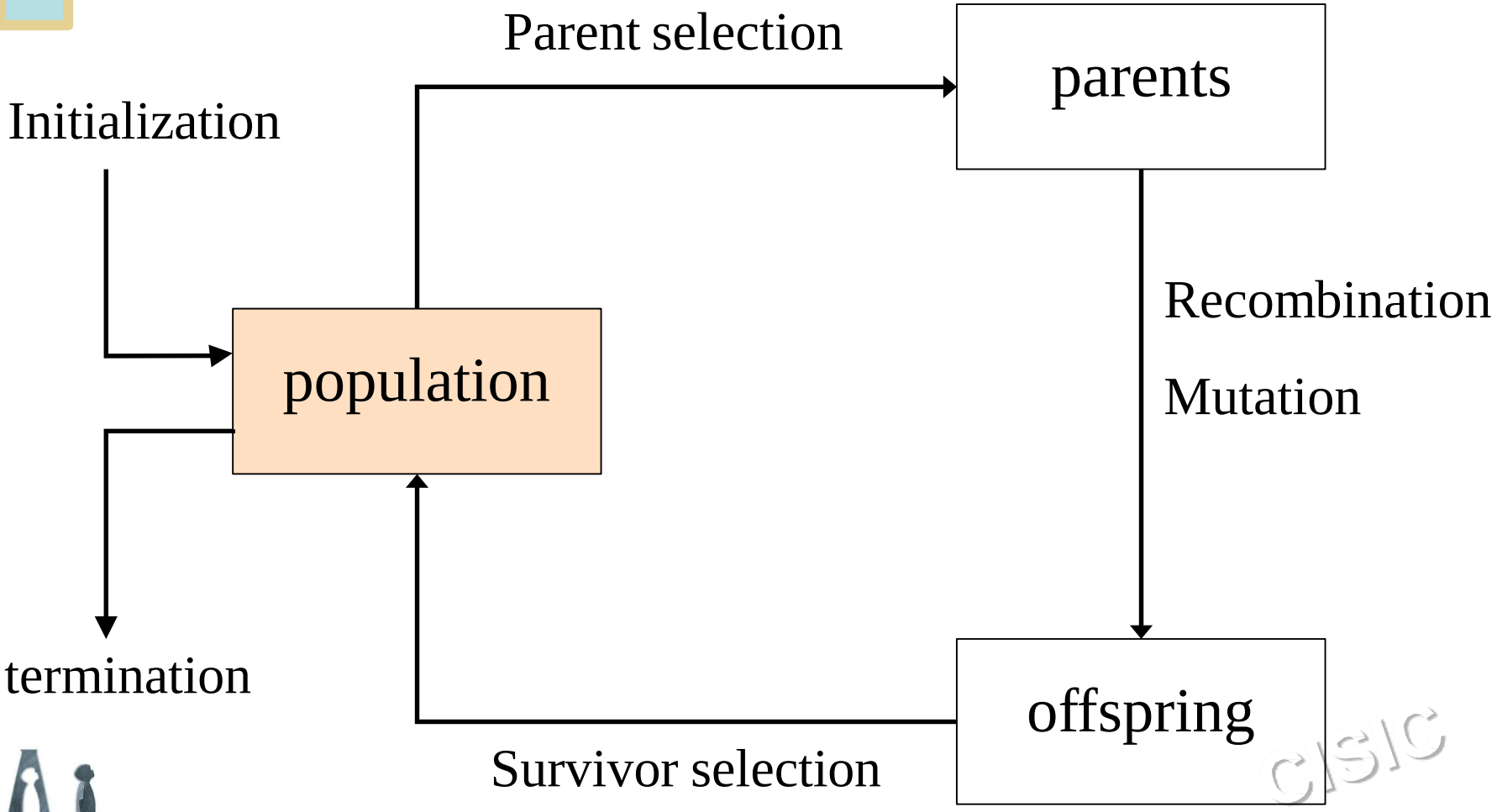
- 20世纪60年代以来，**如何模仿生物**来建立功能强大的算法，进而将它们运用于**复杂的优化问题**，越来越成为一个研究热点。进化计算在这一背景下蕴育而生
  - EAs fall into the category of "generate and test" algorithms
  - They are stochastic(随机的), population-based (基于种群的) algorithms

进化计算
—What are EAs

- 一种随机搜索启发式算法
  - 模拟由个体组成的群体的集体学习过程；
  - 同时也是启发式的，不过
    - 运行时间没有执行上界，
    - 不能保证一定找到最优解，
    - 并且不能保证解很好.

进化计算

——一般框架


Parent selection
parents
Initialization
Recombination
Mutation
population
termination
offspring
Survivor selection

进化计算

— When to use EAs?

- Benefits of Evolutionary Algorithms:
  - easy to apply
  - easy to implement
  - easy to test
  - often deliver satisfactory results in acceptable time
  - not much harm done, if no success
- Use, if
  - no better algorithm is known,
  - there is no time to develop a problem-specific algorithm, computation time vs. time for development
  - there is no expertise to develop a problem-specific algorithm.

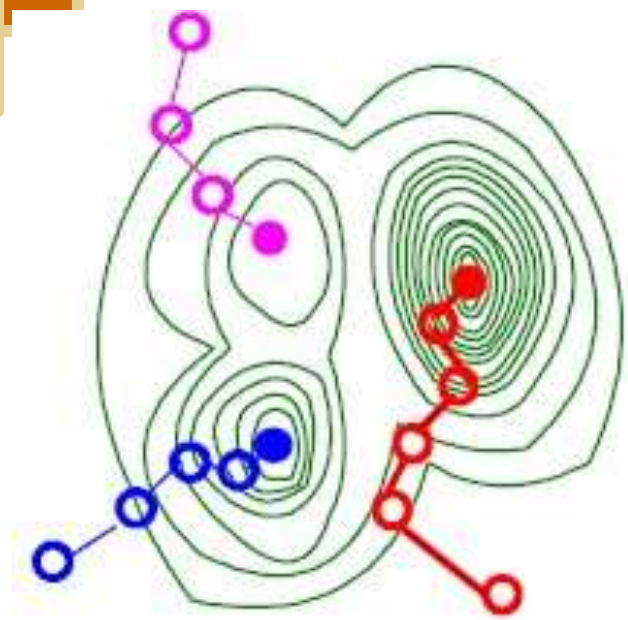
进化计算

— General Limits

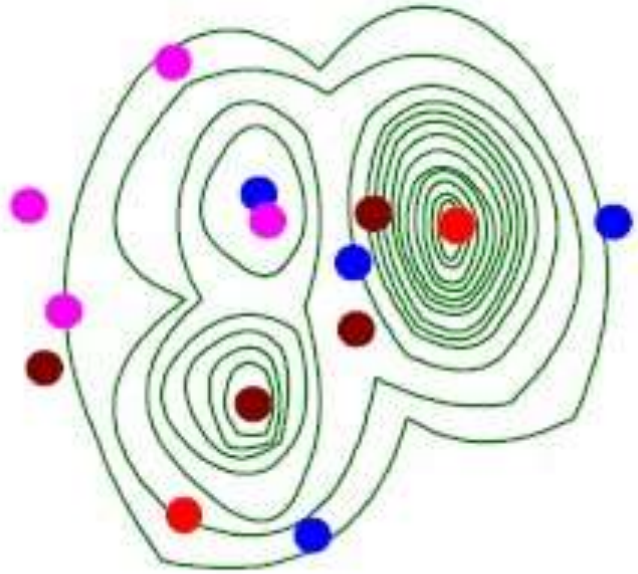
- 算法簇EA family
  - Genetic Algorithms
  - Evolution Strategies
  - Evolutionary Programming
  - Genetic Programming
- 哪个最优?
  - “There ain’t **n**o such thing as a **f**ree **l**unch.”没有免费的午餐
  - **NFL-Theorem**: “On average, all randomized search heuristics perform equal.”

进化计算

— EAs vs. SST



Single Search Technique



Genetic Search

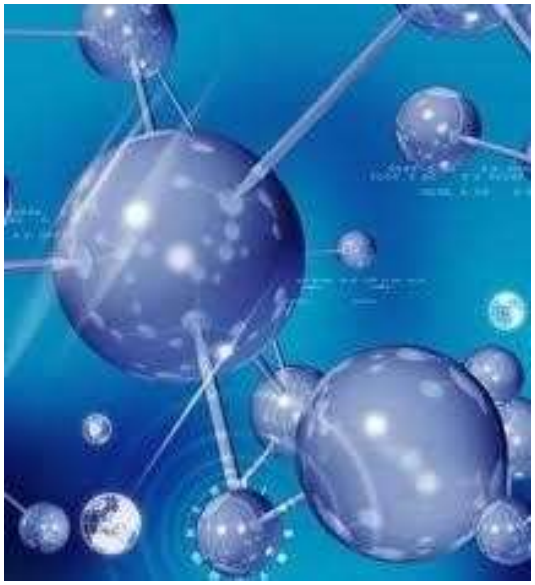
- **单一搜索技术(SST)**可能会陷入不需要的最小值.
- SST的收敛性取决于所选的初始条件.

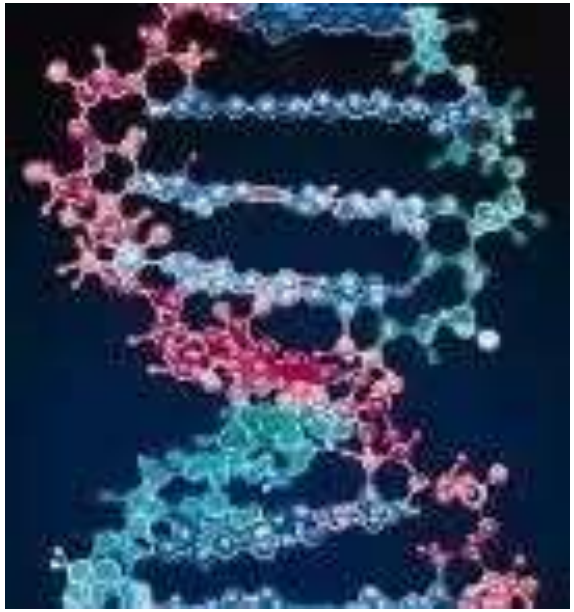
进化计算

— How to Build an EA

- A number of steps that we have to perform
  - 设计合适的表示方法
  - 确定如何初始化种群
  - 设计从基因型到表现型的映射方法
  - 设计个体的进化方法
  - 设计合适的变异操作
  - 设计合适的重组操作
  - 确定如何管理种群
  - 确定如何选择双亲个体
  - 确定如何选择被替换的个体
  - 确定何时终止进化

遗传算法
Genetic Algorithm, GA





主要内容

- 遗传算法的理论基础
- 遗传算法简介
- 遗传编码与解码
- 适应度函数
- 遗传操作
- **Demo**
- 遗传算法的控制参数
- 遗传算法的主要问题
- 遗传算法示例

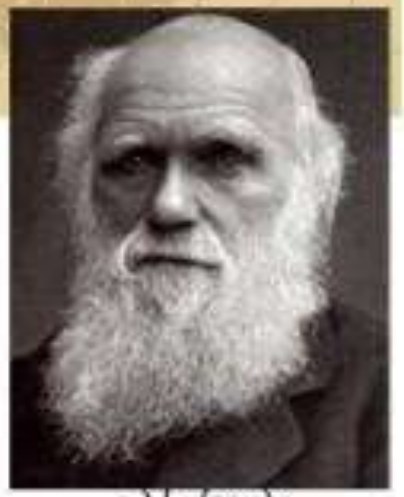
1.遗传算法的理论基础

- 进化论(Evolutionism)
  - Charles Darwin (1809-1882)
    - Extremely controversial and influential book (1859)—"Origin of Species"
    - "物尽天择，适者生存"

It is not the strongest of the species that survive, but the one most responsive to change.

—— *Charles Darwin*

- 遗传学说(Genetics)
  - Gregor Mendel (1822-1884)
    - Inheritance of characteristics



达尔文
(Charles Darwin)

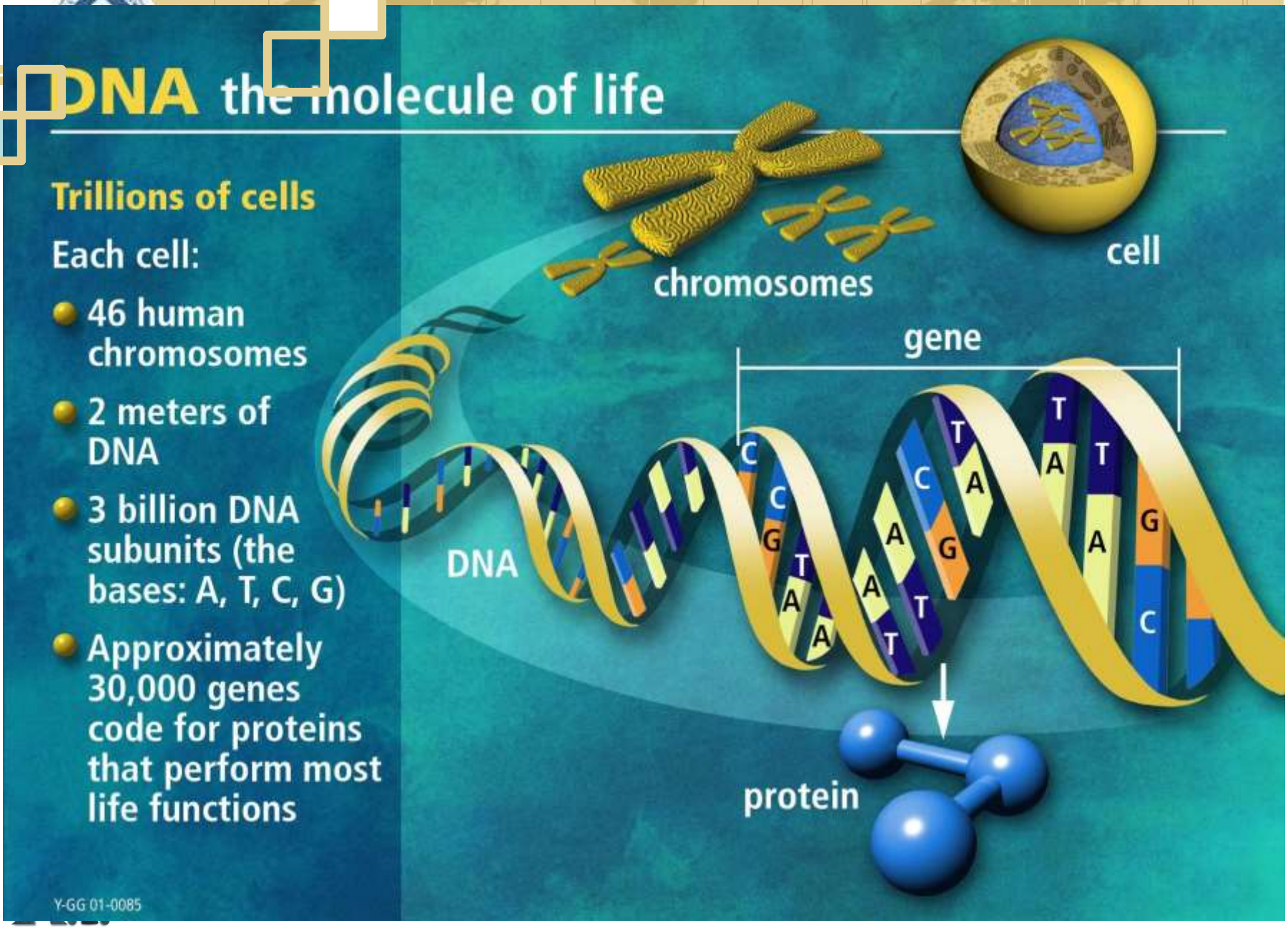
孟德尔
(Gregor Mendel)

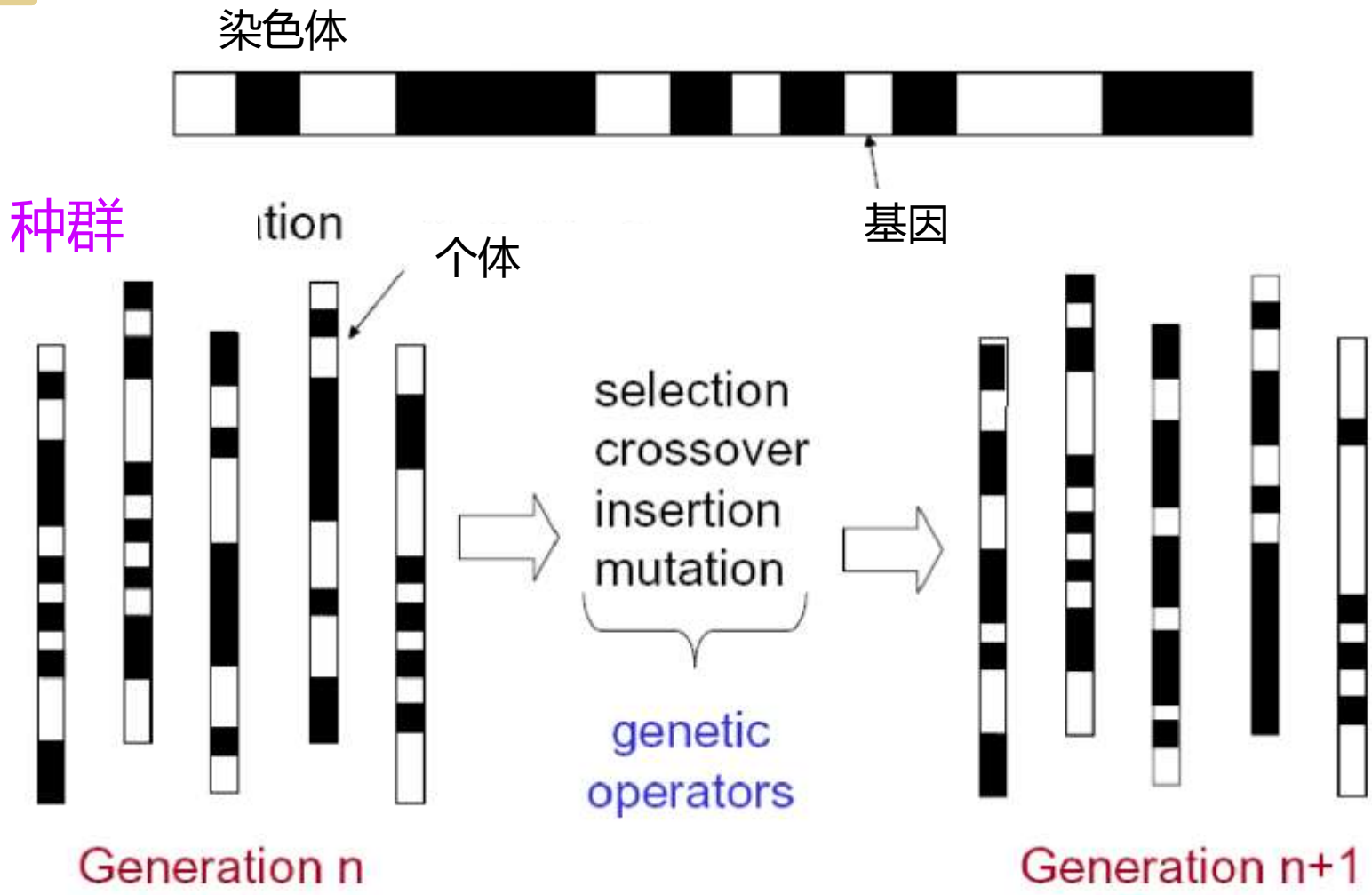
遗传算法的理论基础

- 进化论
- 遗传学
  - Gregor Mendel
  - … the great unknown …. Walter Sutton (1877-1916)
    - 基因学说
  - Hugo de Varis (1848-1935)
    - 突变学说

萨顿
(Walter Sutton)

德弗里斯
(Hugo de Varis)


DNA the molecule of life
Trillions of cells
Each cell:
46 human chromosomes
2 meters of DNA
3 billion DNA subunits (the bases: A, T, C, G)
Approximately 30,000 genes code for proteins that perform most life functions
chromosomes
cell
gene
DNA
protein
Y-GG 01-0085


染色体
基因
种群
ıtion
个体
selection
crossover
insertion
mutation
genetic
operators
Generation n
Generation n+1

GA Terminology

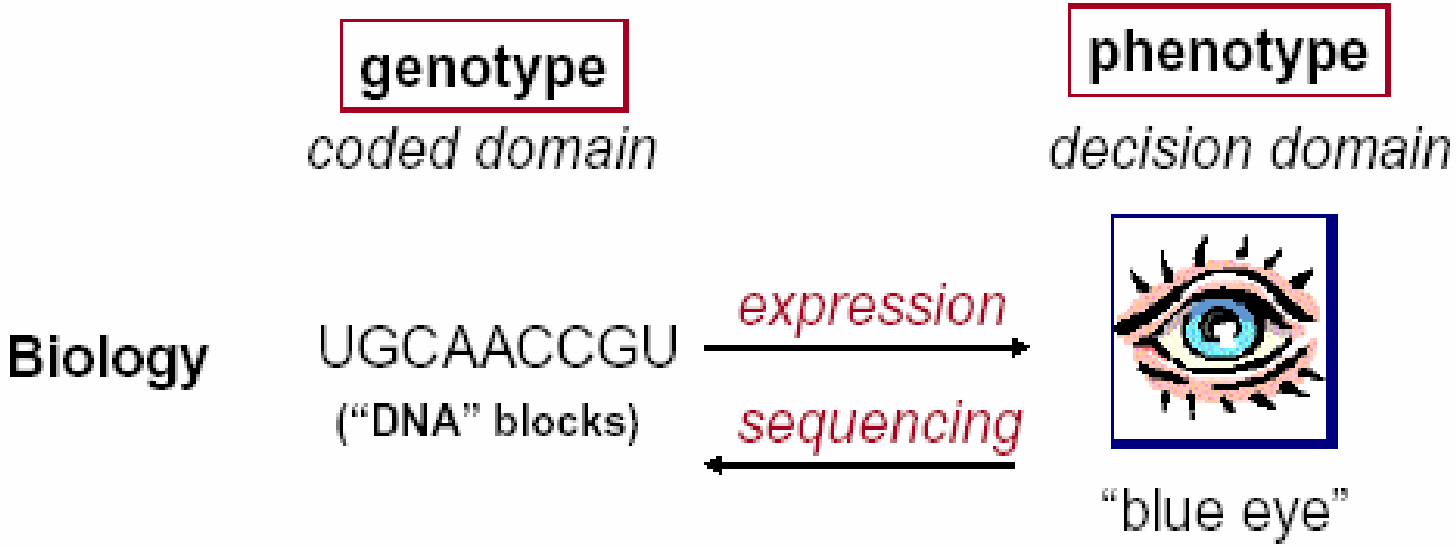
- Individual 个体：GA所处理的基本对象、结构。
- Population 群体
  - 若干个体的集合
  - 问题的一些解的集合，如p1={x1,x2,…,x100}
- Chromosome染色体（Bit String 基因链码、Individual 个体）
  - 决定生物性状
  - 个体的表示形式，一个染色体代表问题的一个解

  如：1552对应的染色体1100001000
- Gene基因：染色体中的元素
- Locus 基因位：某一基因在染色体中的位置。

Basic principles

- 个体的特征是一组参数： Genes(基因)组成
- 基因组成一个串: Chromosome(染色体)
- 染色体构成genotype(基因型)
- 其内部表现（即基因型）是某种基因组合，它决 定了个体的形状的外部表现: the phenotype(表现型)
- Fitness (适应度)：度量某个物种对于生存环境的适应程度

Example


genotype
coded domain
phenotype
decision domain
Biology
UGCAACCGU
("DNA" blocks)
expression
sequencing
"blue eye"

GA Introduction

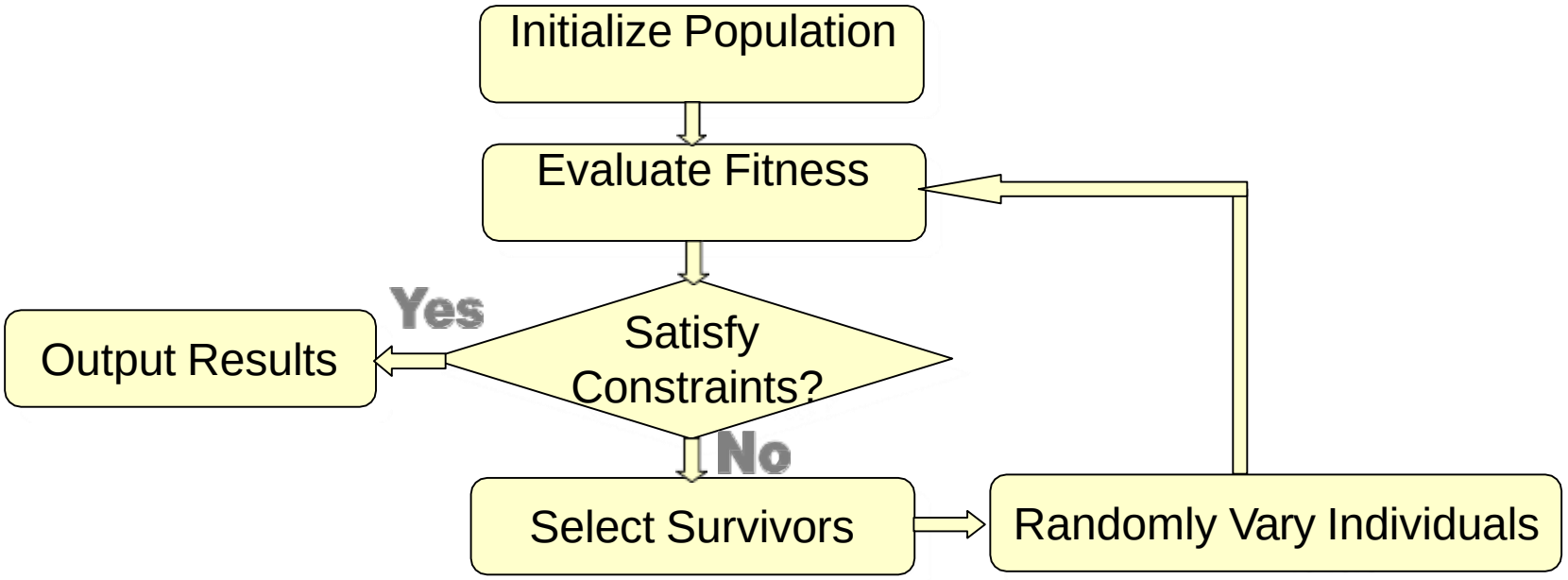
Inspired by natural evolution

- Population of individuals (种群)
  - 代表着一个问题的可行解
- 每个个体都有一个适应度函数Fitness function
  - 适应度越高，说明越适应环境
- 根据他们的适应度, 选择父母为新一代繁殖后代
  - 更健康的个体有更多的机会繁殖
  - 产生的种群会代替原来的种群，原来的种群就死亡了
- 子代具有双亲的特性，原来种群中两个个体的重组
- If well designed, population will converge to optimal solution

(收敛)

GA's Principle

- 模拟自然界优胜劣汰的进化现象，把搜索空间映射为遗传空间，把可能的解编码成一个向量——染色体，向量的每个元素称为基因。通过不断计算各染色体的适应值，选择最好的染色体，获得最优解。



Simple Genetic Algorithm (SGA)

BEGIN

生成一个初始化种群；

计算每个个体的适应度；

重复

选择

交叉

变异

鼓励进化，降低多样性

产生适当的多样性，避免早熟

评价新的种群

直到收敛，或者达到终止条件

END

Genetic Algorithm Steps

- Representation (**Encoding**)
- Initialize a population
  - **randomly**
- How to select individuals to be parents (**Fitness Evaluation**)
- **Survivor Selection**
- Reproduction
- When to stop the algorithm

(1) Enceding

遗传算法不能直接处理问题空间的参数，必须把问题的解 的参数形式转换成基因链码的表示形式，这一转换操作叫编码。

编码原则：

①完备性: 问题空间的所有可能解都能表示为所设计的基 因链码形式。

②健全性:任何一个基因链码都对应于一个可能解。

④非冗余性: 问题空间和表达空间一一对应。

[说明]

很难设计同时满足上述3个性质的编码方案，但完备性是必须的。

对于具体应用，应具体选择。

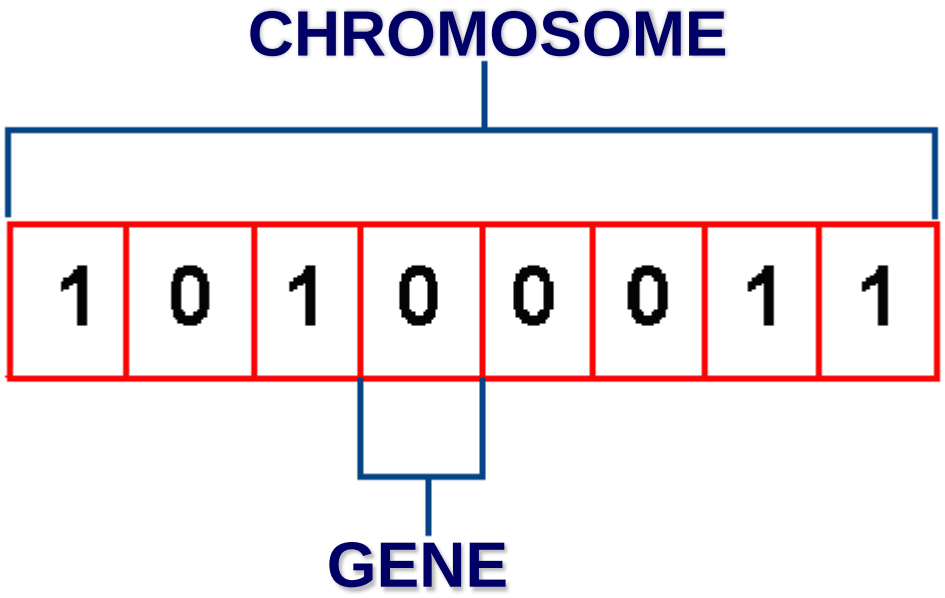
(1) Encoding

- 通常将解对应的参数（基因）连接（串联）形成一个字符串（染色体）
- 所有种类的字母表都可以用于染色体（数字、字符），但通常使用**二进制串**表示
- Chromosomes could be:
  - Bit strings (0101 ... 1100)
  - Real numbers (43.2 -33.1 ... 0.0 89.2)
  - Permutations of element (E11 E3 E7 ... E1 E15)
  - Lists of rules (R1 R2 R3 ... R22 R23)
  - Program elements (genetic programming)
  - ... any data structure ...

(1) Encoding

Example: Discrete Representation (Binary alphabet)

- 表示可以使用离散值（二进制、整数或具有离散值集的任何其他系统）.
- 比如，用以下二进串表示一个个体.



Example: Discrete Representation (Binary alphabet)

8 bits Genotype

Phenotype:

- Integer
- Real Number
- Schedule
- ...
- Anything?

Example: Discrete Representation (Binary alphabet)

Phenotype could be integer numbers

Genotype:

1	0	1	0	0	0	1	1

Phenotype:

= 163

$$1*2^7 + 0*2^6 + 1*2^5 + 0*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0 =$$

$$128 + 32 + 2 + 1 = 163$$

Example: Discrete Representation (Binary alphabet)

Phenotype could be Real Numbers

e.g. a number between 2.5 and 20.5 using 8 binary digits

Genotype: **Phenotype:**

1	0	1	0	0	0	1	1

= 13.9609

$2^8=256$

$$x = 2.5 + \frac{163}{255}(20.5 - 2.5) = 13.9609$$

Binary Encoding

假设某一参数的取值范围是$[u_{min}, u_{max}]$，我们用长度为l的二进制编码符号串来表示该参数，则它总共能够产生2^l种不同的编码，参数编码时的对应关系如下：

00000000…00000000 = 0	u_{min}
00000000…00000001 = 1	$u_{min} + \delta$
00000000…00000010 = 2	$u_{min} + 2\delta$
……	
11111111…11111111=2^l–1	u_{max}

其中，δ为二进制编码的编码精度，其公式为：

$$\delta = \frac{u_{max} - u_{min}}{2^l - 1}$$

Binary Decoding

假设某一个体的编码是：

$$x: \quad b_l b_{l-1} b_{l-2} \ldots\ldots b_2 b_1$$

则对应的解码公式为：

$$x = u_{min} + \left(\sum_{i=l}^{1} b_i \cdot 2^{i-1} \right) \cdot \frac{u_{max} - u_{min}}{2^l - 1}$$

(1) Encoding

Example 设 -3.0 ≤x≤12.1 ，精度要求δ=1/10000，由公式：

$$\delta = \frac{u_{max} - u_{min}}{2^l - 1}$$

得：

$$2^l = \frac{U_{max} - u_{min}}{\delta} + 1 = \frac{12.1 + 3.0}{1/10000} + 1$$

$$= 151001$$

即：$2^{17} < 151001 < 2^{18}$

*x*需要18位{0/1} 符号表示。 如：0100010010110100 00

解码：

$$x = u_{min} + \left(\sum_{i=l}^{1} b_i \cdot 2^{i-1}\right) \cdot \frac{U_{max} - u_{min}}{2^l - 1}$$

$$= -0.3 + 70352 \times (12.1+3)/(2^{18}-1)$$

$$= 1.052426$$

(1) Encoding

Binary Code Advantages:

1) Simple and easy

2) Convenient to analyze by schema

Binary Code Disadvantages: 1) 相邻整数的二进制编码可能具有Long Hamming distance，将降低遗传算子的搜索效率。

2 缺乏精度可调功能。

3 对于高维优化问题，会导致很长的串长。

Example: Discrete Representation (Binary alphabet)

Phenotype could be a Schedule

e.g. **8 jobs**, **2 time steps**

Genotype:

1	0	1	0	0	0	1	1

=

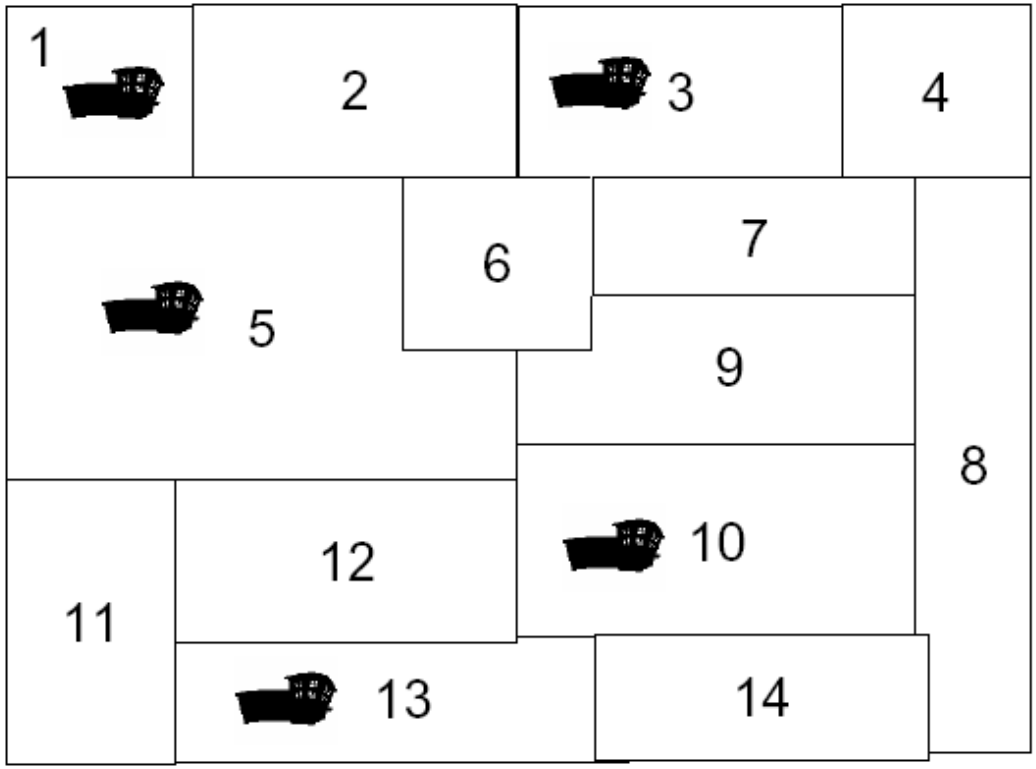
Job	Time Step
1	2
2	1
3	2
4	1
5	1
6	1
7	2
8	2

Phenotype

(1) Encoding

Example: Discrete Representation (Binary alphabet)

- A representation for the fire station location problem

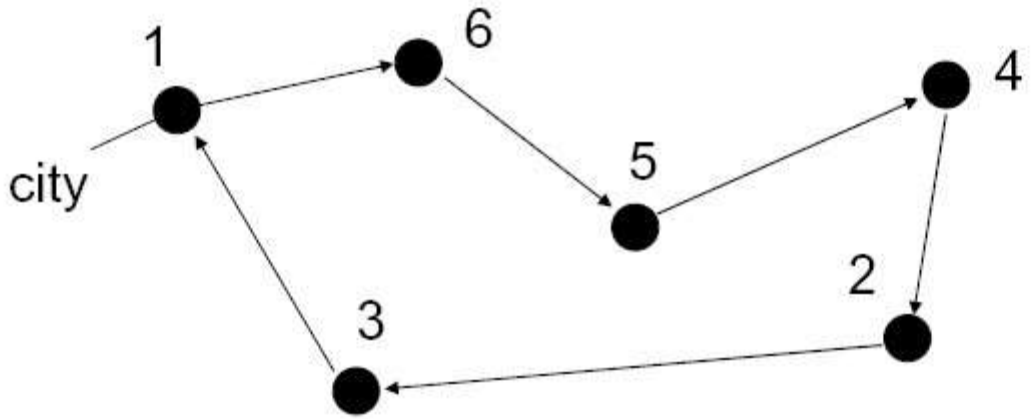


1 0 1 0 1 0 0 0 0 1 0 0 1 0

"1" represents a fire station

(1) Encoding

- Example: Order based representation
  - Traveling Salesman Problem
  - Two representations of the TSP



The arcs

1 2 3 4 5 6
6 3 1 2 4 5

The ordering

1 6 5 4 2 3

Same problem, but two different chromosome representations

(2) Fitness Function---Evaluating an Individual

- Each chromosome has a "fitness"
- The objective function (value) is usually mapped into the fitness of each individual
- For the TSP the fitness is usually the cost of the tour (time, distance, price)
- Choosing the right fitness function is very important, but also quite difficult

(2) Fitness Function

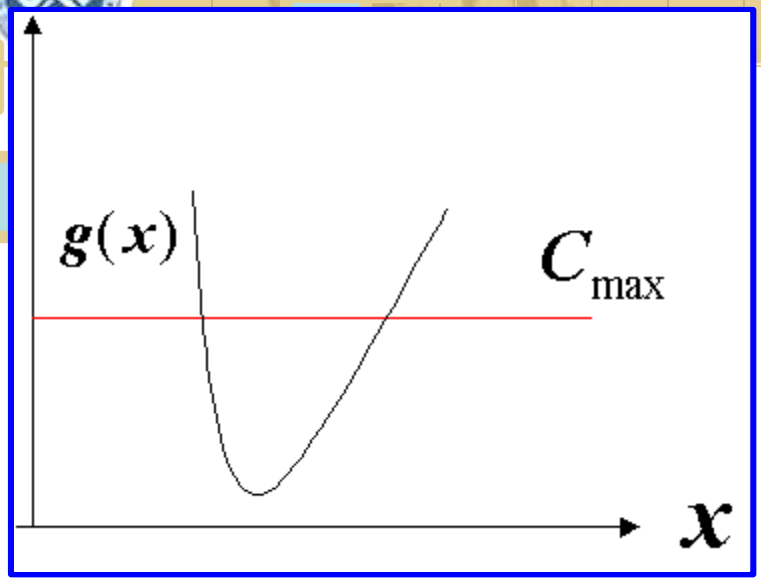
1) 适应度函数值必须非负。根据情况做适当的处理

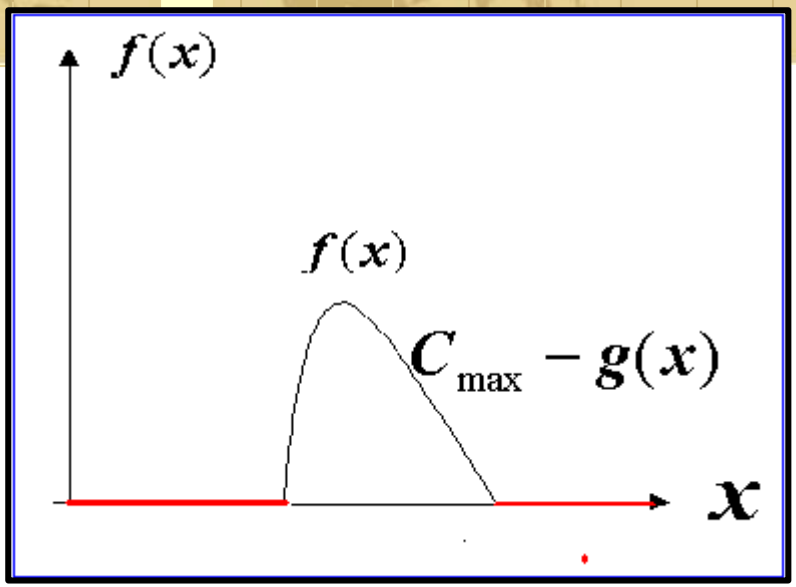
优化求极小$g(x)$时

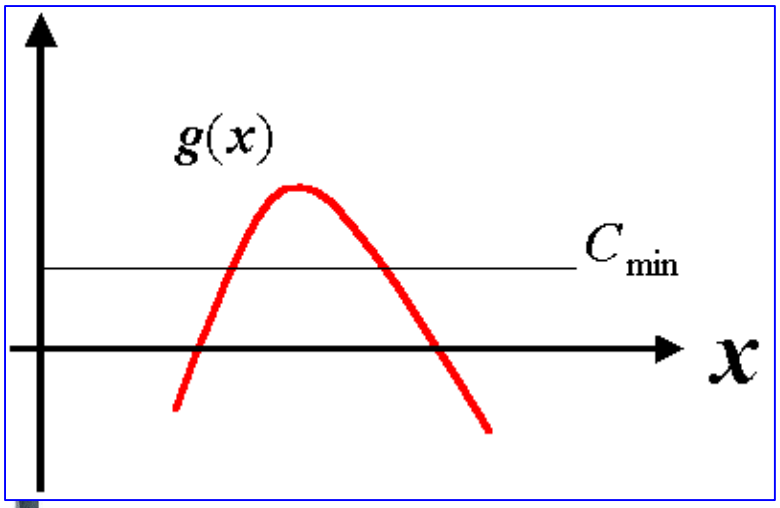
$$f(x)=\begin{cases} C_{\max}-g(x) & \text{当 } g(x)<C_{\max}; \\ 0 & \text{其它情况。} \end{cases}$$

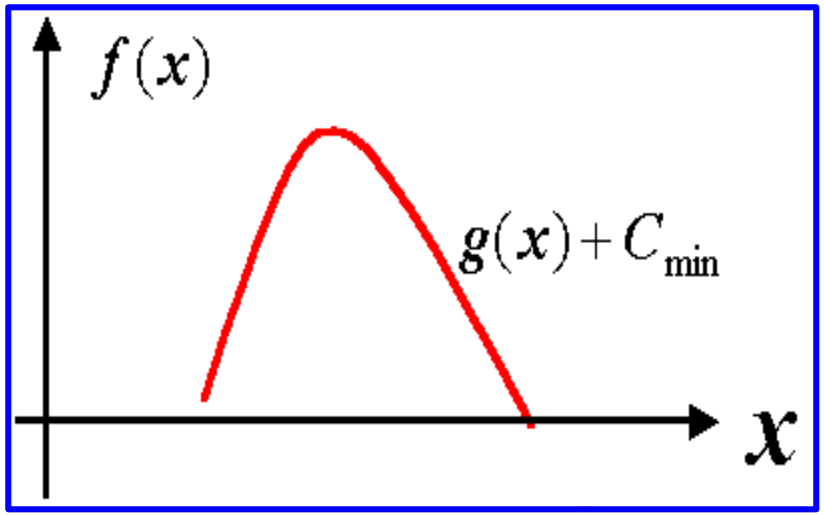
目标求极大，有负值：

$$f(x)=\begin{cases} u(x)+C_{\min} & \text{当 } u(x)+C_{\min}>0; \\ 0 & \text{其它情况。} \end{cases}$$


g(x)
C max
x


f(x)
f(x)
C max − g(x)
x


g(x)
C min
x


f(x)
g(x)+C min
x

2）适应度函数的设计对遗传算法的影响

- 适应度函数影响遗传算法的迭代停止条件
  - 发现满足最大值或次优解
  - 发现群体个体进化已趋于稳定状态，即占群体一定比例的个体已完全是同一个体
- 适应度函数与问题约束条件
  - 将带约束优化问题转换成一个附带考虑代价或惩罚的非约束优化问题。

(3) Selection (Reproduction)

We want to have some way to ensure that better individuals have a better chance of being parents than less good individuals.

Selection Operator 选择算子

- 适应度比例方法（目前最基础、最常用）
  - Roulette Wheel 轮盘法
- 最佳个体保存方法
  - 思想：把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，前提是下一代不存在该个体
  - 优点：进化过程中某一代的最优解可不被交叉或变异操作破坏
  - 缺点：局部最优个体的遗传基因会急速增加而使进化有可能陷于局部解。

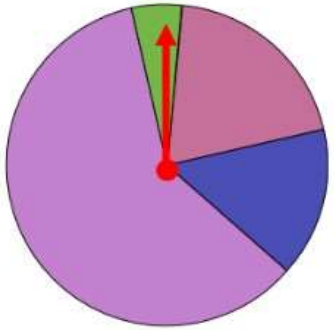
- 期望值方法
  - 基于轮盘法的改进。计算每个个体的下一代生存的期望数目，若参与交叉则期望数目减0.5，若不参与则减1，当期望值小于0则该个体不参与选择。
- 排序选择方法
  - 思想：根据计算的适应度大小在群体中对个体排序，然后把事先设计好的概率表按序分配给个体，作为各自的选择概率
- 联赛选择方法
  - 类似体育比赛制度，从群体中任选一定数目的个体（称为联赛规模），其中适应度最高的个体保存到下一代，反复执行这一过程直到保存到下一代的个体数达到预先设定的数目为止。

选择方法——适应度比例法（轮盘法）

按各染色体适应度大小比例来决定其被选择数目的多少。

某染色体被选的概率：P_c

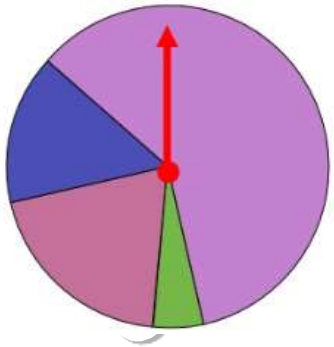
$$P_c = f(x_i) \Big/ \sum f(x_i)$$



x_i 为种群中第i个染色体，

$f(x_i)$ 第i个染色体的适应度值

$\sum f(x_i)$ 种群中所有染色体适应度值之和。



具体步骤

1 计算各染色体适应度值

2 累计所有染色体适应度值，记录中间累加值S - mid 和最后累加值sum = $\sum f(x_i)$

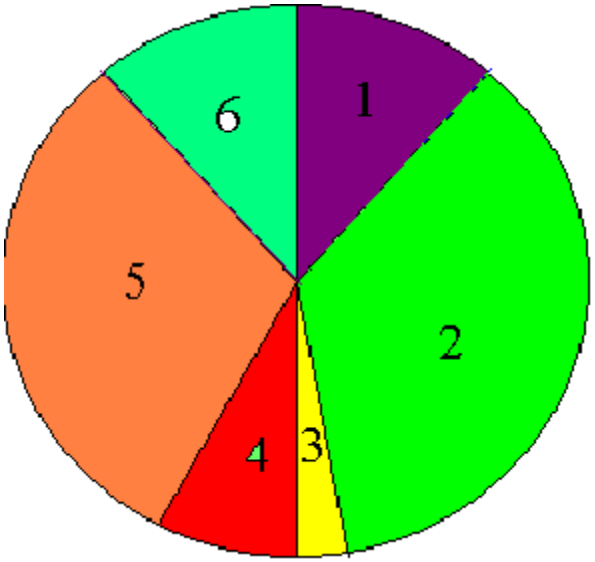
3 产生一个随机数N，0〈N 〈sum

4 选择对应中间累加值S - mid 的第一个染色体进入交换集

5 重复（3）和（4），直到获得足够的染色体。

举例：

1 具有6个染色体的二进制编码、适应度值、P_c累计值。



序号	染色体	适应度值	所占比例	累计
1	01110	8	16	8
2	11000	15	30	23
3	00100	2	4	25
4	10010	5	10	30
5	01100	12	24	42
6	00011	8	16	50

用转轮方法进行选择

染色体的适应度和所占的比例

2. 10个染色体种群按比例的选择过程

染色体被选的概率

染色体编号	1	2	3	4	5	6	7	8	9	10
适应度	8	2	17	7	2	12	11	7	3	7
被选概率	0.1	0.02	0.22	0.09	0.02	0.16	0.14	0.09	0.03	0.09
适应度累计	8	10	27	34	36	48	59	66	69	76

被选的染色体个数

随机数	23	49	76	13	1	27	57
所选染色体号码	3	7	10	3	1	3	7

- Roulette wheel disadvantages:
  - Danger of premature convergence because outstanding individuals take over the entire population very quickly
  - Low selection pressure when fitness values are near each other
  - Behaves differently on transposed versions of the same function

(4) Crossover

- 选择复制不能创新,交叉解决染色体的创新
  - Two parents produce two offspring
  - There is a chance **that** the chromosomes of the two parents are copied unmodified as offspring
  - There is a chance **that** the chromosomes of the two parents are randomly recombined (crossover) to form offspring
  - Generally the chance of crossover is between 0.6 and 1.0

Crossover operator

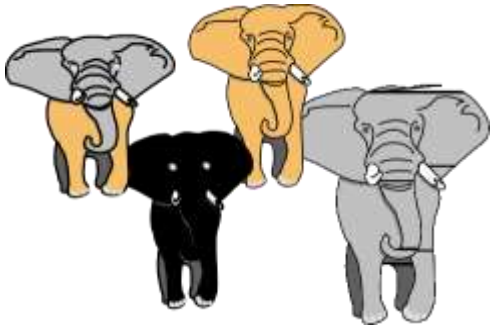
- Single point crossover (Simple crossover)
  - 个体中随机设定交叉点，该点前或后的两个个体的部分结构进行互换，生成两个新的个体。 Example:

A：10110111 | 00 → Single point crossover → A’：10110111 11

B：00011100 | 11 → B’：00011100 00

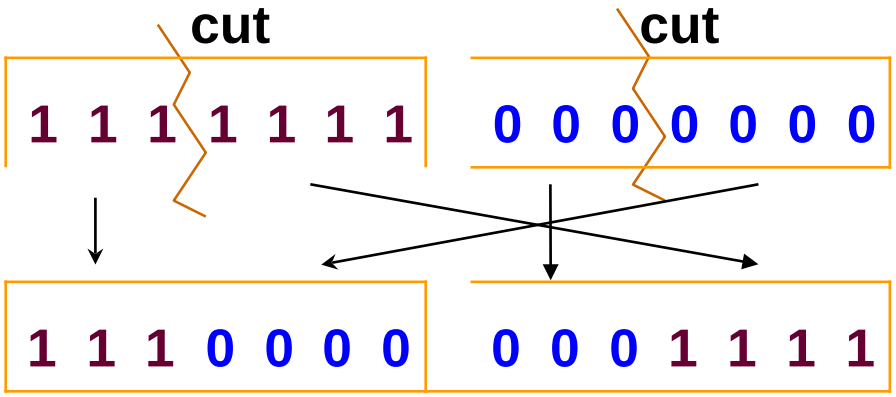
Example: Crossover for Discrete Representation

Whole Population:

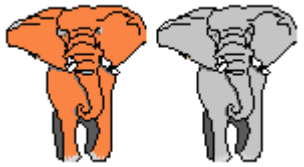


. . .

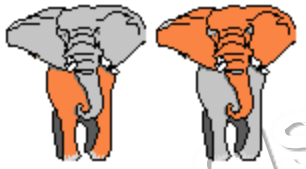
Each chromosome is cut into *n* pieces randomly. (Example for *n*=1,one point crossover, random position is 3)



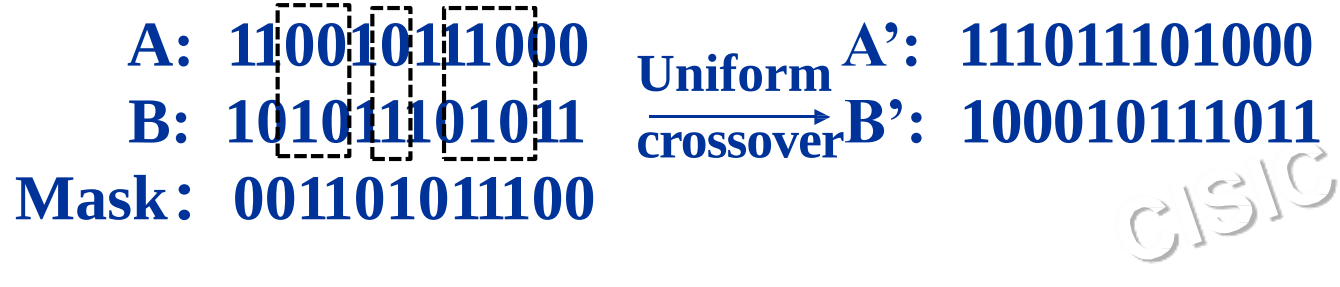
parents



offspring



- Two point crossover
- Multi point crossover
- Uniform crossover
  - A random mask is generated
  - The mask determines which bits are copied from one parent and which from the other parent
  - Example:

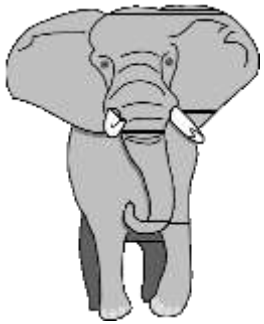


(4) Mutation

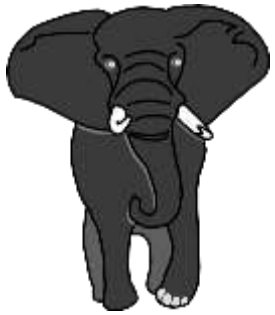
- There is a chance that a gene of a child is changed randomly
- Generally the chance of mutation is low (e.g. 0.001)
- In binary code,1->0 or 0->1.
- Generating new offspring from single parent
- Maintaining the diversity of the individuals
  - Crossover can only explore the combinations of the current gene pool
  - Mutation can "generate" new genes
- Mutation Operator
  - 基本变异算子、均匀变异、非一致变异、自适应变异

Example: Mutation for Discrete Representation

before 1 1 1 1 1 1 1



after 1 1 1 0 1 1 1



mutated gene

Mutation usually happens with probability p_m for each gene

变异概率

$$P_m = \frac{B}{M \cdot \lambda}$$

式中 B——每代中变异的基因数目；

M——每代中群体拥有的个体数目

λ——个体中基因串长度。

[变异操作示例]

变异字符的位置是随机确定的，如下表所示。某群体有3个个体，每个体含4个基因。针对每个个体的每个基因产生一个[0, 1] 区间具有3位有效数字的均匀随机数。假设变异概率$p_m = 0.01$，则随机数小于0.01的对应基因值产生变异。表中3号个体的第4位的随机数为0.001，小于0.01，该基因产生变异，使3号个体由0010 变为0011 。其余基因的随机数均大于0.01，不产生变异。

个体编号	10 个体	随机数				新个体
1	1 0 1 0	0.801	0.102	0.266	0.373	
2	1 1 0 0	0.120	0.796	0.105	0.840	
3	0 0 1 0	0.760	0.473	0.894	0.001	0 0 1 1

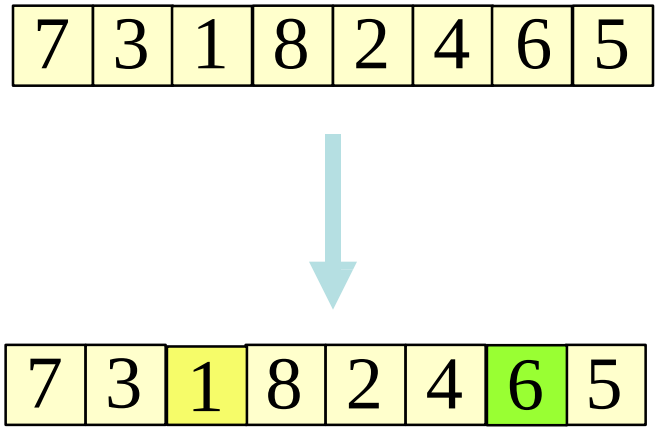
Example: Mutation for real valued representation

- Mutate values by adding some random noise
- Often, a Gaussian/normal distribution $N(0,\sigma)$ is used, where
  - 0 is the mean value
  - σ is the standard deviation
- let

$$x'_i = x_i + N(0,\sigma)$$

Example: Mutation for order based representation (Swap)

Randomly select two different genes and swap them.



标准遗传算法（GA）的基本参数

①种群规模P: 参与进化的染色体总数，一般取20～100

②终止代数T: 遗传运算的终止进化代数，一般取为100～500

④交换率: P_c 一般为0.4~0.99

⑤变异率: P_m 一般为0.0001~0.1

[说明]

这4个运行参数对遗传算法的求解结果和求解效率都有一定的影响，但目前尚无合理选择它们的理论依据。在遗传算法的实际应用中，往往需要经过多次试算后才能确定出这些参数合理的取值大小或取值范围。

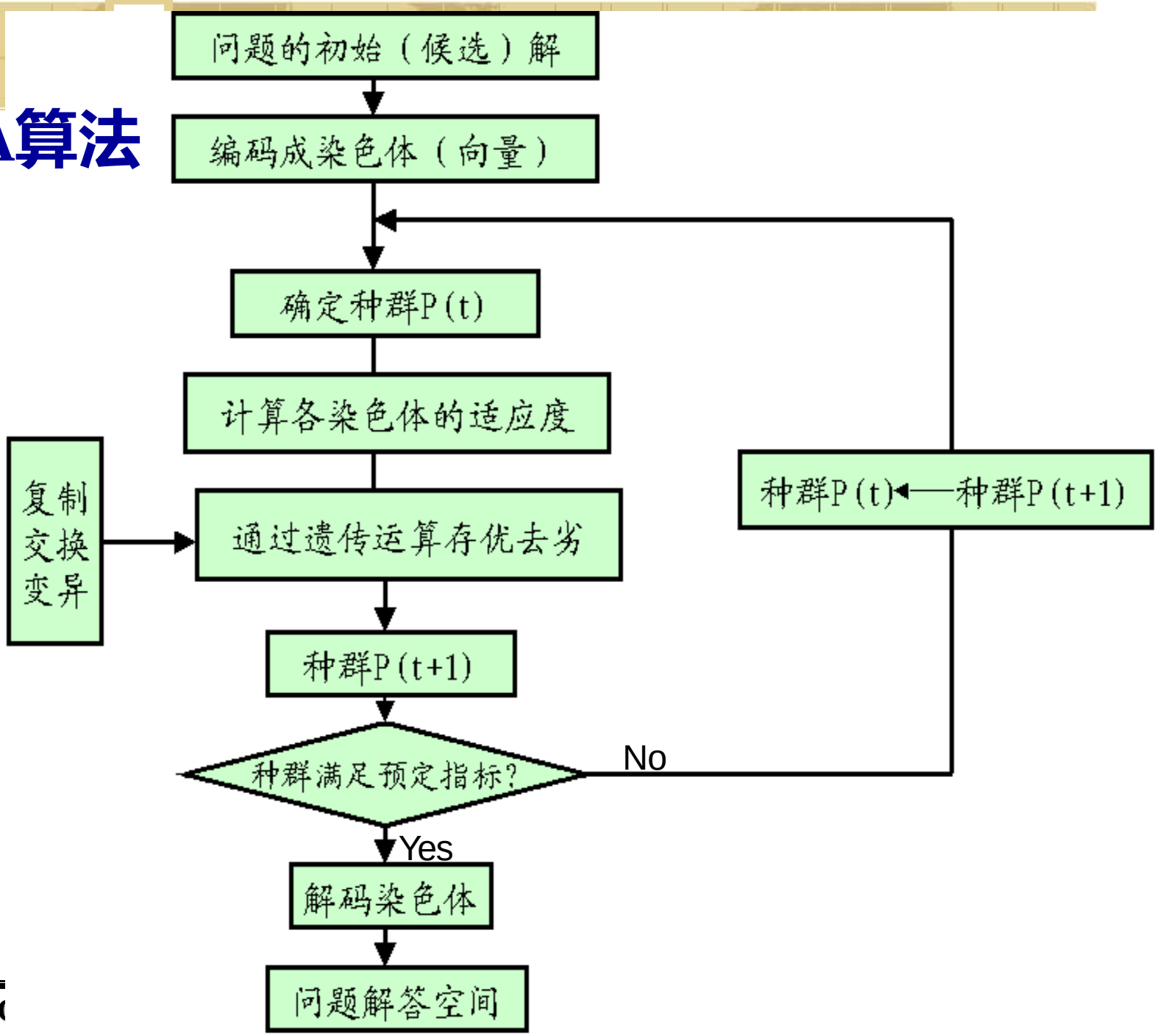
基本遗传算法的形式化定义

基本遗传算法可定义为一个7元组：

$$GA = (M, F, s, c, m, p_c, p_m)$$

M——Population size；F——Individual fitness function；s——Selection operator；c——Crossover operator：m——Mutation operator；p_c——Crossover probability；p_m——Mutation probability；

标准GA算法


问题的初始（候选）解
编码成染色体（向量）
确定种群P(t)
计算各染色体的适应度
复制
交换
变异
通过遗传运算存优去劣
种群P(t+1)
种群满足预定指标?
No
Yes
种群P(t)←种群P(t+1)
解码染色体
问题解答空间

举例: 设函数 $f(x)=x^2$, 求其在区间 $[0,31]$ 的最大值。

初始种群和它的适应度值 **变异概率取0.001**

编号	初始种群（随机）	x 值	适应度 $f(x)=x^2$	选择概率 $f_i/\sum f_i$	$f_i/\bar{f}_i$	实际选择数目
1	01101	13	169	0.14	0.58	1
2	11000	24	576	0.49	1.97	2
3	01000	8	64	0.06	0.22	0
4	10011	19	361	0.31	1.23	1
和			1170	1.00	4.00	4
平均			293	0.25	1.00	1
最大			576	0.49	1.97	2

每次选择都对个体进行一次复制。

染色体的交换操作

复制后交换种群集	交换配对(随机选择)	交换位置(随机选择)	新种群	x 值(无符号整数)	$f(x)=x^2$
0110|1	2	4	01100	12	144
1100|0	1	4	11001	25	625
11|000	4	2	11011	27	729
10|011	3	2	10000	16	256
和					1754
平均					439
最大					729

由于群体中共有20×0.001=0.02位基因可以变异，这意味着群体中通常没有一位基因可变异。

基本遗传算法应用举例

—— 基本遗传算法在函数优化中的应用

[例] Rosenbrock函数的全局最大值计算。

$$\max \quad f(x_1,x_2) = 100\,(x_1^2 - x_2)^2 + (1-x_1)^2$$

$$\text{s.t.} \quad -2.048 \le x_i \le 2.048 \qquad (x_i=1,2)$$

如图所示：

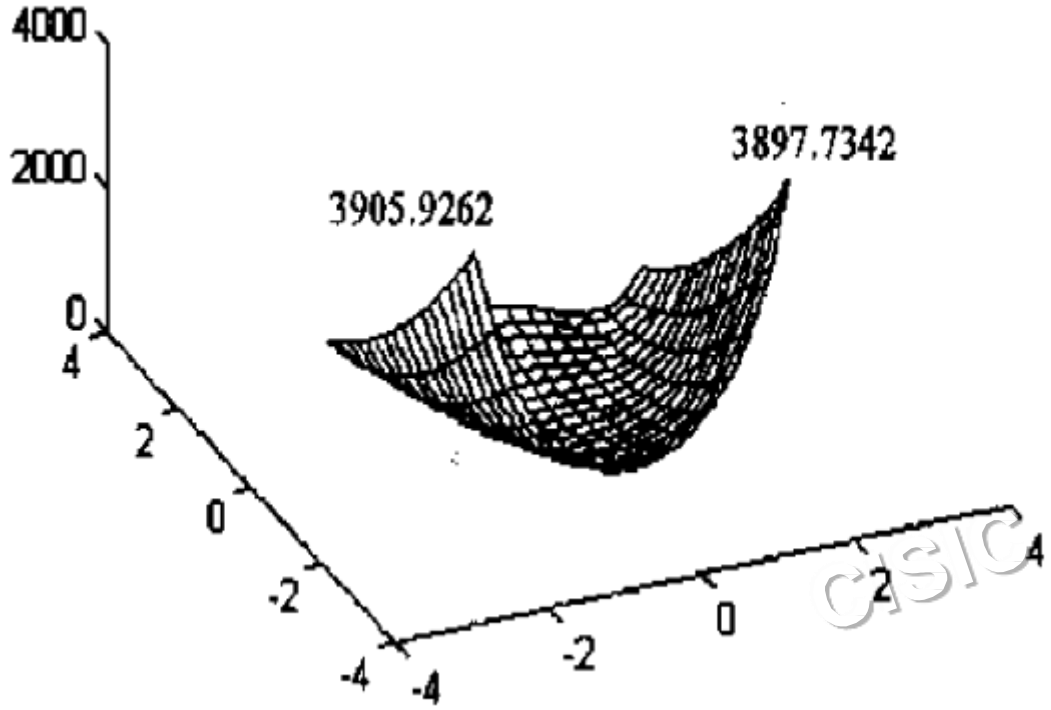
该函数有两个局部极大点，分别是：

f(2.048, -2048)=3897.7342

和

f(-2.048,-2.048)=3905.9262

其中后者为全局最大点。



下面介绍求解该问题的遗传算法的构造过程：

第一步：确定决策变量及其约束条件。

$$\text{s.t.} \quad -2.048 \le x_i \le 2.048 \qquad (x_i=1,2)$$

第二步：建立优化模型。

$$\max \ f(x_1,x_2) = 100\,(x_1^2-x_2)_2 + (1-x_1)_2$$

第三步：确定编码方法。

用长度为10位的二进制编码串来分别表示二个决策变量x_1,x_2。10位二进制编码串可以表示从0到1023之间的1024个不同的数，故将x_1,x_2的定义域离散化为1023个均等的区域，包括两个端点在内共有1024个不同的离散点。从离散点-2.048到离散点2.048，依次让它们分别对应于从0000000000(0)到1111111111(1023)之间的二进制编码。再将分别表示x_1和x_2的二个10位长的二进制编码串连接在一起，组成一个20位长的二进制编码串，它就构成了这个函数优化问题的染色体编码方法。例如

X：0000110111 1101110001　　就表示一个个体的基因型。

第四步：确定解码方法。

解码时先将20位长的二进制编码串切断为二个10位长的二进制编码串，然后分别将它们转换为对应的十进制整数代码，分别记为y_1和y_2。

依据前述个体编码方法相对定义域的离散化方法可知，将代码y_i转换为变量x_i的解码公式为：

$$x_i = 4.096 \times \frac{y_i}{1023} - 2.048 \qquad (i = 1,2)$$

例如，对前述个体

X：0000110111 11011 10001

它由这样的两个代码所组成：

$y_1 = 55$

$y_2 = 881$

经上式的解码处理后，得到：

$x_1 = -1.828$

$x_2 = 1.476$

第五步：确定个体评价方法

由式 $f(x_1,x_2) = 100\ (x_1^2-x_2)^2 + (1-x_1)^2$ 可知，Rosenbrock函数的值域总是非负的，并且优化目标是求函数的最大值，故这里可将个体的适应度直接取为对应的目标函数值，并且不再对它作其他变换处理，即有：$F(x) = f(x_1,x_2)$

第六步：设计遗传算子

选择运算使用比例选择算子；

交叉运算使用单点交叉算子；

变异运算使用基本位变异算子。

第七步：确定遗传算法的运行参数

对于本例，设定基本遗传算法的运行参数如下：

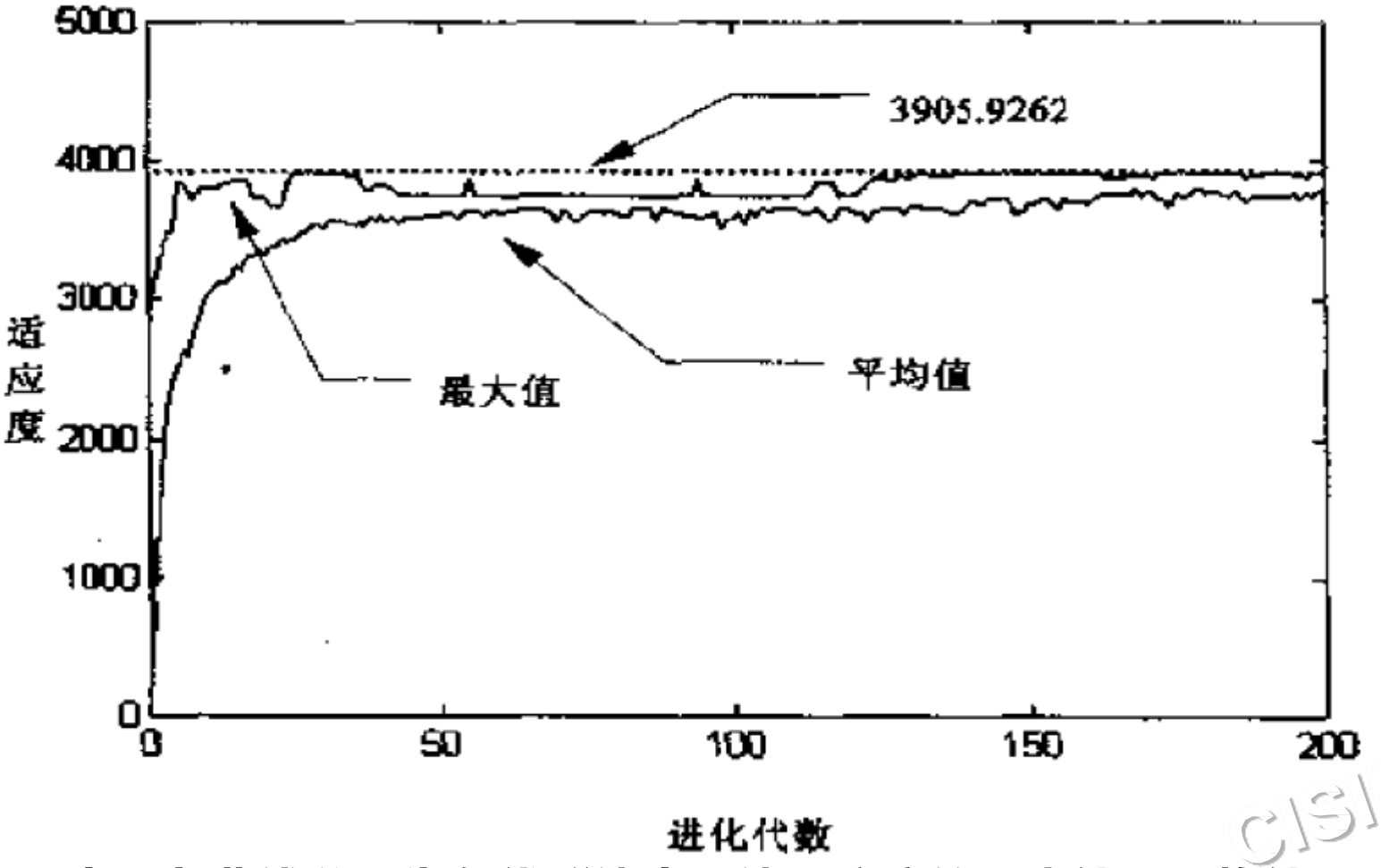
群体大小：$M = 80$

终止代数：$T = 200$

交叉概率：$p_c = 0.6$

变异概率：$p_m = 0.001$

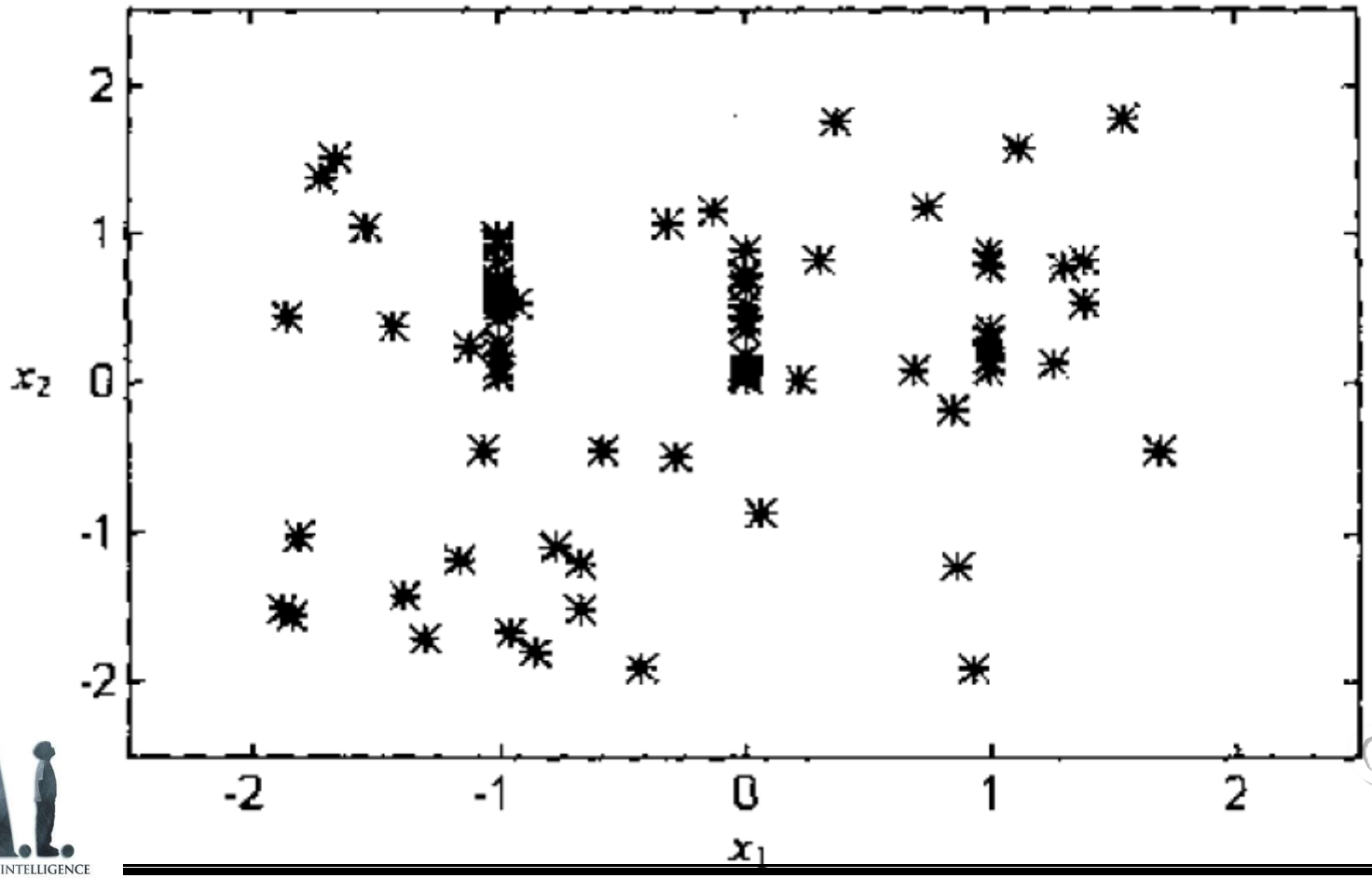
下图为其进化过程示例及运行结果。



图中两条曲线分别为各代群体中个体适应度的最大值和平均值。

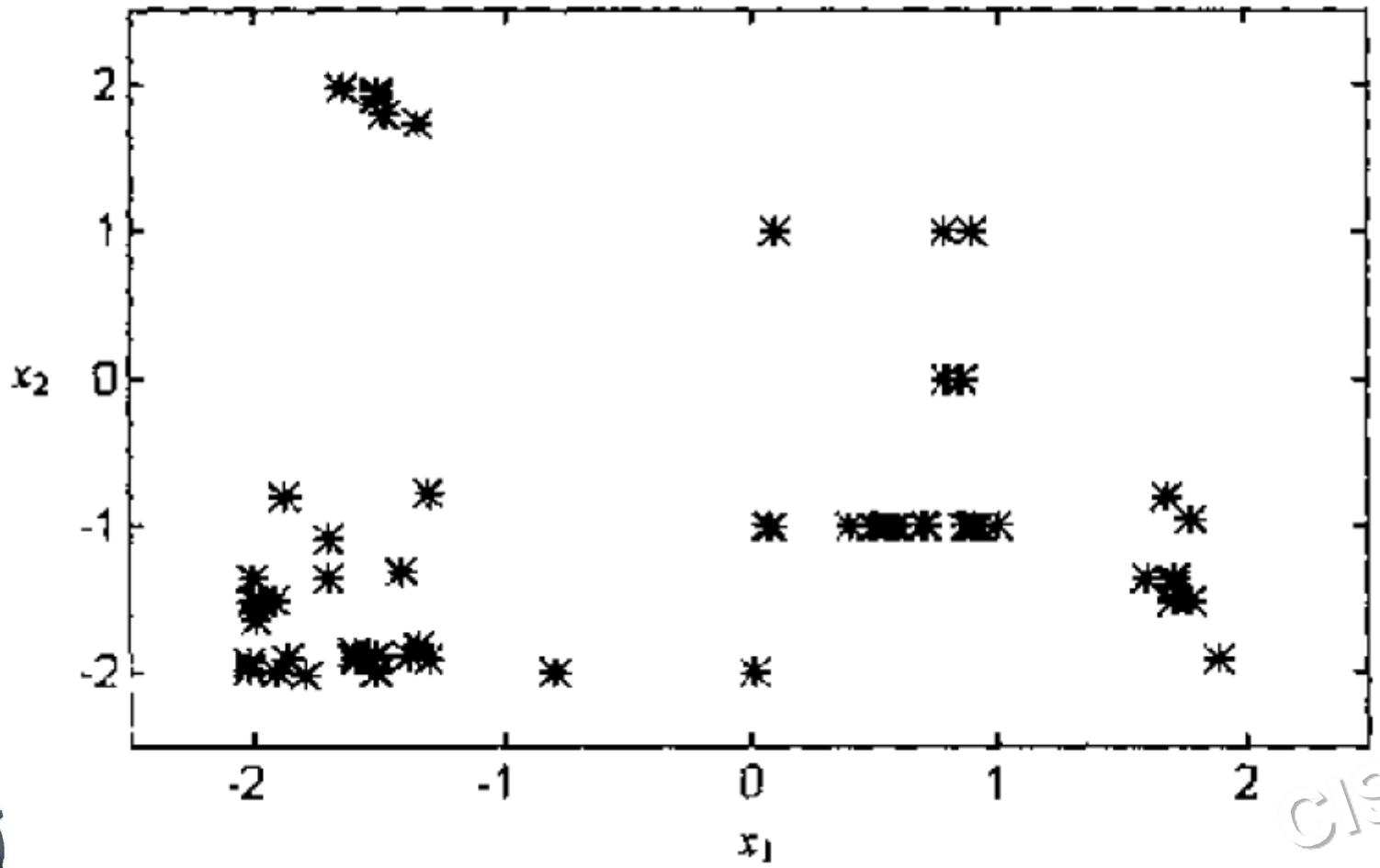
下图所示分别为初始群体、第5代群体、第10代群体和第100代群体中个体的分布情况。

在图(a)中各个个体分布得比较均匀。



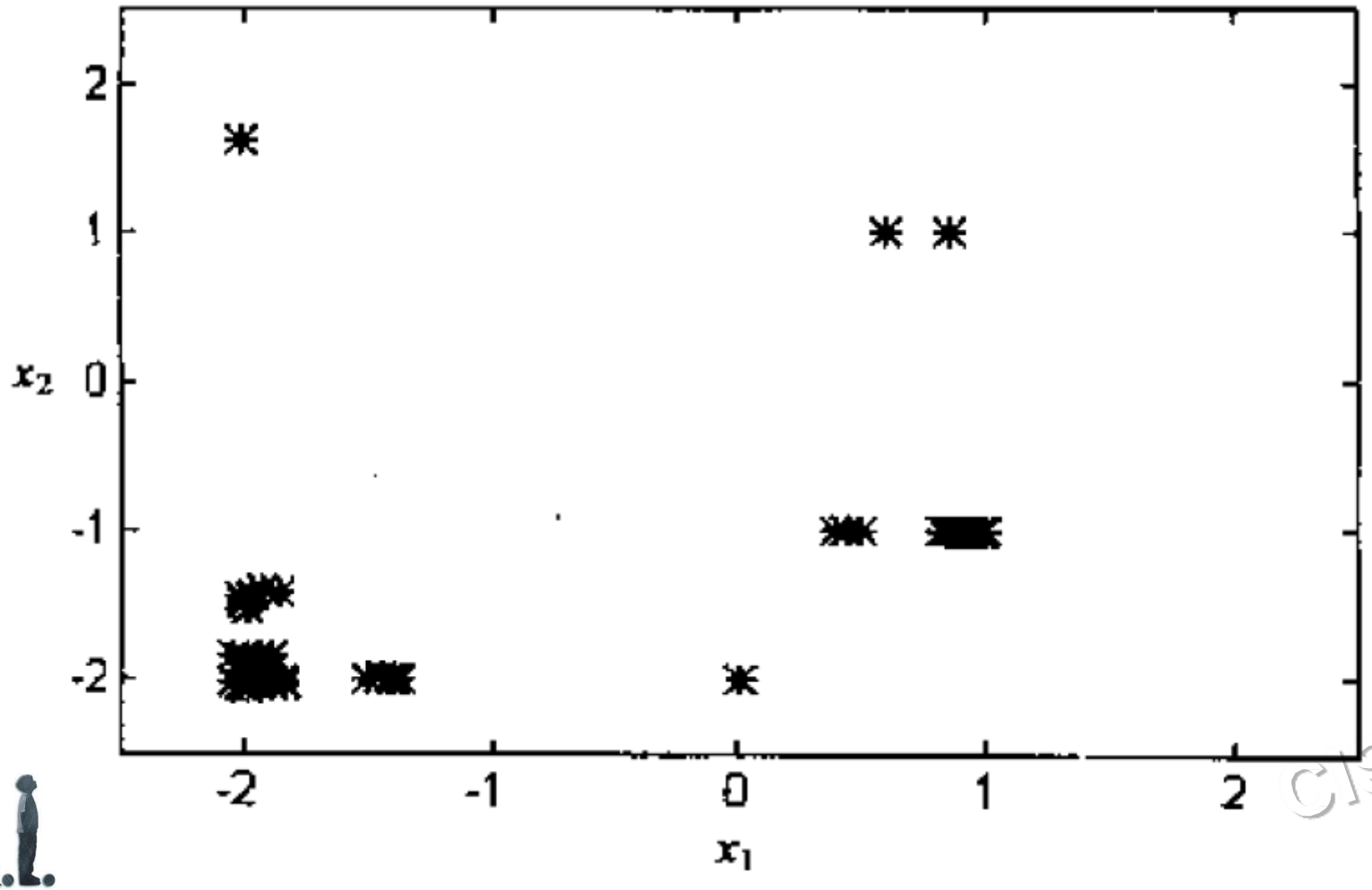
(a)

在图(b)中大量的个体分布在最优点和次最优点附近。



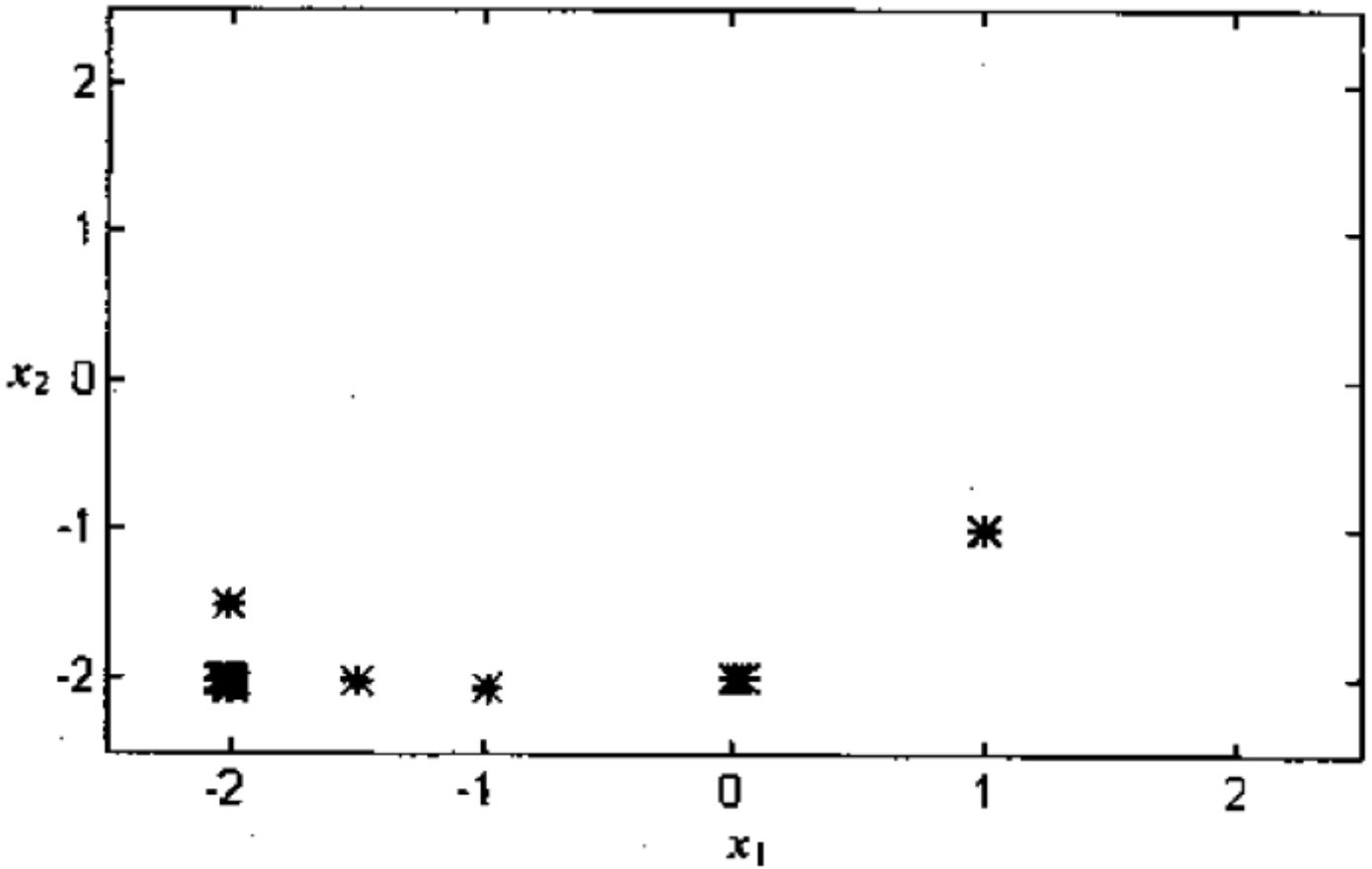
(b)

从图(c) 中可以看出，次最优点也被淘汰。



(c)

从图(d)中可以看出，个体更加集中在最优点附近。



(d)

由该组图我们可以看出，随着进化过程的进行，群体中适应度较低的一些个体被逐渐淘汰掉，而适应度较高的一些个体会越来越多，并且它们都集中在所求问题的最优点附近，从而最终就可搜索到问题的最优解。

GAs Advantages

- 特点：
  - 通用
  - 鲁棒
  - 次优解、满意解
- 遗传算法能解决的问题：
  - 优化
  - NP难
  - 高度复杂的非线性问题