

# 人工智能技术

## Artificial Intelligence

——人工智能: 经典智能+计算智能+机器学习

AI: Classical Intelligence + Computing Intelligence + Machine Learning

王鸿鹏、王润花、韩明静、许丽

南开大学人工智能学院



# 第二部分 计算智能 Computational Intelligence

## ——第八章：进化计算 Chapter 8: Evolutionary Computation

# 计算

**计算**，数学用语，是一种将单一或复数之输入值转换为单一或复数之结果的一种思考过程。

## 计算关系

在数学计算中，一个计算式包括数据，计算符或算子以及计算结果。因此数学计算中的关系是计算原理中必须阐明的理论基础。

## 数据与数据的关系

数据在一个计算式中，则称数据存在计算关系。有些计算关系由数据的内在性质（例如系数矩阵，级数中的具体项，合式公式中的项），物理位置（一幅图像中数据的显示或表示，直角坐标系中曲线的关系，cpu阵列，数据的存储）决定。

## 数据与运算符的关系

- 1) 自然数据的表示。例如求一个曲面梯形的面积。
- 2) 人工数据的处理（例如 程序中的数据）。
- 3) 自然数据的人工处理。例如：放大一幅图像的一部分。

# 计算

## 运算符与运算符的关系

1) 整体与元素的关系.集合数据例如矩阵, 从矩阵加到元素加, 实现对集合元素的处理. 相同运算符对不同数据产生的计算效果可不同 (例如C++语言的重载, 多态等)。

2) 高阶的运算符, 常常是低阶运算符的组合, 再使用一个新出现的计算符, 构成一个序列. 例如积分: 级数的极限计算.使复杂的数据元计算能够实现.

### 3) 低阶运算与新运算的发现

对新形式数据的新计算, 常常用到如何组合低级运算符, 构建一个新的高阶运算符.因此计算并不是化简这一个过程.有些同学认为计算就是越来越简单, 因此对数学失去了兴趣.实际上, 还存在一个可逆的过程, 即如何用低阶的, 离散的运算符, 处理复杂的数据结构以及庞大的计算量, 也是一个很有趣的问题.

### 4) 相同的计算, 常常有不同的性质.

例如.线性代数中, 同样是三矩阵乘法, 在相似变换中只要求可逆矩阵, 而在二此型的标准型变换中, 则要求正交矩阵.计算是对特定数据元的计算,因此数据元的性质对运算符的选择, 计算的实现有决定性作用.计算表达式常常有不同的形式.代数式, 方程, 函数, 行列式, 微积分或者数理统计计算式等等, 实现对不同数据的具体计算.

# 计算

## “计算”的应用

计算与人类：由于现代人类各个课题学科繁多，涉及面广，而分类又细。而当今的每个学科都需要进行大量的计算。

- 天文学研究组织需要计算机来分析太空脉冲（pulse），星位移动；
- 生物学家需要计算机来模拟蛋白质的折叠（protein folding）过程，发现基因组的奥秘；
- 药物学家想要研制治愈癌症或各类细菌与病毒的药物，医学家正在研制防止衰老的新办法；
- 数学家想计算最大的质数和圆周率的更精确值；
- 经济学家要用计算机分析计算在几万种因素考虑下某个企业/城市/国家的发展方向从而宏观调控；
- 工业界需要准确计算生产过程中的材料，能源，加工与时间配置的最佳方案。

广义的计算包括数学计算，逻辑推理，文法的产生式，集合论的函数，组合数学的置换，变量代换，图形图像的变换，数理统计等；人工智能解空间的遍历，问题求解，图论的路径问题，网络安全，代数系统理论，上下文表示感知与推理，智能空间等；甚至包括数字系统设计（例如逻辑代数），软件程序设计（文法），机器人设计，建筑设计等设计问题。

# 智能计算

**智能计算(Intelligent Computing)**是一种经验化的计算机思考性程序，是辅助人类去处理各式问题的具有独立思考能力的系统。智能计算属于人工智能（AI）的一个重要分支，是由通用计算发展而来；它既是对通用计算的延续与升华，更是应对人工智能趋势的新计算形态。

计算智能包括遗传算法、进化算法、蚁群算法、免疫算法、启发式算法、粒子群算法、模拟退火算法、麻雀搜索算法、禁忌搜索算法、混合智能算法、神经网络、机器学习、生物计算、模糊逻辑、模式识别、知识发现、数据挖掘等。计算智能具有持续进化、环境友好和开放生态等特征；其实它就是借用自然界规律的启迪根据其原理模仿设计求解问题的智能算法。

# 智能计算

智能算法要解决的一般是最优化问题；而最优化问题就是求解一个可行的甚至是最优的方案的决策问题。最优化问题可以分为：

- (1) 求解一个函数中，使得函数值最小的自变量取值的函数优化问题；
- (2) 在一个解空间里面，寻找最优解，使目标函数值最小的组合优化问题。这些问题都有一个共同的特征就是通过模仿人类智能的某一个(某一些)方面而设计出最优化算法的目的。

各种智能算法在解决全局最优解的问题上有着独到的优点，并且它们有一个共同的特点：都是模拟了事物在自然界的发展过程。如**遗传算法**借鉴了自然界优胜劣汰的进化思想；**神经网络**更是直接模拟了人脑。这些智能算法都有别于一般的按照图灵机进行精确计算的程序，尤其是人工神经网络，是对计算机模型的一种新的诠释，跳出了冯·诺依曼机的圈子，按照这种思想来设计的计算机有着广阔的发展前景。

## 什么是计算智能，它与传统的人工智能有何区别？

第一个对计算智能定义是由贝兹德克(J. C. Bezdek)于1992年提出的。

due to J.C. Bezdek who states that:

"...(strictly) computational systems depend on numerical data supplied by manufactured sensors and do not rely upon knowledge".

Bezdek对一些相关术语给予一定的符号和简要说明或定义。给出了有趣的ABC:

- |                                    |                   |
|------------------------------------|-------------------|
| Ⓐ A- Artificial/Symbolic,          | 表示人工的(非生物的), 即人造的 |
| Ⓑ B- Biological/Organic,           | 表示物理的+化学的+(?)=生物的 |
| Ⓒ C- Computational/Numeric System, | 表示数学+计算机          |

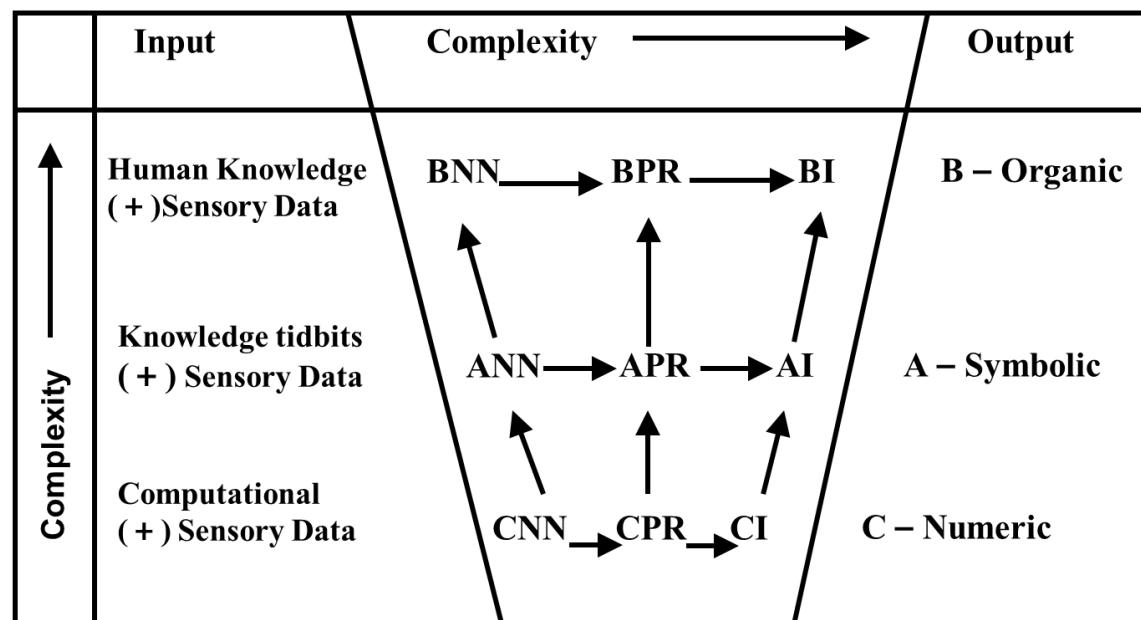


贝兹德克  
(J.C. Bezdek)



## ABC 及其与神经网络(NN), 模式识别(PR)和智能(I)之间的关系

Later, in 1994, Bezdek offers that CI is low-level computation in the style of the mind", whereas AI is mid-level computation in the style of the mind".



Relationships among components of intelligent system (after Bezdek 1994)



表 ABC及相关领域的定义

BNN	人类智能硬件：大脑	人的传感输入的处理
ANN	中层模型：CNN+知识精品	以大脑方式中的中层处理
CNN	低层， 生物激励模型	以大脑方式的传感数据处理
BPR	对人的传感数据结构的搜索	对人的感知环境中结构的识别
APR	中层模型：CPR+知识精品	中层数值和语法处理
CPR	对传感数据结构的搜索	所有CNN+模糊、统计和确定性模型
BI	人类智能软件：智力	人类的认知、记忆和作用
AI	中层模型：CI+知识精品	以大脑方式的中层认知
CI	计算推理的低层算法	以大脑方式的低层认知

“...a system is computational intelligence when it deal only with the numerical (low-level) data, has a pattern recognition component, and does not use knowledge in the AI sense, and additionally when it exhibit:

- ① Computational adaptivity
- ② Computational fault tolerance
- ③ Speed approaching human-like turnaround
- ④ Error rates that approximate human performance

## ➤ Some other opinions:

### ➤ Conference: Computational Intelligence

- Defining "Computational Intelligence" is impossible, to accommodate in a framework established individualities such as fuzzy logic, evolutionary computation, machine learning, Bayesian reasoning, etc.

### ➤ Book: "Computational Intelligence: An Introduction", Andries P. Engelbrecht, Wiley 2002

- Computational intelligence is the study of adaptive mechanisms to enable or facilitate intelligent behavior in complex and changing environments. As such, computational intelligence includes evolutionary computing, swarm intelligence, fuzzy logic, etc.

要定义“计算智能”并不简单。要在一个正式的定义中容纳诸如模糊集合、神经网络、进化计算、机器学习、贝叶斯推理等各自具有其既定特性的不同领域，即便不是不可能，也是非常困难的。

计算智能是关于在复杂和变化的环境中如何实现智能行为的自适应机制研究。因此，计算智能综合了人工神经网络、进化计算、群智能和模糊系统。

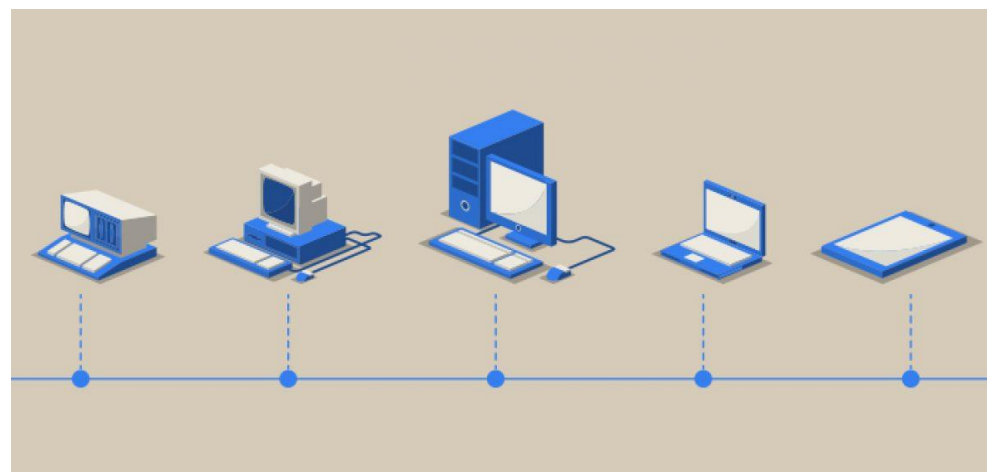


# 进化计算

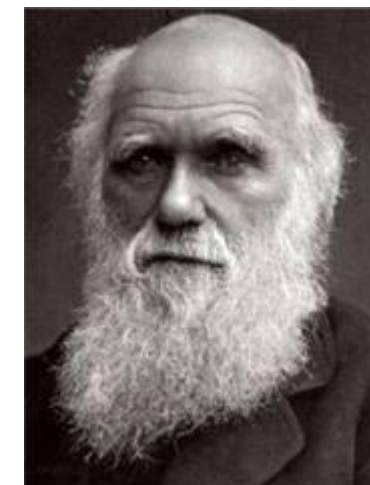
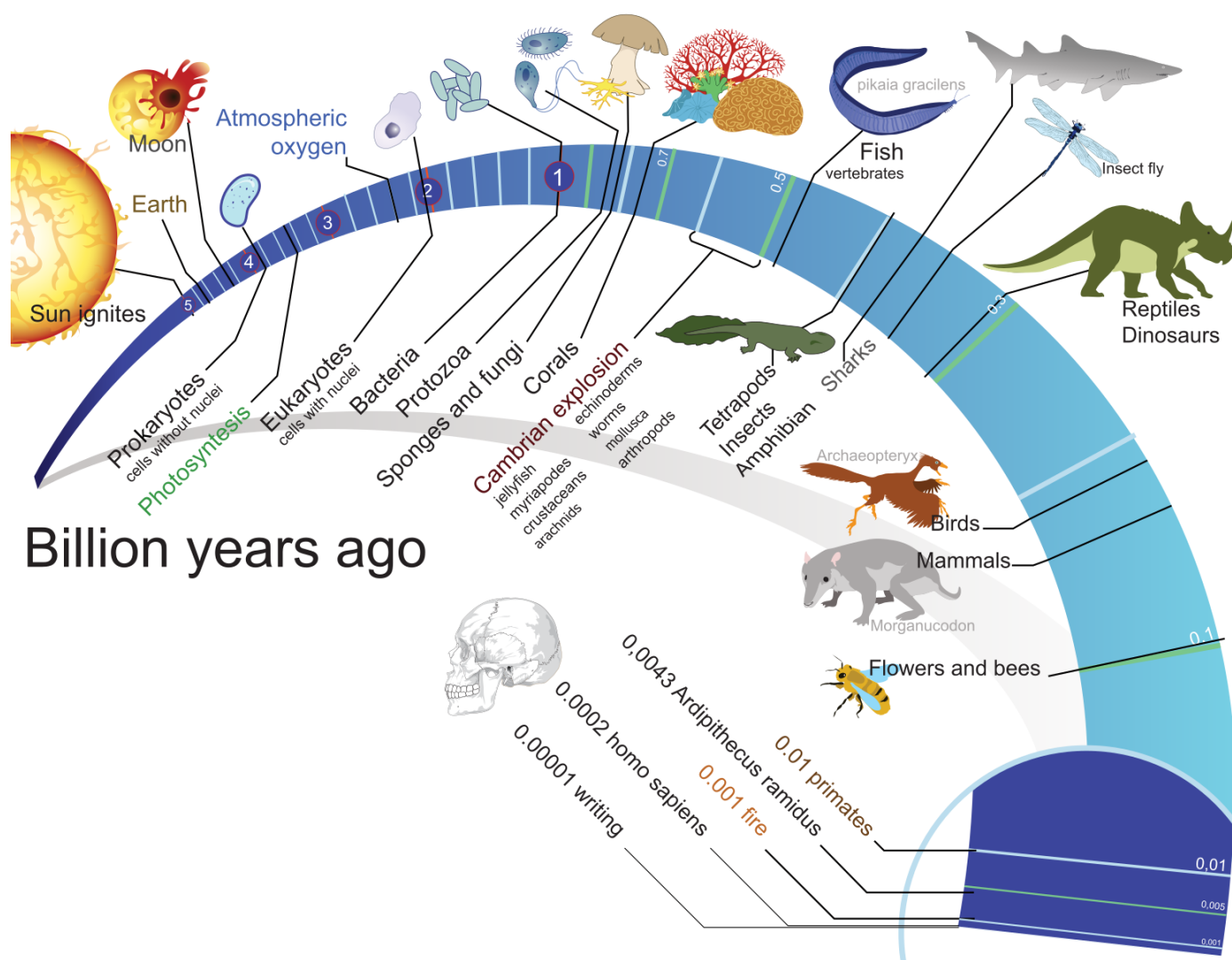
## Evolutionary Computation



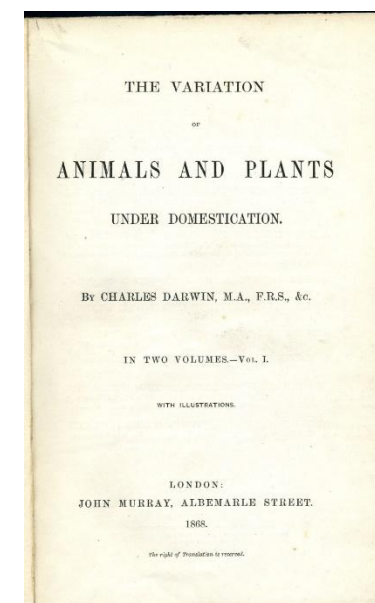
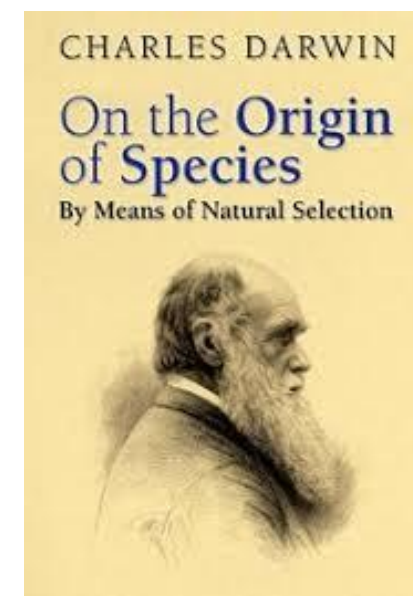
进化，又称演化（evolution），在生物学中是指**种群里的遗传性状在世代之间的变化**。所谓性状是指基因的表现，在繁殖过程中，基因会经复制并传递到子代，基因的突变可使性状改变，进而造成个体之间的遗传变异。新性状又会因物种迁徙或是物种间的水平基因转移，而随着基因在种群中传递。当这些遗传变异受到非随机的自然选择或随机的遗传漂变影响，在种群中变得较为普遍或不再稀有时，就表示发生了进化。简略地说，进化的实质便是：种群基因频率的改变。



# 达尔文——进化论



Charles Darwin





# 进化计算的发展历史

在20世纪40年代，**生物模拟**就成为计算科学的一个组成部分。

- 自动机理论是假设机器是由类似神经元的基本单元组成的；
- 人工神经网络的产生与完善也得益于生物学理论的成果；
- 人工生命的研究与生物学的关系更为密切。

自从“进化”的理论被人们接受后，关于进化的研究得到很快的发展。虽然目前对进化的机制还没有完全搞清楚，但进化的一些特征被人们所认识，生物学的这种研究成果在“工程”优化问题中的应用就是**进化计算**。



# 进化计算的发展历史

**进化计算**的研究始于上个世纪50年代，其思想是将自然界中的进化过程引入到工程研究领域，以解决工程中的优化问题，其中使用了遗传、选择等概念。

- 20世纪60年代，美国密歇根大学J.Holland教授提出了**遗传算法**(Genetic Algorithm, GA)；K.De Jong率先将遗传算法应用于函数优化。
- 20世纪60年代中期，L.J.Fogel 等美国学者提出了**进化编程**(Evolutionary Programming, EP)。
- 20世纪60年代，德国学者I. Rechenberg和H.P. Schwefel开始了**进化策略**(Evolution Strategy, ES)的研究。
- 到20世纪90年代，GA、EP、ES走向统一，统称为进化算法，并增加了新成员——由美国斯坦福大学J. R. Koza提出**遗传编程**(Genetic Programming, GP)

# 现有进化计算范例

随着对自然界的进一步认识和研究的不深入，研究人员提出大量的进化计算范例，已超过30种。

<b>蚁群优化算法</b> Ant colony optimization	文化算法 Cultural algorithm	<b>粒子群优化算法</b> Particle swarm optimization	差异进化算法 Differential evolution
人工免疫系统 Artificial immune system	量子进化算法 Quantum-inspired evolutionary algorithm	和声搜索算法 Harmony search	细菌觅食算法 Bacterial foraging optimization
人工鱼群算法 Artificial fish swarm algorithm	人口迁移算法 Population migration algorithm	生物地理算法 Biogeography-based optimization	组搜索算法 Group search optimizer
萤火虫算法 Firefly algorithm	布谷鸟算法 Cuckoo search	蝙蝠算法 Bat algorithm	教学算法 Teaching-learning-based optimization algorithm
密母算法 Memetic algorithm	<b>人工蜂群算法</b> Artificial bee colony algorithm		

## Evolutionary Algorithms, EAs 进化计算

- 遗传算法(Genetic Algorithm, GA)
- 进化策略(Evolution Strategy, ES)
- 遗传编程(Genetic programming, GP)
- 进化编程(Evolutionary Programming, EP)
- 神经进化(NeuroEvolution)
- .....

## 遗传算法

Genetic Algorithm, GA



# 遗传算法

遗传算法是模仿生物遗传学和自然选择机理，通过人工方式构造的一类随机优化搜索算法，是对生物进化过程的一种数学仿真，是进化计算的一种最重要形式。遗传算法与传统数学模型是截然不同的，非常适用于处理传统搜索方法难以解决的复杂和非线性优化问题。

遗传算法借用了生物遗传学的观点，通过自然选择、遗传、变异等作用机制，实现各个个体的适应性的提高。这一点体现了自然界中“物竞天择，适者生存”的进化过程。

1962年Holland教授首次提出了GA算法的思想，从而吸引了大批的研究者，迅速推广到优化、搜索、机器学习、自适应控制、规划设计和人工生命等方面，并奠定了坚实的理论基础。

# 生物遗传与遗传算法概念迁移

## 生物遗传概念

适者生存

个体(Individual)

染色体(Chromosome)

基因(Gene)

适应性(Fitness)

群体(Population)

婚配(Marry)

变异(Mutation)

## 遗传算法概念

目标值比较大的解被选择的可能性大  
解

解的编码(字符串、向量等)

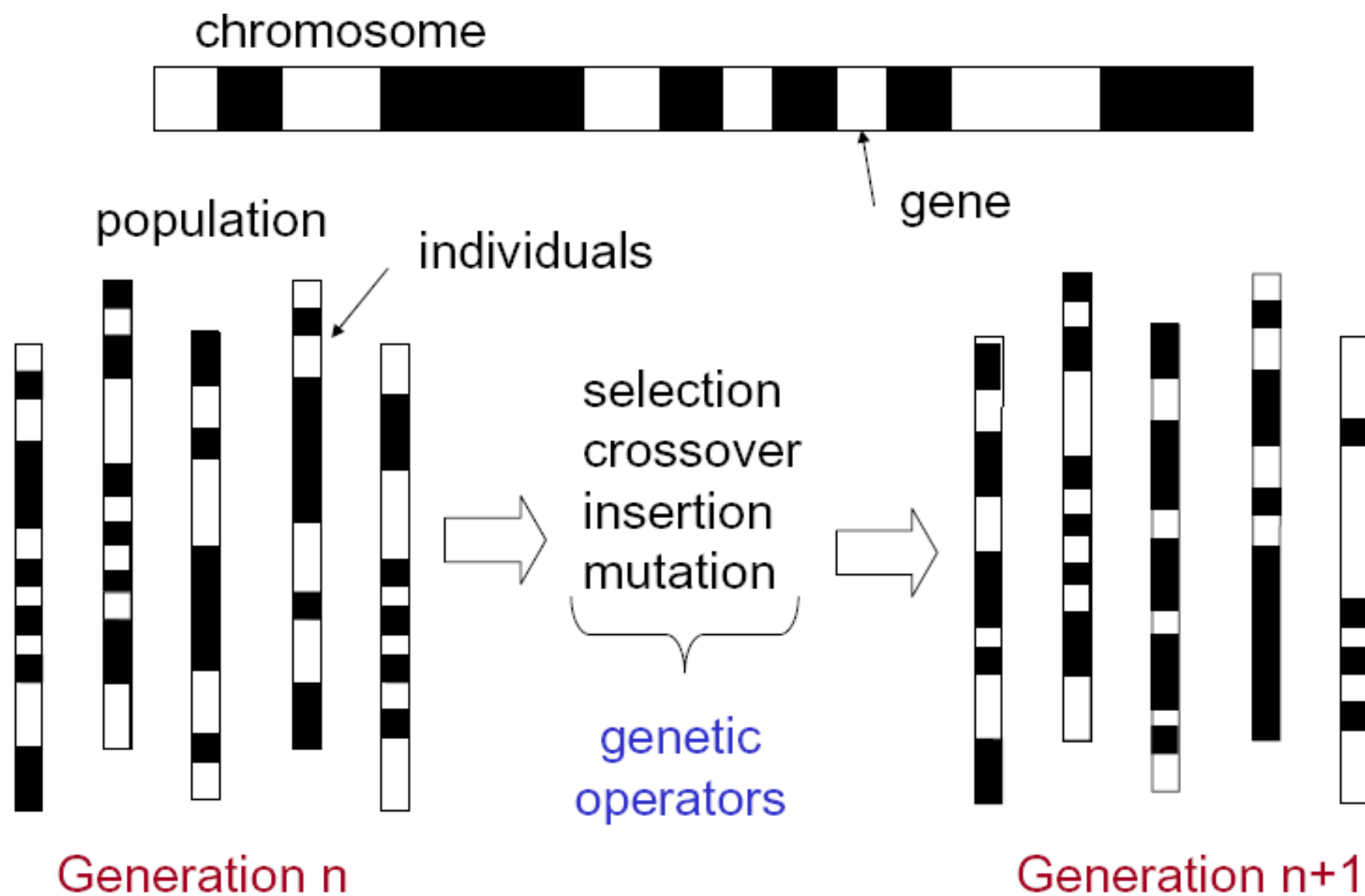
解的编码中每一分量

适应度函数与函数值

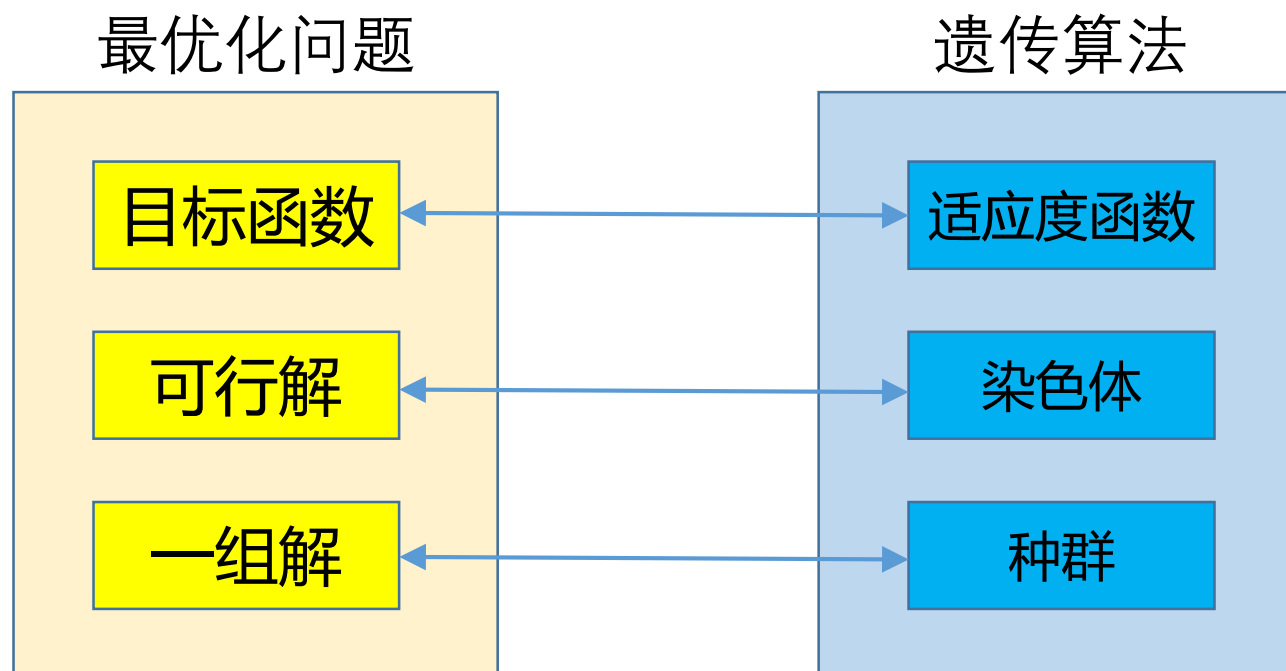
根据适应度值选定的一组解  
(解的个数为群体的规模)

交叉(Crossover)选择两个染色体进行  
交叉产生一组新的染色体的过程

编码的某一分量发生变化的过程



# 生物遗传与遗传算法概念迁移



遗传算法的基本思想：

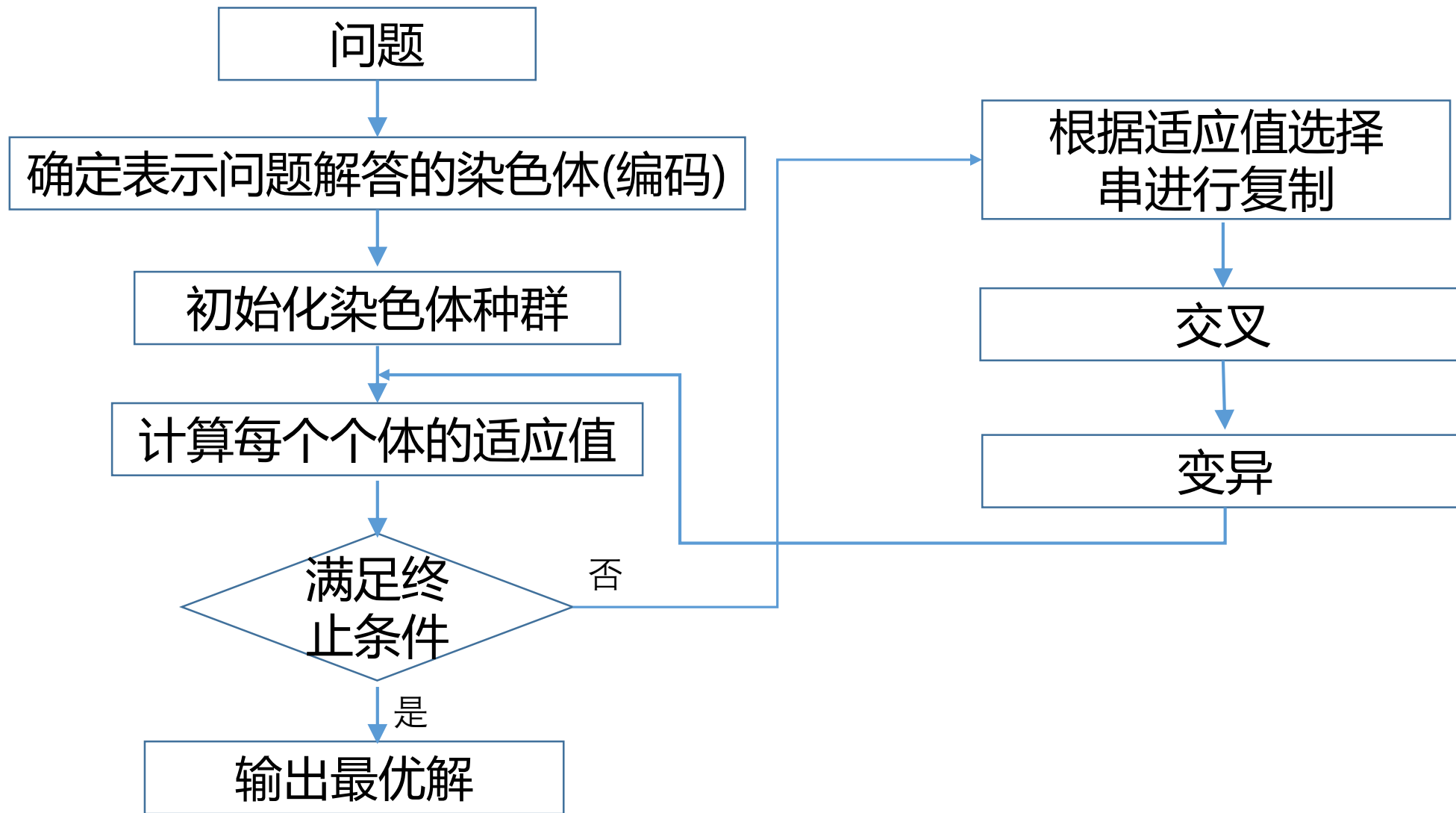
在求解问题时从多个解开始，然后通过一定的法则进行逐步迭代以产生新的解。



# 遗传算法的发展历史

- 1962年, Fraser提出了自然遗传算法。
- 1965年, Holland首次提出了人工遗传操作的重要性。
- 1967年, Bagley首次提出了遗传算法这一术语。
- 1970年, Cavicchio把遗传算法应用于模式识别中。
- 1971年, Hollstien在论文《计算机控制系统中人工遗传自适应方法》中阐述了遗传算法用于数字反馈控制的方法。
- 1975年, 美国J. Holland出版了《自然系统和人工系统的适配》; DeJong完成了重要论文《遗传自适应系统的行为分析》。
- 20世纪80年代以后, 遗传算法进入兴盛发展时期。

# 遗传算法基本流程



# 遗传算法的特点

遗传算法是一种全局优化概率算法，主要特点有：

- 遗传算法对所求解的优化问题没有太多的数学要求，由于进化特性，搜索过程中不需要问题的内在性质，无论是线性的还是非线性的，离散的还是连续的都可处理，可直接对结构对象进行操作。利用随机技术指导对一个被编码的参数空间进行效率搜索。采用群体搜索策略，易于并行化。仅用适应度函数来评估个体，并在此基础上进行遗传操作，使种群中个体之间进行信息交换。
- 进化算子的各态历经性使得遗传算法能够非常有效地进行概率意义地全局搜索。
- 遗传算法对于各种特殊问题可以提供极大地灵活性来混合构造领域独立地启发式，从而保证算法地有效性。

# 编码

编码 是遗传算法能否用到工程上的关键

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。

# 编码

## 位串编码:

**一维染色体编码方法:** 将问题空间地参数编码为一维排列的染色体的方法。

**1、二进制编码:** 用若干二进制数表示一个个体, 将原问题的解空间映射到位串空间  $B = \{0,1\}$  上, 然后再位串空间进行遗传操作。

在经典地遗传算法中, 每一个个体也称为染色体, 由二进制串组成。

0	0	1	0	1	1
(6)	(5)	(4)	(3)	(2)	(1)

**优点:** 类似于生物染色体的组成, 算法易于用生物遗传理论解释, 遗传操作如交叉、变异等易实现; 算法处理的模式数最多。

**缺点:** 位串很长, 相邻整数的二进制编码可能具有较大的Hamming距离, 降低了遗传算子的搜索效率

15: 01111 16: 10000

要先给出求解的精度

求解高维优化问题的二进制编码串长, 算法的搜索效率低

## 二进制编码：

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， $\delta$ 为二进制编码的编码精度，其公式为：

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

二进制解码：

假设某一个体的编码是：

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

则对应的解码公式为：

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

# 编/解码举例

**Example** 设  $-3.0 \leq x \leq 12.1$  , 精度要求  $\delta=1/10000$ , 由公式:

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

得:

$$\begin{aligned} 2^l &= \frac{U_{\max} - u_{\min}}{\delta} + 1 = \frac{12.1 + 3.0}{1/10000} + 1 \\ &= 151001 \end{aligned}$$

即:  $2^{17} < 151001 < 2^{18}$

即:  $x$  需要18位 {0/1} 符号表示。 如: 010001001011010000

$$\begin{aligned} \text{解码: } x &= u_{\min} + \left( \sum_{i=l}^1 b_i \cdot 2^{i-1} \right) \cdot \frac{U_{\max} - u_{\min}}{2^l - 1} \\ &= -0.3 + 70352 \times (12.1 + 3) / (2^{18} - 1) \\ &= 1.052426 \end{aligned}$$



# 编码

位串编码:

## Gray编码

Gray编码: 将二进制编码通过一个变换进行转换得到的编码。

二进制串  $\langle \beta_1 \beta_2 \dots \beta_n \rangle$

Gray  $\langle \gamma_1 \gamma_2 \dots \gamma_n \rangle$

二进制编码  $\rightarrow$  Gray编码

Gray编码  $\rightarrow$  二进制编码

$$\gamma_k = \begin{cases} \beta_1 & k = 1 \\ \beta_{k-1} \oplus \beta_k & k > 1 \end{cases}$$

$$\beta_k = \sum_{i=1}^k \gamma_i \pmod{2}$$

# 编码

位串编码：

## 实数编码

- 采用实数表达法不必进行数制转换，可直接在解的表现型上进行遗传操作。

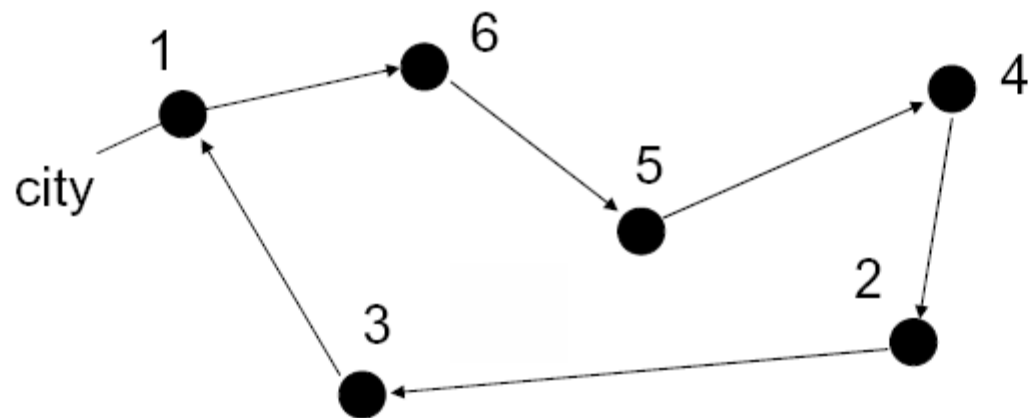
## 多参数映射编码

- 多参数映射编码的基本思想：把每个参数先进行二进制编码得到子串，再把这些子串连成一个完整的染色体。
- 多参数映射编码中的每个子串对应各自的编码参数，所以，可以有不同的串长度和参数的取值范围。

# 编/解码举例

## 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

## 1. 初始种群的产生

(1) 根据问题固有知识，把握最优解所占空间在整个问题空间中的分布范围，然后，在此分布范围内设定初始群体。

(2) 随机产生一定数目的个体，从中挑选最好的个体加到初始群体中。这种过程不断迭代，直到初始群体中个体数目达到了预先确定的规模。

# 群体设定

## 2. 种群规模的确定

- 群体规模太小，遗传算法的优化性能不太好，易陷入局部最优解。

- 群体规模太大，计算复杂。

- **模式定理**表明：长度为  $\lambda$  规模为  $M$  中有  $2^\lambda \sim M \times 2^\lambda$  个模式，并在此基础上能够不断形成和优化积木块，直到找到最优解。

# 适应度

生物体对环境的适应度不同表现出不同的生命力。在遗传算法中，每个个体对应优化问题的一个解  $x_i$ ，每个解对应一个函数值  $f_i$

$f_i$  越大， $x_i$  的值就越好，即适应度越高。

适应度函数的尺度变换：要选择合适区间的尺度，解决欺骗问题

**欺骗问题**：在遗传算法中，将所有妨碍适应度高的个体产生，从而影响遗传算法正常工作的问题统称为欺骗问题(Deceptive problem)

**过早收敛**：缩小这些个体的适应度，以降低这些超级个体的竞争力

**停滞现象**：改变原始适应值的比例关系，以提高个体之间的竞争力

# 适应度的尺度变换


适应度函数的**尺度变换(Fitness scaling)**或**定标**：对适应度函数值域的某种映射变换。

(1) 线性变换  $f' = af + b$  满足  $f'_{avg} = f_{avg}$ ,  $f'_{max} = C_{mult} \cdot f_{avg}$

$$a = \frac{(C_{mult} - 1)f_{avg}}{f_{max} - f_{avg}}$$

$$b = \frac{(f_{max} - C_{mult}f_{avg})f_{avg}}{f_{max} - f_{avg}}$$

满足最小  
适应度值非负



$$a = \frac{f_{avg}}{f_{avg} - f_{min}}$$

$$b = \frac{-f_{min}f_{avg}}{f_{avg} - f_{min}}$$

# 适应度的尺度变换

适应度函数的尺度变换(Fitness scaling)或定标：对适应度函数值域的某种映射变换。

## (1) 非线性变换

幂函数变换法

$$f' = f^K$$

指数变换法

$$f' = e^{-af}$$



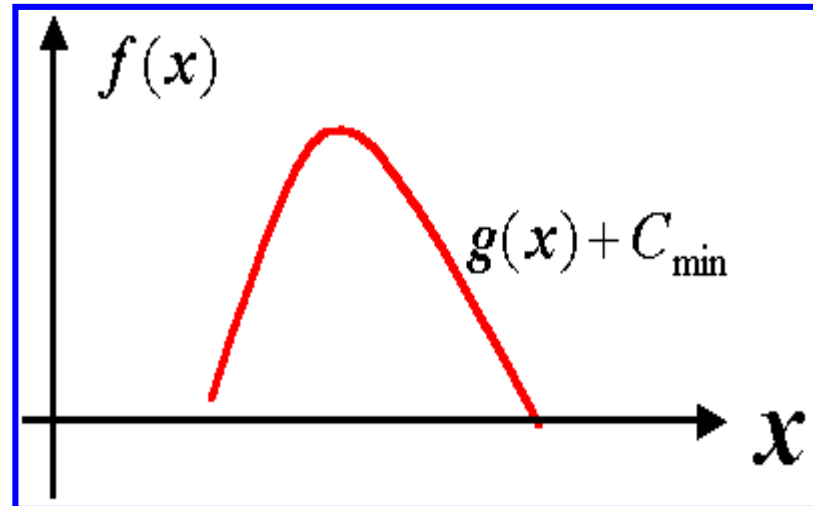
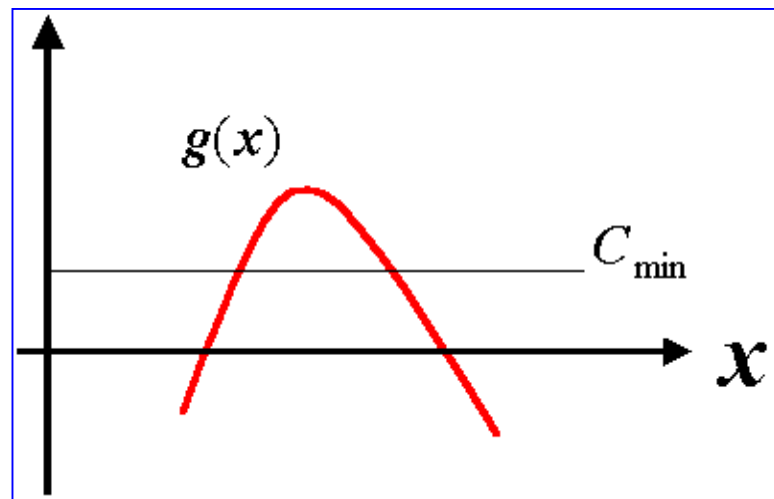
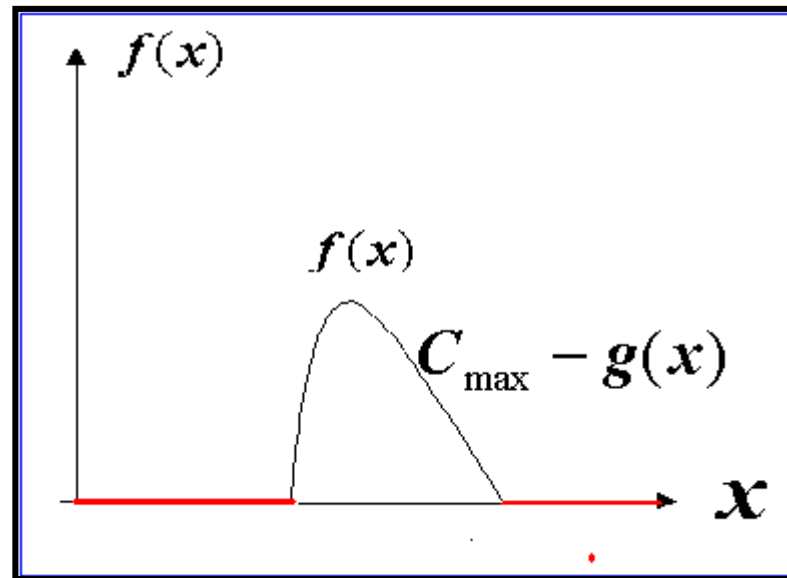
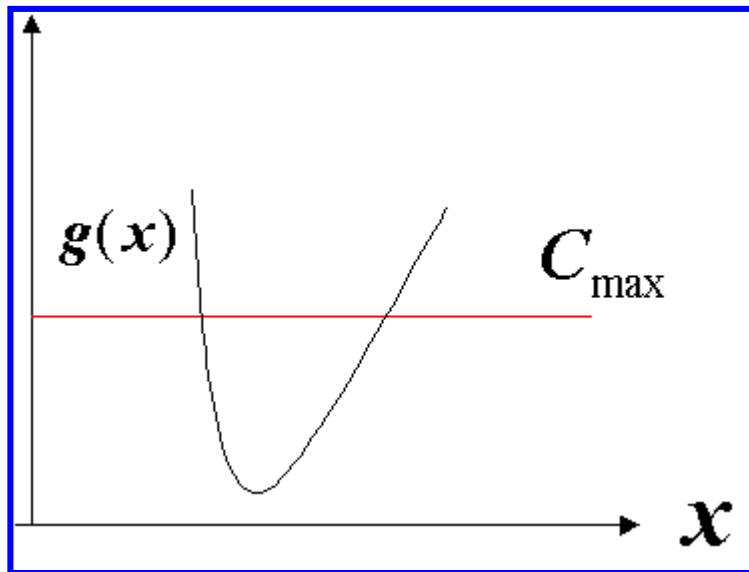
# 适应度

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

# 适应度



## (1) 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。

# 群体

一个群体是若干个个体的集合。由于每个个体代表了问题的一个解，一个群体就是问题的一些解的集合。例如：

$$\mathbf{p}_1 = \{x_1 \quad x_2 \quad \cdots \quad x_{100}\}$$

表示100个解的集合。

选择的目的是为了从当前群体中选出优良的个体，使它们有机会作为父代产生后代个体。判断个体优良与否的准则就是各自的适应度值。

个体的适应度越高，其被选择的机会就越多。“物竞天择，适者生存。”

适应度比例法(Fitness proportional model)或蒙特卡洛法(Monte Carlo)具体操作：

计算所有个体适应度的总合

$$\sum_{i=1}^n f_i$$

计算每个个体的适应度所占的比例

$$\frac{f_i}{\sum_{i=1}^n f_i}$$

以此作为选择该个体的选择概率。

各个个体被选择的概率和其适应度成正比

保证概率在  
0到1之间

# 交叉

许多生物体的繁衍是通过染色体的交叉完成的。在遗传算法中，“染色体”的交叉是重要的环节。“染色体”的交叉过程可成为一个操作算子，其实现过程如下：

选择群体中的两个个体  $x_1, x_2$

以这两个个体为双亲做基因链码的交叉，产生两个新个体：  $x'_1, x'_2$

交叉方法：在链码上随机的选择一个截断点，将  $x_1, x_2$  的基因链码断开，然后互相交换后半部分，从而组合得到新个体  $x'_1, x'_2$

父辈		后代	
$x_1$	1000 10011110	$x'_1$	1000 11000110
$x_2$	0110 11000110	$x'_2$	0110 10011110

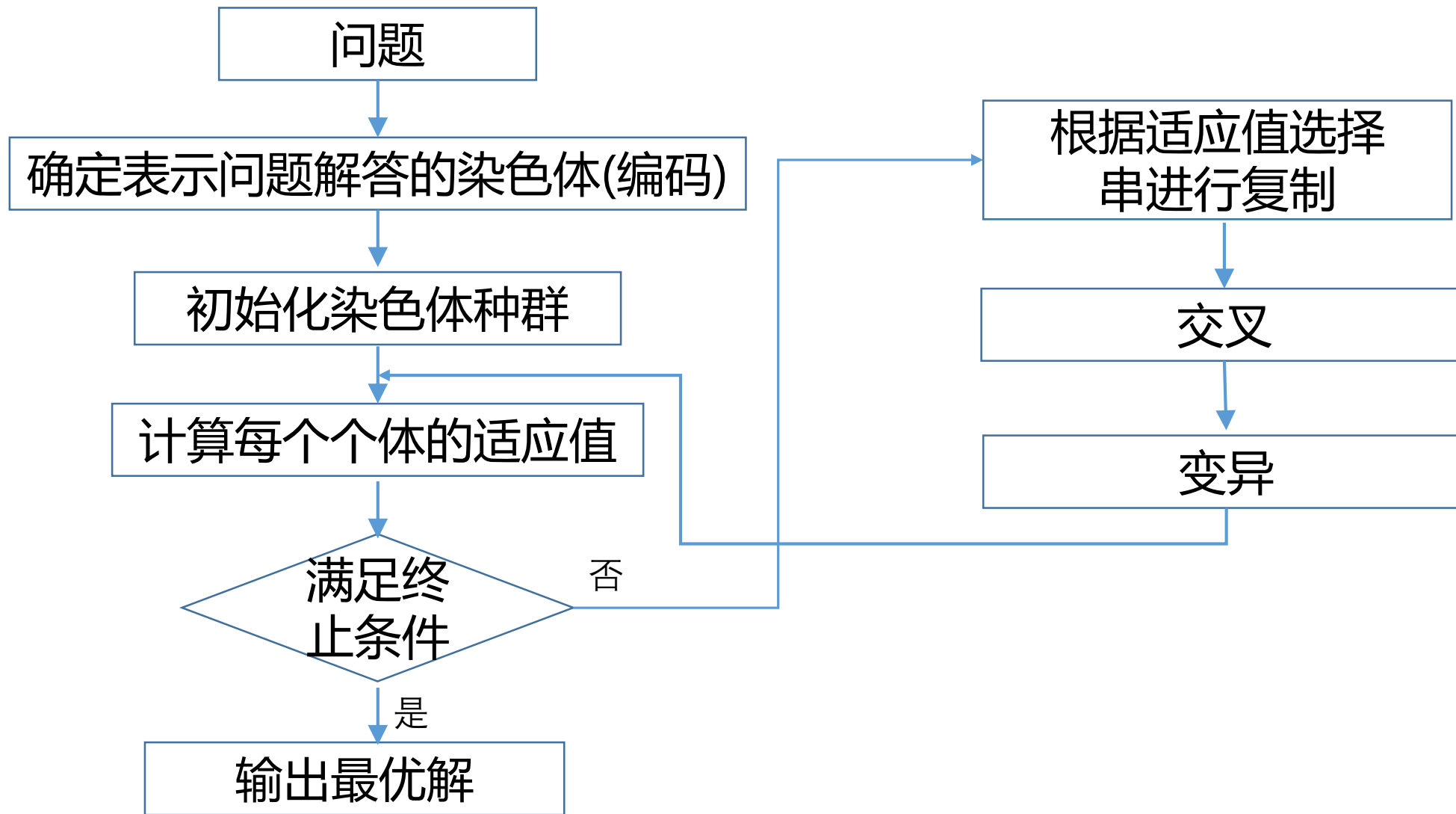
在生物的繁衍过程中，变异是一个重要的环节。通过在染色体上某些基因位置产生突变，使得新产生的个体与其它个体有所不同。在遗传算法中利用这一过程产生有“个性”的新个体。具体操作如下：

对群体中的某个个体，即基因链码，随机的选取某一位(即某个基因)，将该基因变化(将1变为0，或将0变为1)。

100011000110      100011010110

位点变异、逆转变异、插入变异、互换变异、移动变异.....

# 遗传算法基本流程





# 遗传算法步骤

1. 令进化代数 $g=0$ ，定义初始化群体 $P(g)$ 。
2. 对 $P(g)$ 中每个个体进行估值。
3. 从 $P(g)$ 中选择两个个体，并对两个个体进行交叉、变异，得到新一代群体 $P(g+1)$ ，令 $g=g+1$ 。
4. 如果终止条件满足，则算法结束；否则转到步骤2。

由上面的步骤可以看出：遗传算法的计算过程很简单，其中选择、交叉和变异是三个主要的操作过程(或操作算子)。使得遗传算法具有与其他传统方法所不同的特性。

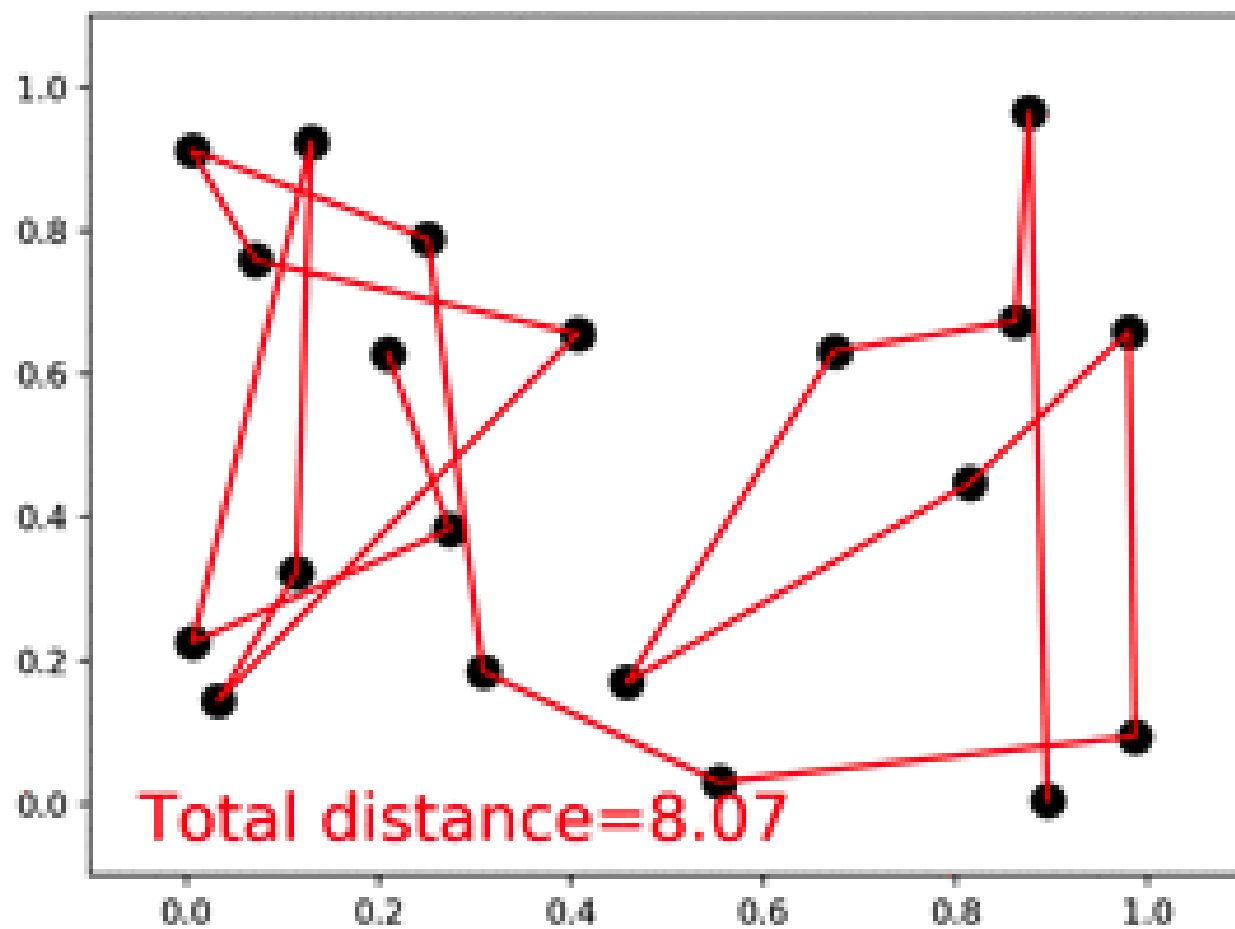
# 遗传算法的5个基本要素

1. 个体的参数编码形式;
2. 初始群体的设定;
3. 适应度函数的设计;
4. 控制参数的设定;

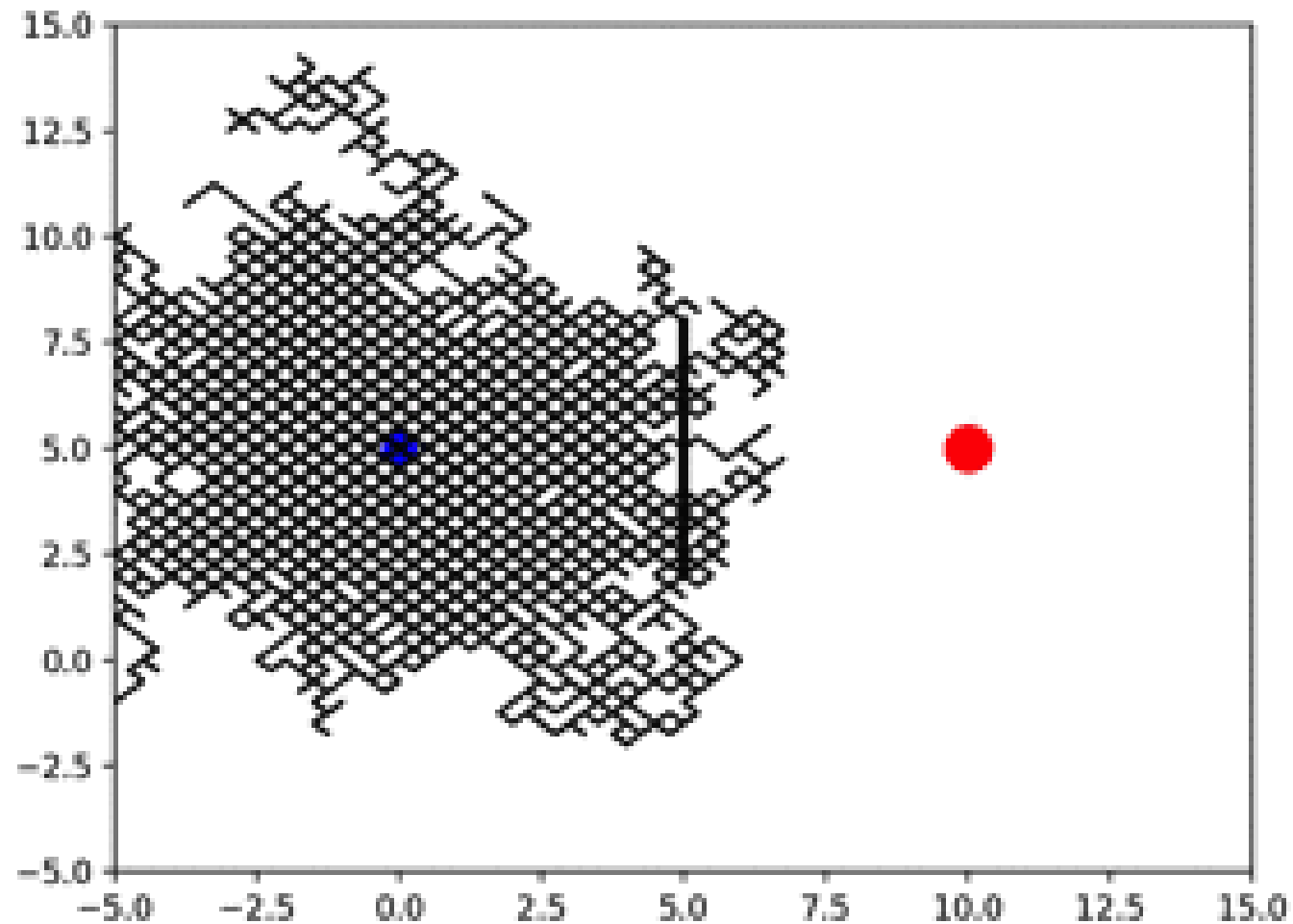
群体的规模, 在遗传操作中的概率应用。

5. 遗传操作设计。

# 旅行商问题(TSP)



# 最短路径问题



# 遗传算法的应用

## (1) 函数优化

函数优化是遗传算法的经典应用领域，也是对遗传算法进行性能评价的常用算例。

对于一些非线性、多模型、多目标的函数优化问题，用其他优化方法较难求解，用遗传算法可以方便地得到较好的结果。

# 遗传算法的应用

## (2) 组合优化

随着问题规模的增大，组合优化问题的搜索空间也急剧扩大，有时在目前的计算机上用枚举法很难或甚至不可能求出其精确最优解。对这类复杂问题，人们已意识到应把主要精力放在寻求其满意解上，而遗传算法是寻求这种满意解的最佳工具之一。实践证明，遗传算法对于组合优化中的NP完全问题非常有效。

例如，遗传算法已经在求解旅行商问题、背包问题、装箱问题、图形划分问题等方面得到成功的应用。

# 遗传算法的应用

## (3) 生产调度问题

生产调度问题在很多情况下所建立起来的数学模型难以精确求解，即使经过一些简化之后可以进行求解，也会因简化得太多而使得求解结果与实际相差甚远。而目前在现实生产中也主要是靠一些经验来进行调度。现在遗传算法已成为解决复杂调度问题的有效工具，在单件生产车间调度、流水线生产车间调度、生产规划、任务分配等方面遗传算法都得到了有效的应用。

# 遗传算法的应用

## (4) 自动控制

在自动控制领域中很多与优化相关的问题需要求解，遗传算法已在其中得到了初步的应用，并显示出了良好的效果。

例如用遗传算法进行航空控制系统的优化、使用遗传算法设计空间交会控制器、基于遗传算法的模糊控制器的优化设计、基于遗传算法的参数辨识、基于遗传算法的模糊控制规则的学习、利用遗传算法进行人工神经网络的结构优化设计和权值学习等，都显示出了遗传算法在这些领域中应用的可能性。



# 遗传算法的应用

## (5) 机器人学

机器人是一类复杂的难以精确建模的人工系统，而遗传算法的起源就来自于对人工自适应系统的研究，所以机器人学理所当然地成为遗传算法的一个重要应用领域。例如，遗传算法已经在移动机器人路径规划、关节机器人运动轨迹规划、机器人逆运动学求解、细胞机器人的结构优化和行为协调等方面得到研究和应用。

# 遗传算法的应用

## (6) 图像处理

图像处理是计算机视觉中的一个重要研究领域。在图像处理过程中，如扫描、特征提取、图像分割等不可避免地会存在一些误差，这些误差会影响图像处理的效果。如何使这些误差最小是使计算机视觉达到实用化的重要要求。

遗传算法在这些图像处理中的优化计算方面找到了用武之地，日前已在模式识别、图像恢复、图像边缘特征提取等方面得到了应用。

# 遗传算法的应用

## (7) 人工生命

人工生命是用计算机、机械等人工媒体模拟或构造出的具有自然生物系统特有行为的人造系统。自组织能力和自学习能力是人工生命的两大主要特征。人工生命与遗传算法有着密切的关系，基于遗传算法的进化模型是研究人工生命现象的重要基础理论。

虽然人工生命的研究尚处于启蒙阶段，但遗传算法已在其进化模型、学习模型、行为模型、自组织模型等方面显示出了初步的应用能力，并且必将得到更为深入的应用和发展。人工生命与遗传算法相辅相成，遗传算法为人工生命的研究提供了一个有效的工具，人工生命的研究也必将促进遗传算法的进一步发展。

# 遗传算法的应用

## (8) 遗传编程

Koza发展了遗传编程的概念，他使用了以LISP语言所表示的编码方法，基于对一种树型结构所进行的遗传操作来自动生成计算机程序。虽然遗传编程的理论尚未成熟，应用也有一些限制，但它已成功地应用于人工智能、机器学习等领域。

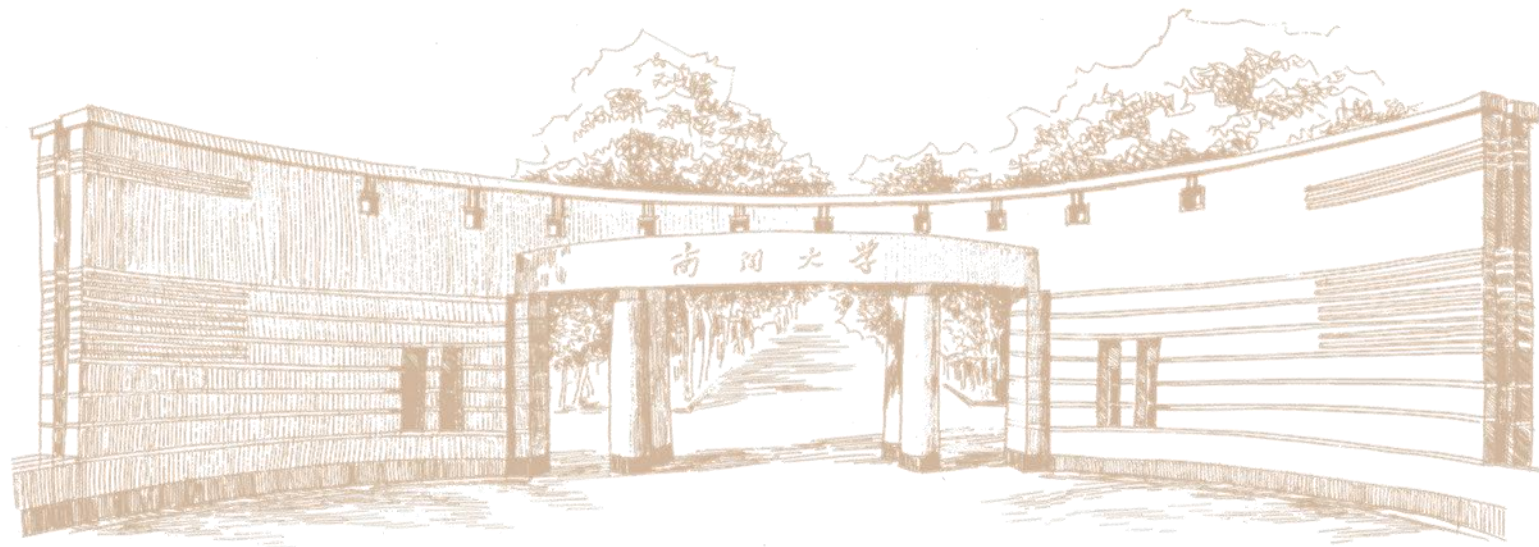
# 遗传算法的应用

## (9) 机器学习

学习能力是高级自适应系统所应具备的能力之一。基于遗传算法的机器学习，特别是分类器系统，在很多领域中都得到了应用。例如，遗传算法被用于学习模糊控制规则，利用遗传算法来学习隶属度函数，从而更好地改进了模糊系统的性能；基于遗传算法的机器学习可用来调整人工神经网络的连接权，也可用于人工神经网络的网络结构优化设计；分类器系统也在学习式多机器人路径规划系统中得到了成功的应用。

Q&A

THANKS!



## 1. 个体选择概率分配方法

- 选择操作也称为复制 (reproduction) 操作：从当前群体中按照一定概率选出优良的个体，使它们有机会作为父代繁殖下一代子孙。
- 判断个体优良与否的准则是各个个体的适应度值：个体适应度越高，其被选择的机会就越多。

## 1. 个体选择概率分配方法

### (1) 适应度比例方法 (fitness proportional model) 或蒙特卡罗法 (Monte Carlo)

- 各个个体被选择的概率和其适应度值成比例。
- 个体  $i$  被选择的概率为：

$$p_{si} = \frac{f_i}{\sum_{i=1}^M f_i}$$



## 1. 个体选择概率分配方法

### (2) 排序方法 (rank-based model)

#### ① 线性排序: J. E. Baker

- 群体成员按适应值大小从好到坏依次排列:  $x_1, x_2, \dots, x_N$
- 个体  $x_i$  分配选择概率  $p_i$

$$p_i = \frac{a - bi}{M(M + 1)}$$

- 按转盘式选择的方式选择父体

## 1. 个体选择概率分配方法

### (2) 排序方法 (rank-based model)

#### ② 非线性排序: Z. Michalewicz

- 将群体成员按适应值从好到坏依次排列，并按下式分配选择概率：

$$p_i = \begin{cases} q(1-q)^{i-1} & i = 1, 2, \dots, M-1 \\ (1-q)^{M-1} & i = M \end{cases}$$

## 1. 个体选择概率分配方法

### (2) 排序方法 (rank-based model)

- 可用其他非线性函数来分配选择概率，只要满足以下条件：

(1) 若  $P = \{x_1, x_2, \dots, x_M\}$  且  $f(x_1) \geq f(x_2) \geq \dots \geq f(x_M)$ , 则  $p_i$  满足

$$p_1 \geq p_2 \geq \dots \geq p_M$$

$$(2) \sum_{i=1}^M p_i = 1$$

## 2. 选择个体方法

### (1) 转盘赌选择 (roulette wheel selection)

- 按个体的选择概率产生一个轮盘，轮盘每个区的角度与个体的选择概率成比例。
- 产生一个随机数，它落入转盘的哪个区域就选择相应的个体交叉。

个体	1	2	3	4	5	6	7	8	9	10	11
适应度	2.0	1.8	1.6	1.4	1.2	1.0	0.8	0.6	0.4	0.2	0.1
选择概率	0.18	0.16	0.15	0.13	0.11	0.09	0.07	0.06	0.03	0.02	0.0
累积概率	0.18	0.34	0.49	0.62	0.73	0.82	0.89	0.95	0.98	1.00	1.00

第1轮产生一个随机数：0.81

第2轮产生一个随机数：0.32

## 2. 选择个体方法

### (2) 锦标赛选择方法 (tournament selection model)

- 锦标赛选择方法：从群体中随机选择个个体，将其中适应度最高的个体保存到下一代。这一过程反复执行，直到保存到下一代的个体数达到预先设定的数量为止。
- 随机竞争方法 (stochastic tournament)：每次按赌轮选择方法选取一对个体，然后让这两个个体进行竞争，适应度高者获胜。如此反复，直到选满为止。

## 2. 选择个体方法

(3)  $(\mu, \lambda)$  和  $\mu + \lambda$  选择

$(\mu, \lambda)$  选择

- 从规模为  $\mu$  的群体中随机选取个体通过重组和变异生成  $\lambda (\geq \mu)$  个后代,
- 再选取  $\mu$  个最优的后代作为新一代种群。

$\mu + \lambda$  选择

- 从  $\lambda (\geq \mu)$  个后代与其父体共  $\mu + \lambda$  中选取  $\mu$  个最优的后代。

## 2. 选择个体方法

### (4) Boltzmann锦标赛选择

- 随机选取两个个体  $x_1, x_2$ ，若  $|f(x_1) - f(x_2)| \geq \theta$  则选择适应值好的作为胜者，否则计算概率  $p = \exp[-|f(x_1) - f(x_2)|/T]$ ，若  $p > \text{random}[0,1)$ ，选择差解，否则选择好解。

### (5) 最佳个体保存方法

- **最佳个体（elitist model）保存方法：**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中，保证遗传算法终止时得到的最后结果一定是历代出现过的最高适应度的个体。

## 1. 基本的交叉算子

### (1) 一点交叉 (single-point crossover)

- 一点交叉：在个体串中随机设定一个交叉点，实行交叉时，该点前或后的两个个体的部分结构进行互换，并生成两个新的个体。

### (2) 二点交叉 (two-point crossover)

- 二点交叉：随机设置两个交叉点，将两个交叉点之间的码串相互交换。



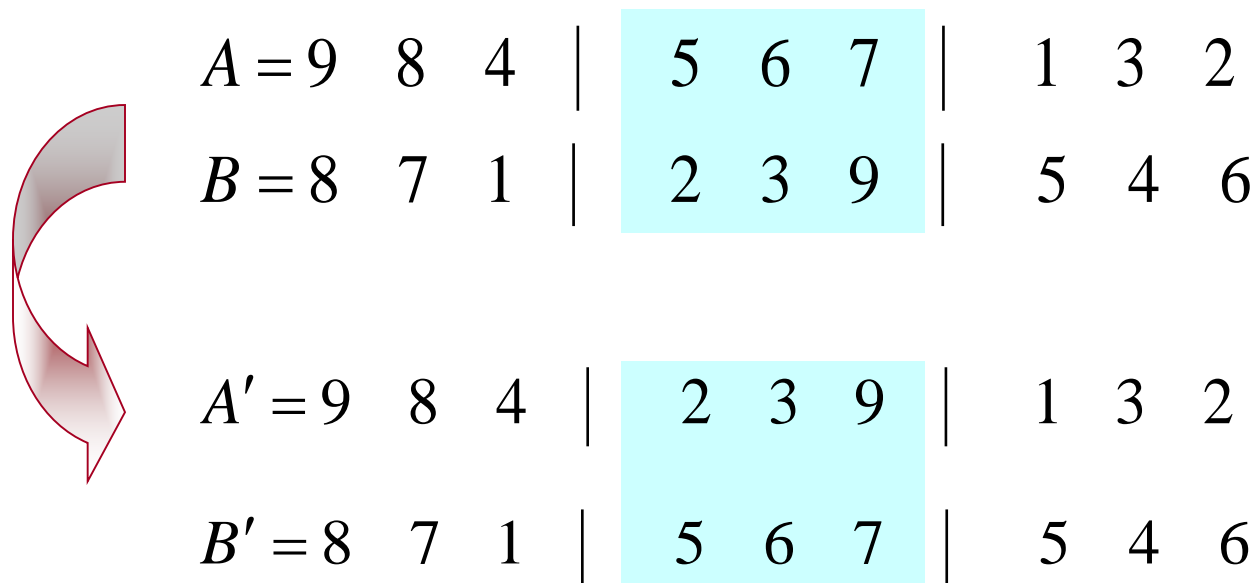
## 1. 基本的交叉算子（续）

### （3）均匀交叉（uniform crossover）或一致交叉

- 均匀交叉：按照均匀概率抽取一些位，每一位是否被选取都是随机的，并且独立于其他位。然后将两个个体被抽取位互换组成两个新个体。

## 2. 修正的交叉方法

### (1) 部分匹配交叉PMX: Goldberg D. E.和R. Lingle(1985)



## 2. 修正的交叉方法(续)

### (2) 顺序交叉OX: Davis L. (1985)



$A' = H$	8	4		5	6	7		1	$H$	$H$
$B' = 8$	$H$	1		2	3	9		$H$	4	$H$

$A'' = 5$	6	7		$H$	$H$	$H$		1	8	4
$B'' = 2$	3	9		$H$	$H$	$H$		4	8	1

$A''' = 5$	6	7		2	3	9		1	8	4
$B''' = 2$	3	9		5	6	7		4	8	1

## 3. 实数编码的交叉方法

### (1) 离散交叉 (discrete crossover)

- **部分离散交叉**：在父解向量中选择一部分分量，然后交换这些分量。 —— 二进制的点式交叉
- **整体离散交叉**：以0.5的概率交换父体  $s_1$  与  $s_2$  的所有分量。 —— 二进制编码的均匀性交叉

### 3. 实数编码的交叉方法（续）

#### (2) 算术交叉 (arithmetical crossover)

- **部分算术**: 先在父解向量中选择一部分分量, 如第  $k$  个分量以后的所有分量, 然后生成  $n-k$  个  $[0, 1]$  区间的随机数, 并将两个后代定义为:

$$s_z = (v_1^{(1)}, \dots, v_k^{(1)}, a_{k+1} v_{k+1}^{(1)} + (1 - a_{k+1}) v_{k+1}^{(2)}, \dots, a_n v_n^{(1)} + (1 - a_n) v_n^{(2)})$$

$$s_w = (v_1^{(2)}, \dots, v_k^{(2)}, a_{k+1} v_{k+1}^{(2)} + (1 - a_{k+1}) v_{k+1}^{(1)}, \dots, a_n v_n^{(2)} + (1 - a_n) v_n^{(1)})$$

$$a_{k+1} = \dots = a_n$$

### 3. 实数编码的交叉方法（续）

#### (2) 算术交叉 (arithmetical crossover)

■ **整体算术交叉**：先生成  $n$  个  $[0,1]$  区间的随机数，则后代分别定义为：

$$z_i = a_i v_i^{(1)} + (1 - a_i) v_i^{(2)} = v_i^{(2)} + a_i (v_i^{(1)} - v_i^{(2)})$$

$$w_i = a_i v_i^{(2)} + (1 - a_i) v_i^{(1)} = v_i^{(1)} + a_i (v_i^{(2)} - v_i^{(1)})$$

$$a_1 = a_2 = \dots = a_n$$

## 1. 整数编码的变异方法

- (1) **位点变异**: 群体中的个体码串, 随机挑选一个或多个基因座, 并对这些基因座的基因值以变异概率作变动。
- (2) **逆转变异**: 在个体码串中随机选择两点 (逆转点), 然后将两点之间的基因值以逆向排序插入到原位置中。
- (3) **插入变异**: 在个体码串中随机选择一个码, 然后将此码插入随机选择的插入点中间。

## 1. 整数编码的变异方法（续）

- (4) **互换变异**：随机选取染色体的两个基因进行简单互换。
- (5) **移动变异**：随机选取一个基因，向左或者向右移动一个随机位数。
- (6) **自适应变异**：类似于位点变异，但变异概率随群体中个体的多样性程度而自适应调整。



## 2. 实数编码的变异方法

### (1) 均匀性变异

$s = (v_1, v_2, \dots, v_k, \dots, v_n)$ : 父解

$s' = (v_1, v_2, \dots, v'_k, \dots, v_n)$ : 变异产生的后代

- 均匀性变异: 在父解向量中随机地选择一个分量（第  $k$  个），然后在  $[a_k, b_k]$  中以均匀概率随机选择  $v'_k$  代替  $v_k$  以得到  $s'$ ，即

$$s' = \begin{cases} v_i, & i \neq k \\ v'_k, & i = k \end{cases}$$

## 2. 实数编码的变异方法（续）

### (2) 正态性变异 (normal distributed mutation)

解向量  $s = (v_1, v_2, \dots, v_n)$       摄动向量  $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n)$

被选个体  $(s, \sigma)$             新个体  $(s', \sigma')$

$$\sigma'_i = \sigma_i \exp(N_i(0, \Delta\sigma))$$

$$v'_i = v_i + N(0, \sigma'_i) \quad i = 1, 2, \dots, n$$

## 2. 实数编码的变异方法（续）

### (3) 非一致性变异

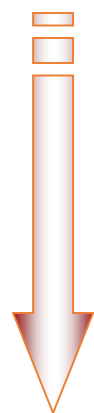
- Z. Michalewicz首先提出将变异算子的结果与演化代数联系起来。
- 在演化初期，变异范围相对较大，而随着演化的推进，变异范围越来越小，起着一种对演化系统的微调作用。

## 2. 实数编码的变异方法（续）

### (3) 非一致性变异

$s = (v_1, v_2, \dots, v_k, \dots, v_n)$ : 父解

$[a_k, b_k]$ : 区间



$$v'_k = \begin{cases} v_k + \Delta(t, b_k - v_k), & rnd(2) = 0 \\ v_k + \Delta(t, b_k - a_k), & rnd(2) = 1 \end{cases}$$

$$\Delta(t, y) = y(1 - r^{(1-t/T)^2})$$

$s' = (v_1, v_2, \dots, v_{k-1}, v'_k, \dots, v_n)$ : 变异后的解

## 2. 实数编码的变异方法（续）

### (4) 自适应变异

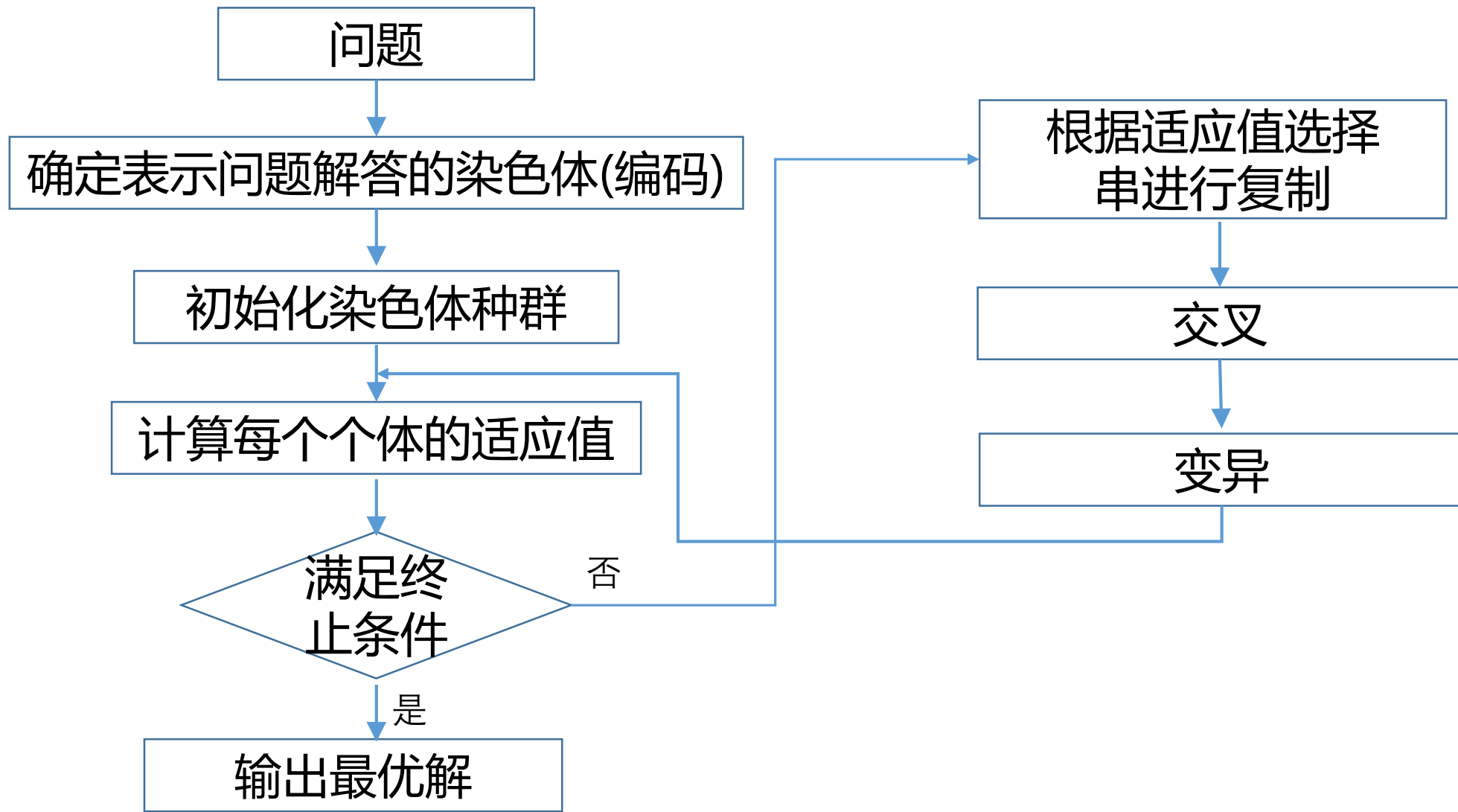
$s = (v_1, v_2, \dots, v_n)$ : 解空间的一个向量

变异温度:  $T = 1 - \frac{f(s)}{f_{\max}}$

■ 自适应变异方式与非一致性变异算子相同，只是将其中的演化代数  $t$  改为  $T$ ，函数表达式变为：

$$\Delta(T, y) = y(1 - r^{T^\lambda})$$

# 遗传算法基本流程



## 遗传算法步骤

(1) 使用随机方法或者其它方法，产生一个有 $N$ 个染色体的初始群体  $pop(1)$ ,  $t := 1$  ;

(2) 对群体中的每一个染色体 $pop_i(t)$ , 计算其适应值

$$f_i = fitness(pop_i(t))$$

(3) 若满足停止条件，则算法停止；否则，以概率

$$p_i = f_i / \sum_{j=1}^N f_j$$

从 $pop(t)$ 中随机选择一些染色体构成一个新种群

$$newpop(t+1) = \{pop_j(t) | j = 1, 2, \dots, N\}$$

- (4) 以概率  $p_c$  进行交叉产生一些新的染色体，得到一个  
新的群体

$$crosspop(t+1)$$

- (5) 以一个较小的概率  $p_m$  使染色体的一个基因发生变异，  
形成  $mutpop(t+1)$ ；  $t := t+1$ ，成为一个新的群体

$$pop(t) = mutpop(t+1)$$

返回 (2)。



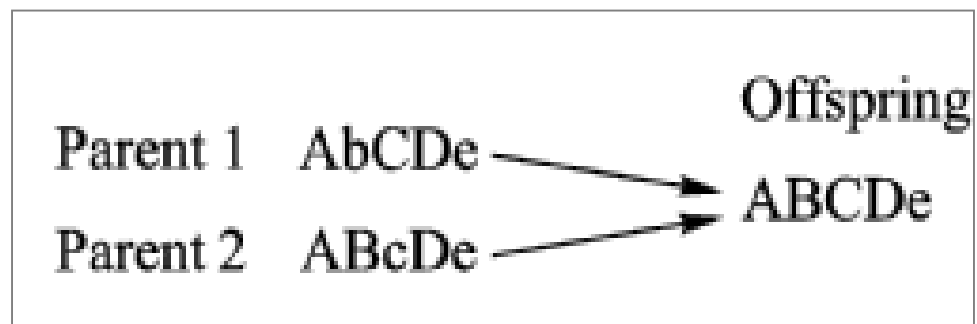
# 遗传算法的改进算法

- 双倍体遗传算法
- 双种群遗传算法
- 自适应遗传算法

# 双倍体遗传算法

## 1. 基本思想

- 双倍体遗传算法采用**显性**和**隐性**两个染色体同时进行进化，提供了一种记忆以前有用的基因块的功能。
- 双倍体遗传算法采用**显性遗传**。



- 双倍体遗传延长了有用基因块的寿命，提高了算法的收敛能力，在变异概率低的情况下能保持一定水平的多样性。

# 双倍体遗传算法

## 2. 双倍体遗传算法的设计

- (1) **编码/解码**: 两个染色体（显性、隐性）
- (2) **复制算子**: 计算显性染色体的适应度，按照显性染色体的复制概率将个体复制到下一代群体中。
- (3) **交叉算子**: 两个个体的显性染色体交叉、隐性染色体也同时交叉。
- (4) **变异算子**: 个体的显性染色体按正常的变异概率变异；隐性染色体按较大的变异概率变异。
- (5) **双倍体遗传算法显隐性重排算子**: 个体中适应值较大的染色体设为显性染色体，适应值较小的染色体设为隐性染色体。

# 双种群遗传算法

## 1. 基本思想

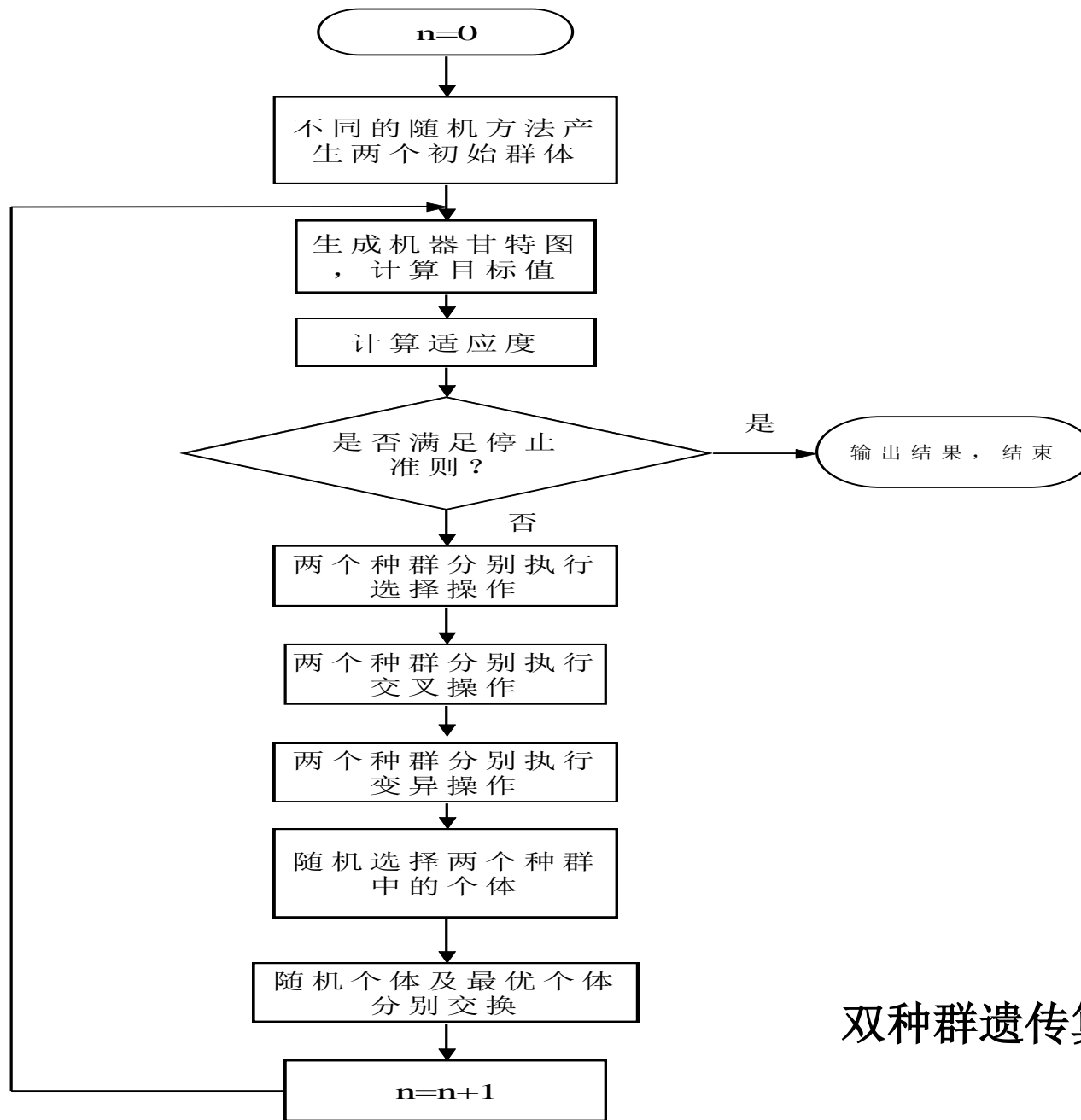
- 在遗传算法中使用多种群同时进化，并交换种群之间优秀个体所携带的遗传信息，以打破种群内的平衡态达到更高的平衡态，有利于算法跳出局部最优。
- **多种群遗传算法**：建立两个遗传算法群体，分别独立地运行复制、交叉、变异操作，同时当每一代运行结束以后，选择两个种群中的随机个体及最优个体分别交换。

# 双种群遗传算法

## 2. 双种群遗传算法的设计

- (1) 编码/解码设计
- (2) 交叉算子、变异算子
- (3) 杂交算子

设种群 $A$ 与种群 $B$ ，当 $A$ 与 $B$ 种群都完成了选择、交叉、变异算子后，产生一个随机数 $num$ ，随机选择 $A$ 中 $num$ 个个体与 $A$ 中最优个体，随机选择 $B$ 中 $num$ 个个体与 $B$ 中最优个体，交换两者，以打破平衡态。



双种群遗传算法程序流程图

# 自适应遗传算法

## 1. 基本思想

■ Srinivas M., Patnaik L. M.等在1994年提出一种自适应遗传算法(adaptive genetic algorithms, AGA):  $P_c$ 和 $P_m$ 能随适应度自动改变。

■ AGA: 当种群各个体适应度趋于一致或者趋于局部最优时, 使  $P_c$ 和 $P_m$ 增加, 以跳出局部最优; 而当群体适应度比较分散时, 使  $P_c$ 和 $P_m$ 减少, 以利于优良个体的生存。

■ 同时, 对于适应度高于群体平均适应值的个体, 选择较低的  $P_c$ 和 $P_m$ , 使得该解得以保护进入下一代; 对低于平均适应值的个体, 选择较高的  $P_c$ 和 $P_m$ 值, 使该解被淘汰。

## 2. 自适应遗传算法的步骤

- (1) 编码/解码设计。
- (2) 初始种群产生： $N$  ( $N$  是偶数) 个候选解，组成初始解集。
- (3) 定义适应度函数为  $f = 1/ob$ ，计算适应度  $f_i$ 。
- (4) 按轮盘赌规则选择  $N$  个个体，计算  $f_{avg}$  和  $f_{max}$ 。
- (5) 将群体中的各个个体随机搭配成对，共组成  $N/2$  对，对每一对个体，按照自适应公式计算自适应交叉概率  $P_c$ ，随机产生  $R(0,1)$ ，如果  $R < P_c$  则对该对染色体进行交叉操作。



## 2. 自适应遗传算法的步骤（续）

- (6) 对于群体中的所有个体，共 $N$ 个，按照自适应变异公式计算自适应变异概率，随机产生  $R(0,1)$ ，如果  
则对该染色体进行变异操作。
- (7) 计算由交叉和变异生成新个体的适应度，新个体与父代一起构成新群体。
- (8) 判断是否达到预定的迭代次数，是则结束；否则转  
(4)。

# 自适应遗传算法

## 3. 适应的交叉概率与变异概率

$$P_c = \begin{cases} \frac{k_1(f_{\max} - f')}{f_{\max} - f_{\text{avg}}}, & f' > f_{\text{avg}} \\ k_2, & f' \leq f_{\text{avg}} \end{cases} \quad P_m = \begin{cases} \frac{k_3(f_{\max} - f)}{f_{\max} - f_{\text{avg}}}, & f > f_{\text{avg}} \\ k_4, & f \leq f_{\text{avg}} \end{cases}$$

■ 普通自适应算法中，当个体适应度值越接近最大适应度值时，交叉概率与变异概率就越小；当等于最大适应度值时，交叉概率和变异概率为零。

■ 改进的思想：当前代的最优个体不被破坏，仍然保留（最优保存策略）；但较优个体要对应于更高的交叉概率与变异概率。

# 自适应遗传算法

## 3. 自适应的交叉概率与变异概率（续）

### ■ F—自适应方法：

当  $f' = f_{\max}$ ,  $P_c = P_{c2} > 0$ ;

当  $f = f_{\max}$ ,  $P_m = P_{m2} > 0$ 。

$$P_c = \begin{cases} P_{c1} - \frac{(P_{c1} - P_{c2})(f' - f_{\text{avg}})}{f_{\max} - f_{\text{avg}}}, & f' > f_{\text{avg}} \\ P_{c1}, & f' \leq f_{\text{avg}} \end{cases}$$

$$P_m = \begin{cases} P_{m1} - \frac{(P_{m1} - P_{m2})(f_{\max} - f)}{f_{\max} - f_{\text{avg}}}, & f > f_{\text{avg}} \\ P_{m1}, & f \leq f_{\text{avg}} \end{cases}$$

$$P_{c1} = 0.9, P_{c2} = 0.6, P_{m1} = 0.1, P_{m2} = 0.001$$

## 3. 自适应的交叉概率与变异概率（续）

### ■ S—自适应方法：

$$P_c = \begin{cases} k_1 \sin\left(\frac{\pi}{2} \cdot \frac{f_{\max} - f'}{f_{\max} - f_{\text{avg}}}\right), & f' > f_{\text{avg}} \\ k_2, & f' \leq f_{\text{avg}} \end{cases}$$

$$P_m = \begin{cases} k_3 \sin\left(\frac{\pi}{2} \cdot \frac{f_{\max} - f}{f_{\max} - f_{\text{avg}}}\right), & f > f_{\text{avg}} \\ k_4, & f \leq f_{\text{avg}} \end{cases}$$

$$k_1 = 1.0, k_2 = 1.0, k_3 = 0.5, k_4 = 0.5,$$

# 自适应遗传算法

## 3. 自适应的交叉概率与变异概率（续）

### ■ C — 自适应方法:

$$P_c = \begin{cases} 1 - k_1 \cos\left(\frac{\pi}{2} \cdot \frac{f_{\max} - f'}{f_{\max} - f_{\text{avg}}}\right), & f' > f_{\text{avg}} \\ k_2, & f' \leq f_{\text{avg}} \end{cases}$$

$$P_m = \begin{cases} 1 - k_3 \cos\left(\frac{\pi}{2} \cdot \frac{f_{\max} - f}{f_{\max} - f_{\text{avg}}}\right), & f > f_{\text{avg}} \\ k_4, & f \leq f_{\text{avg}} \end{cases}$$

$$k_1 = 1.0, k_2 = 1.0, k_3 = 0.5, k_4 = 0.5,$$

# 基于遗传算法的生产调度方法

- 基于遗传算法的流水车间调度方法
- 基于遗传算法的混合流水车间调度方法

# 基于遗传算法的流水车间调度方法

## 1. 流水车间调度问题

■ 问题描述:  $n$  个工件要在  $m$  台机器上加工, 每个工件需要经过  $m$  道工序, 每道工序要求不同的机器,  $n$  个工件在  $m$  台机器上的加工顺序相同。工件在机器上的加工时间是给定的, 设为

$$t_{ij} (i = 1, \dots, n; j = 1, \dots, m)$$

■ 问题的目标: 确定  $n$  个工件在每台机器上的最优加工顺序, 使最大流程时间达到最小。

# 基于遗传算法的流水车间调度方法

## 1. 流水车间调度问题

### ■ 假设:

- (1) 每个工件在机器上的加工顺序是给定的。
- (2) 每台机器同时只能加工一个工件。
- (3) 一个工件不能同时在不同的机器上加工。
- (4) 工序不能预定。
- (5) 工序的准备时间与顺序无关，且包含在加工时间中。
- (6) 工件在每台机器上的加工顺序相同，且是确定的。



# 基于遗传算法的流水车间调度方法

## 1. 流水车间调度问题

### ■ 问题的数学模型:

$c(j_i, k)$  : 工件 $j_i$ 在机器 $k$ 上的加工完工时间,  $\{j_1, j_2, \dots, j_n\}$ : 工件的调度  
 $n$ 个工件、 $m$ 台机器的流水车间调度问题的完工时间:

$$c(j_1, 1) = t_{j_1 1}$$

$$c(j_1, k) = c(j_1, k-1) + t_{j_1 k}, \quad k = 2, \dots, m$$

$$c(j_i, 1) = c(j_{i-1}, 1) + t_{j_i 1}, \quad i = 2, \dots, n$$

$$c(j_i, k) = \max\{c(j_{i-1}, k), c(j_i, k-1)\} + t_{j_i k}, \quad i = 2, \dots, n; k = 2, \dots, m$$

最大流程时间:  $c_{\max} = c(j_n, m)$

调度目标: 确定 $\{j_1, j_2, \dots, j_n\}$ 使得 $c_{\max}$ 最小

# 基于遗传算法的流水车间调度方法

## 2. 求解流水车间调度问题的遗传算法设计

### (1) FSP的编码方法

■ 对于FSP，最自然的编码方式是用染色体表示工件的顺序。

对于有四个工件的FSP，第  $k$  个染色体  $v_k = [1, 2, 3, 4]$ ，表示工件的加工顺序为：  $j_1, j_2, j_3, j_4$  。

# 基于遗传算法的流水车间调度方法

## 2. 求解流水车间调度问题的遗传算法设计

### (2) FSP的适应度函数

$c_{\max}^k$  :  $k$  个染色体  $v_k$  的最大流程时间,

FSP的适应度函数:

$$eval(v_k) = \frac{1}{c_{\max}^k}$$

# 基于遗传算法的流水车间调度方法

## 3. 求解FSP的遗传算法实例

例1 Ho 和 Chang(1991) 给出的5个工件、4台机器问题。

加工时间表

工件 $j$	$t_{j1}$	$t_{j2}$	$t_{j3}$	$t_{j4}$
1	31	41	25	30
2	19	55	3	34
3	23	42	27	6
4	13	22	14	13
5	33	5	57	19

# 基于遗传算法的流水车间调度方法

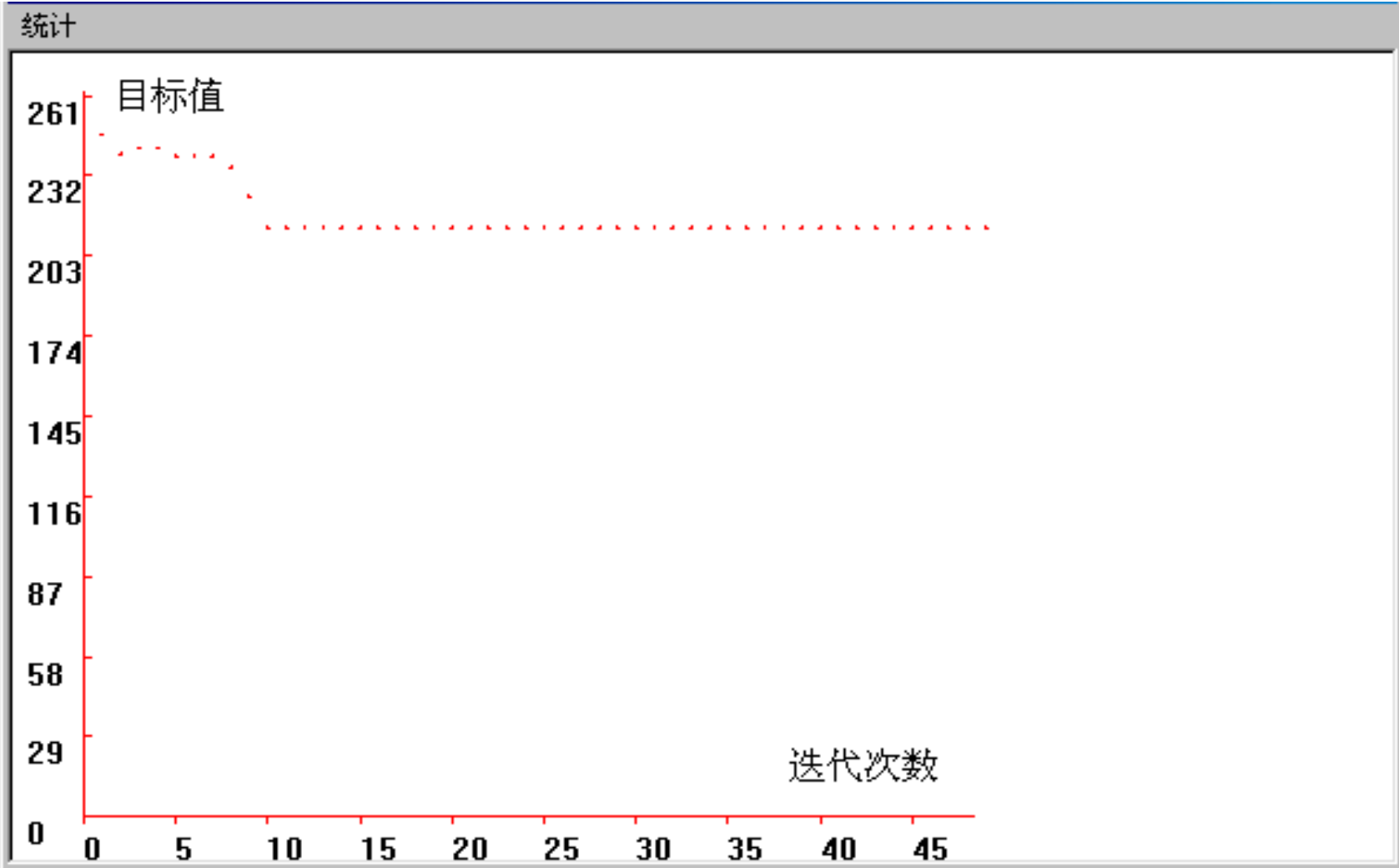
用穷举法求得最优解：4-2-5-1-3，加工时间：213；  
最劣解：1-4-2-3-5，加工时间：294；平均解的加工时间：265。

用遗传算法求解。选择交叉概率  $p_c = 0.6$ ，变异概  $p_m = 0.1$ ，种群规模为20，迭代次数  $N = 50$ 。

遗传算法运行的结果

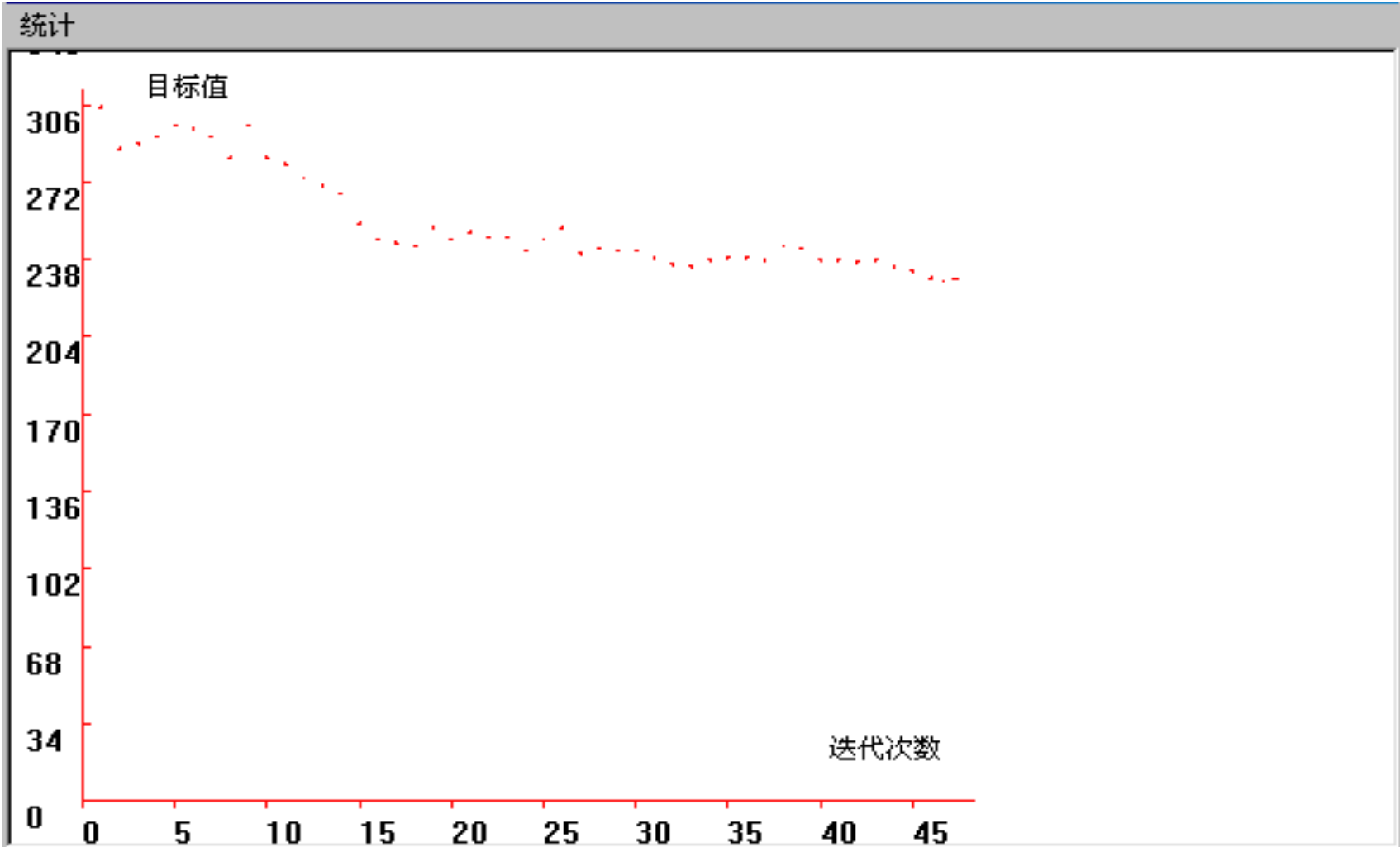
总运行 次数	最好解	最坏解	平均	最好解 的频率	最好解的 平均代数
20	213	221	213.95	0.85	12

# 基于遗传算法的流水车间调度方法



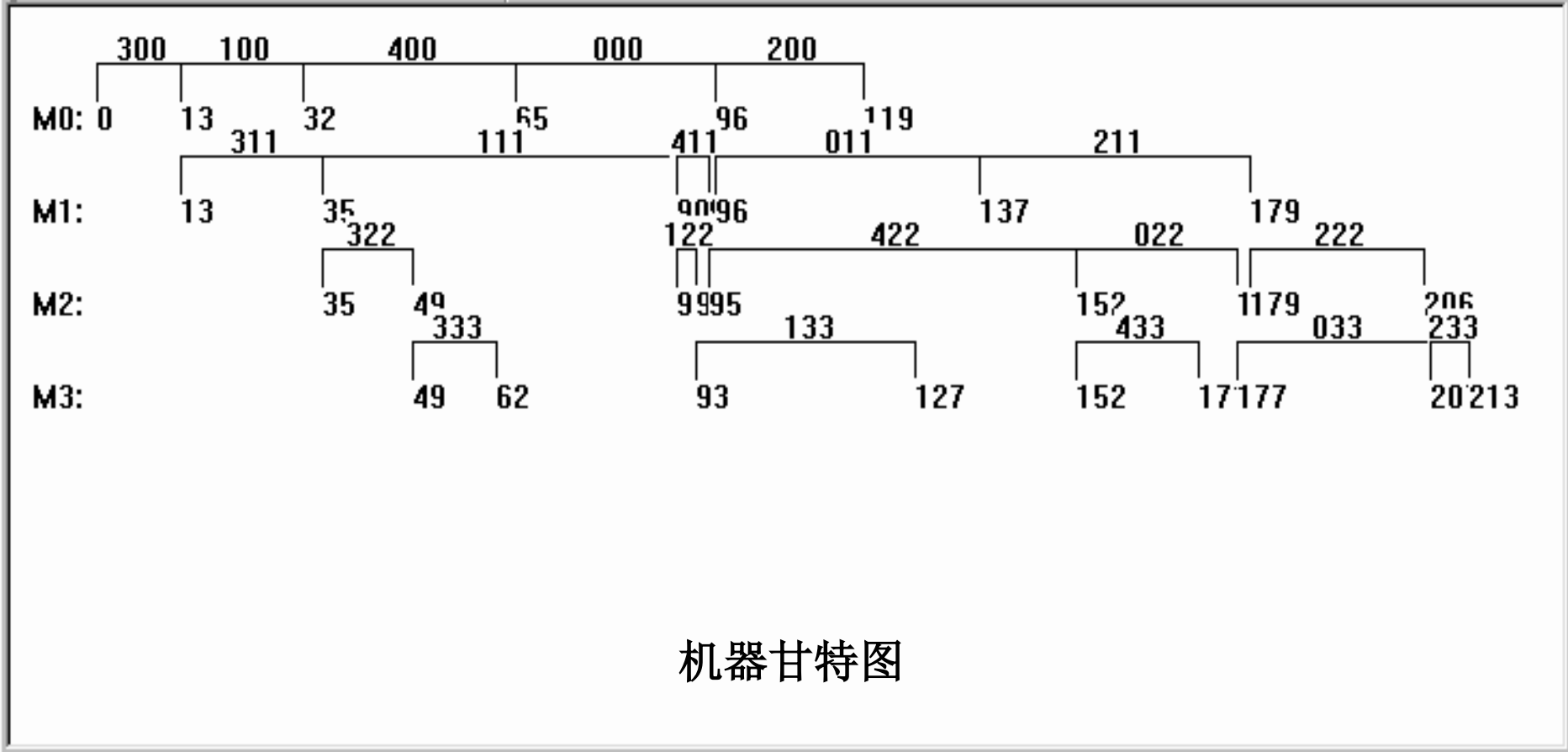
最优解收敛图

# 基于遗传算法的流水车间调度方法



平均值收敛图

# 基于遗传算法的流水车间调度方法





# 基于遗传算法的混合流水车间调度方法

## 1. 混合流水车间调度问题

- 问题的特征：在某些工序上存在并行机器。
- **问题的描述**：需要加工多个工件，所有工件的加工路线都相同，都需要依次通过几道工序，在所有工序中至少有一个工序存在着多台并行机器。
- 需要解决的问题：确定并行机器的分配情况以及同一台机器上工件的加工排序。
- 目标：最小化最大流程时间。

# 基于遗传算法的混合流水车间调度方法

## 2. 混合流水车间调度问题的遗传算法编码方法

■ 假设加工  $N$  个工件，每个工件都要依次经过  $S$  个加工工序，每个工序的并行机器数为  $M_i$ , ( $i=1, \dots, S$ )。所有工序中至少有一个工序存在并行机，即至少有一个  $M_i$  大于1。

### ■ HFSP的编码矩阵

$$A_{S \times N} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{S1} & a_{S2} & \dots & a_{SN} \end{bmatrix}$$

$a_{ij}$  :  $(1, M_i + 1)$  上的一个实数，表示  $j$  工件的第  $i$  个工序在第  $Int(a_{ij})$  台并行机上加工。

$Int(a_{ij}) = Int(a_{ik})$ ,  $j \neq k$  : 多个工件在同一台机器上加工同一个工序

# 基于遗传算法的混合流水车间调度方法

- $i = 1$  , 按  $a_{1j}$  的升序来加工工件。
- $i > 1$  , 根据每个工件的前一个工序的完成时间来确定其加工顺序, 前一个工序先完成的先加工。
- 假如完成时间相同, 也按  $a_{ij}$  的升序来加工。

## ■ 染色体:

$$Ind_k = [a_{11}, a_{12}, \dots, a_{1N}, 0, a_{21}, \dots, a_{2N}, 0, \dots, 0, a_{S1}, a_{S2}, \dots, a_{SN}]$$

染色体的长度:  $S \times N + S - 1$  。

# 基于遗传算法的混合流水车间调度方法

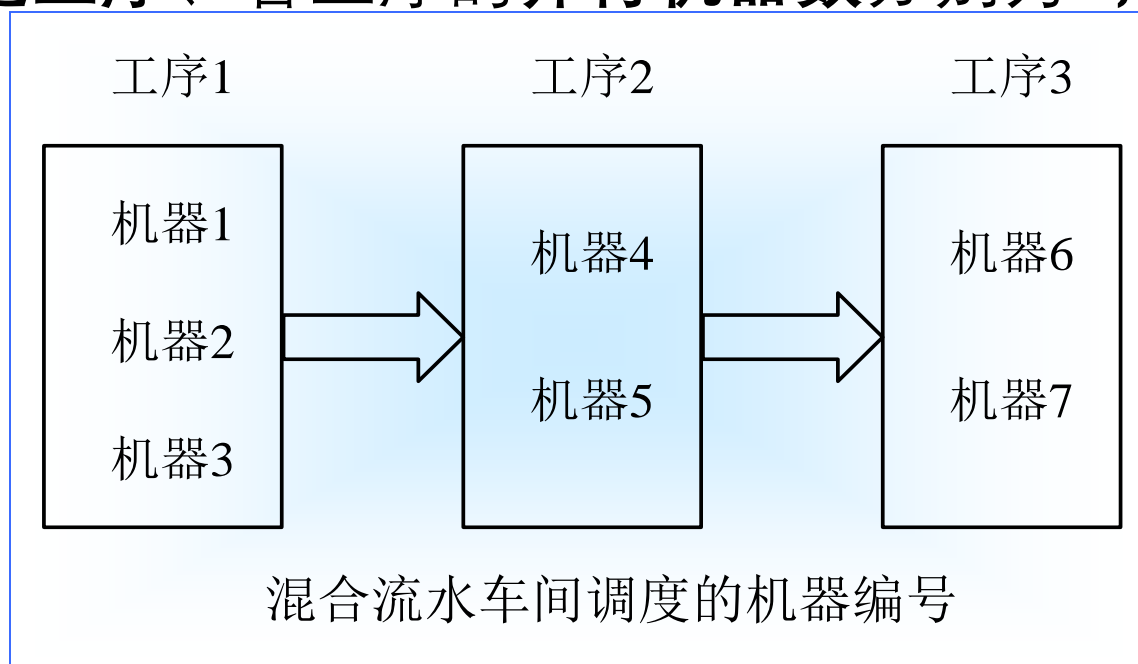
- 例如，对于有3个工件、3道工序、各工序的并行机器数分别为3，2，2的混合流水车间调度问题。

■ 编码矩阵:

$$A = \begin{bmatrix} 2.1 & 2.4 & 1.9 \\ 1.6 & 2.1 & 2.3 \\ 1.1 & 2.4 & 1.2 \end{bmatrix}$$

■ 染色体:

[2.1, 2.4, 1.9, 0, 1.6, 2.1, 2.3, 0, 1.1, 2.4, 1.2]



# 基于遗传算法的混合流水车间调度方法

## 3. 基于遗传算法的求解方法

- (1) 初始群体的产生
- (2) 适应度函数的选择：最大流程时间的倒数
- (3) 选择（非线性排名策略）

①种群成员按适应值从好到坏依次排列  $f_1 > f_2 > \dots > f_N$

②按下式分配复制概率：

$$p_i = \begin{cases} a(1-a)^{i-1} & i = 1, 2, \dots, N-1 \\ (1-a)^{N-1} & i = N \end{cases}$$

# 基于遗传算法的混合流水车间调度方法

## (4) 交叉操作

**分段交叉**：在两个父体的各段中随机选取一部分基因，然后交换，得到子代个体。

## (5) 变异操作：分段

(a)  $d = Rand\{-1,1\}$

(b) 若  $d = 1$  , 则  $r = Rand(0, M_i - a_{ij})$  , 否则  $r = Rand(0, a_{ij})$

(c)  $a'_{ij} = a_{ij} + d \times r$

# 基于遗传算法的混合流水车间调度方法

## 4. 调度实例

- 某汽车发动机厂金加工车间要加工12个工件，每个工件都有车、刨、磨 3个工序，现有3台车床，2台刨床，4台磨床，每台机床的加工能力不同。

# 基于遗传算法的混合流水车间调度方法

工件在每个机器上的加工时间

工件	工序1			工序2		工序3			
	机器1	机器2	机器3	机器4	机器5	机器6	机器7	机器8	机器9
1	2	2	3	4	5	2	3	2	3
2	4	5	4	3	4	3	4	5	4
3	6	5	4	4	2	3	4	2	5
4	4	3	4	6	5	3	6	5	8
5	4	5	3	3	1	3	4	6	5
6	6	5	4	2	3	4	3	9	5
7	5	2	4	4	6	3	4	3	5
8	3	5	4	7	5	3	3	6	4
9	2	5	4	1	2	7	8	6	5
10	3	6	4	3	4	4	8	6	7
11	5	2	4	3	5	6	7	6	5
12	6	5	4	5	4	3	4	7	5



# 基于遗传算法的混合流水车间调度方法

算法中使用的参数为 $\alpha = 0.07$ ,  $P_c = 0.80$ ,  $P_m = 0.01$ , 种群规模为30, 种群经过100代的进化, 目标函数最小值随着种群的进化逐渐地减小, 最后收敛于极值, 目标函数平均值也随着群体的进化逐渐减少, 最后趋近于最优值。

最好的染色体:

[2.77, 3.51, 1.74, 3.52, 2.42, 1.36, 3.28, 3.94, 1.09,  
1.22, 2.24, 3.64, 0, 1.60, 1.13, 1.24, 2.97, 1.73, 1.88,  
1.08, 2.68, 1.16, 2.69, 2.51, 2.96, 0, 4.99, 3.29, 4.95,  
2.35, 1.10, 1.01, 1.73, 1.35, 3.06, 1.20, 4.13, 3.67 ]

# 遗传算法中的遗传算子

遗传算法包括3个遗传算子(Genetic operator), 分别为:

- 选择算子(Selection operator)
- 交叉算子(Crossover operator)
- 变异算子(Mutation operator)

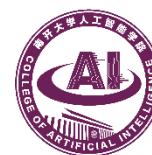
# 人工智能技术实验

## Artificial Intelligence Technology Experiments

许丽，实验师

南开大学人工智能学院

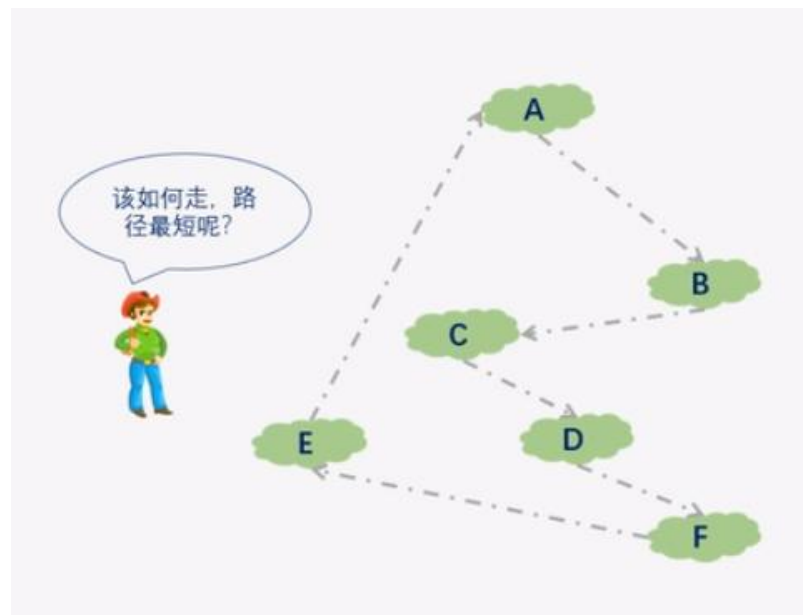
xuli@nankai.edu.cn



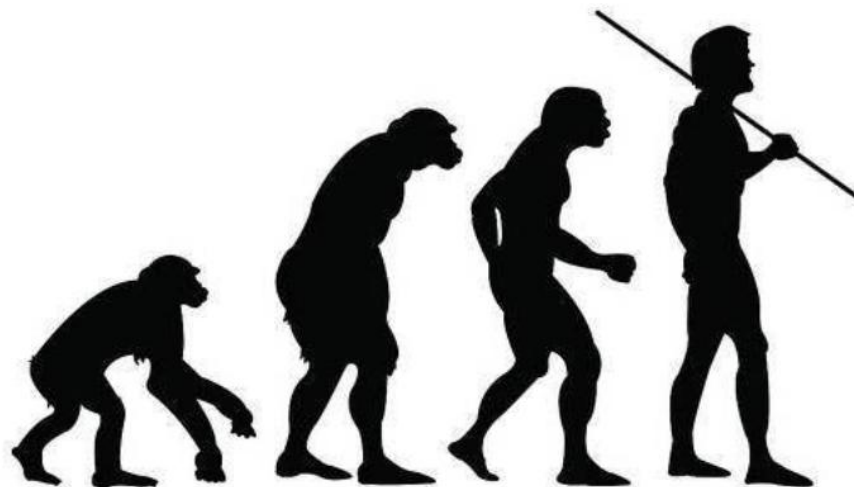
## L4 旅行商问题 (遗传算法)

# 旅行商问题

- 给定一系列城市 and 每对城市之间的距离，求解访问每一座城市一次并回到起始城市的最短回路。
- 旅行商问题(TSP)是一个组合优化方面的问题，已经成为测试组合优化新算法的标准问题。（NP难题）
- 辨析：最短路径问题
- 蛮力：穷举 $O(n!)$



- 遗传算法（Genetic Algorithm, GA）是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，通过模拟自然进化过程搜索**最优解**的方法：适者生存、优胜劣汰
- 遗传算法以一种群体中的所有个体为对象，并利用随机化技术指导对一个被编码的参数空间进行高效搜索。



- 五个要素：参数编码、初始群体的设定、适应度函数的设计、遗传操作设计、控制参数设定
- 编码（可行解）：创造染色体，决定染色体长度，二进制，浮点等；
- 初始群体：种群大小；
- 适应度函数（评价函数）：目标函数值；
- 遗传操作设计：是否选择精英操作、最大迭代次数等
- 控制参数设定：交叉概率、变异概率等

# 遗传算法

- 遗传操作：选择、交叉和变异

选择：轮盘赌选择，联赛选择等；

交叉：一点交叉，两点交叉等；

变异：基本位变异，均匀变异等





- 特点：
  - 直接对结构对象进行操作，不存在求导和函数连续性的限定；
  - 采用概率化的寻优方法，不需要确定的规则就能自动获取和指导优化的搜索空间，自适应地调整搜索方向。
  - 群体性操作，内在的隐并行性和全局寻优能力
- 不一定是最优解。

- $GA(Fitness, Fitness\_threshold, p, r, m)$
- $Fitness$ : 适应度函数
- $Fitness\_threshold$ : 终止判据的阈值
- $p$ : 群体数量
- $r$ : 交叉概率
- $m$ : 变异概率
- 初始化群体:  $P$  (随机产生)
- 评估: 对于  $P$  中的每个  $h$ , 计算  $Fitness(h)$
- 当  $\max Fitness(h) < Fitness\_threshold$ , 产生新一代的  $P_s$ :
- 1. 选择: 用概率方法选择  $P$  的  $(1-r)p$  个成员加入  $P_s$
- 2. 交叉:  $P$  中按概率选择  $r(p/2)$  对假设, 对于每对假设  $\langle h_1, h_2 \rangle$  进行交叉操作, 后代加入  $P_s$
- 3. 变异: 随机从  $P_s$  中选择  $m\%$  的成员进行变异操作
- 4. 更新:  $P \leftarrow P_s$
- 5. 评估: 对于  $P$  中的每个  $h$  计算  $Fitness(h)$
- 从  $P$  中返回适应度最高的结果

# 1.初始化种群

- 确定个体编码方式，即为使用节点编号组成的TSP路径表示  
(1-2-3-4-5-6-7-8) 表示为 (1,2,3,4,5,6,7,8) (实际是loop)
- N个城市，每个个体长度为N，用s行，t列的矩阵表示初始群体，s表示初始群体的个数，t为N
- 城市的位置信息转换成距离矩阵： $N \times N$ 矩阵D存储， $D(i,j)$ 代表城市i和城市j之间的距离
- 初始随机解的生成：随机产生，路径中不能有重复节点

## 2.适应度函数

- 影响收敛性和收敛速度
- n个城市:

$$S = n_1, n_2, \dots, n_n, n_1$$

- 求出基因序列所表达的巡回路线长度:

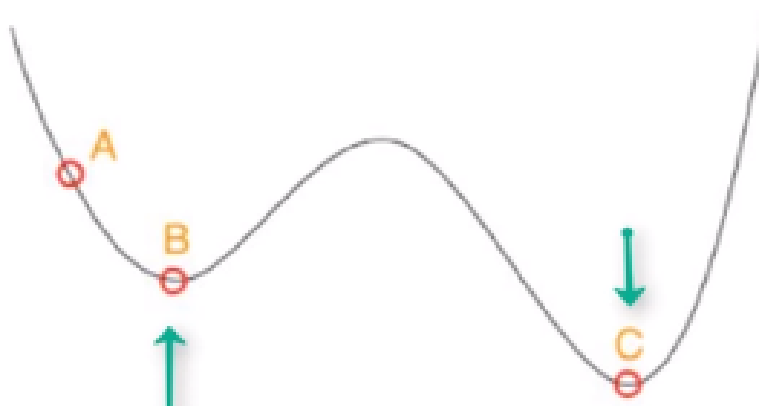
$$d(s) = \sum_{j=1}^n d(n_j, n_{j+1})$$

- 目标是路线长度较短的物种生存下来, 采取倒数的方法来处理, 得到适应度函数:

$$f(s) = \frac{1}{d(s)}$$

### 3.选择操作

- 选择概率：适应度大个体存在机会大，适应度小个体存在机会小
- 精英保留策略：适应度高的物种 问题：局部最优 个数选择
- 轮盘赌选择法



### 3.选择操作-轮盘赌选择法

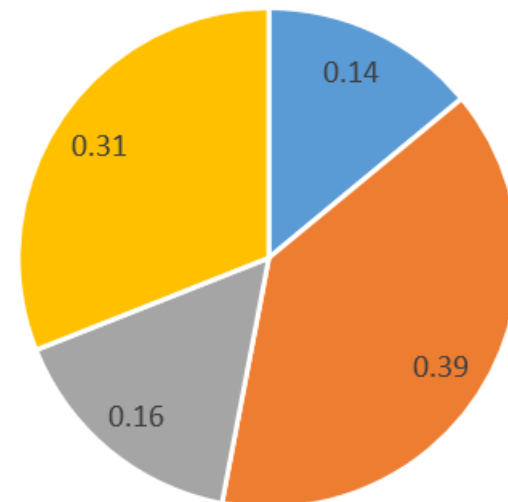
- 1.计算出群体中每个个体的适应度
- 2.计算出每个个体被遗传到下一代群体的概率:

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

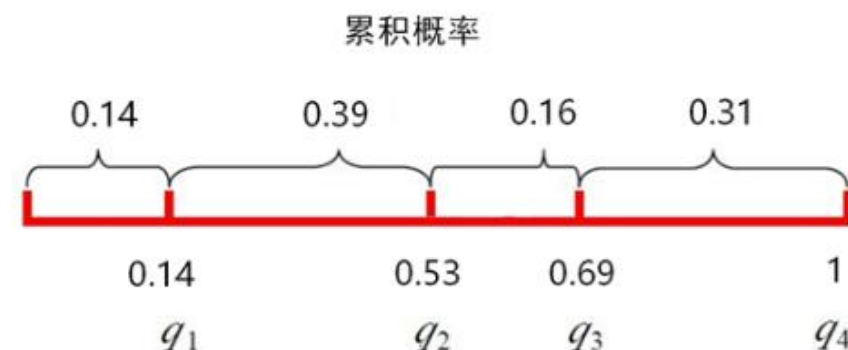
- 3.计算出每个个体的累积概率

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

- 4.选择过程: 0-1之间均匀分布的伪随机数 $f$ , 若 $q_a < f \leq q_b$ , 则个体b被选中, 重复执行

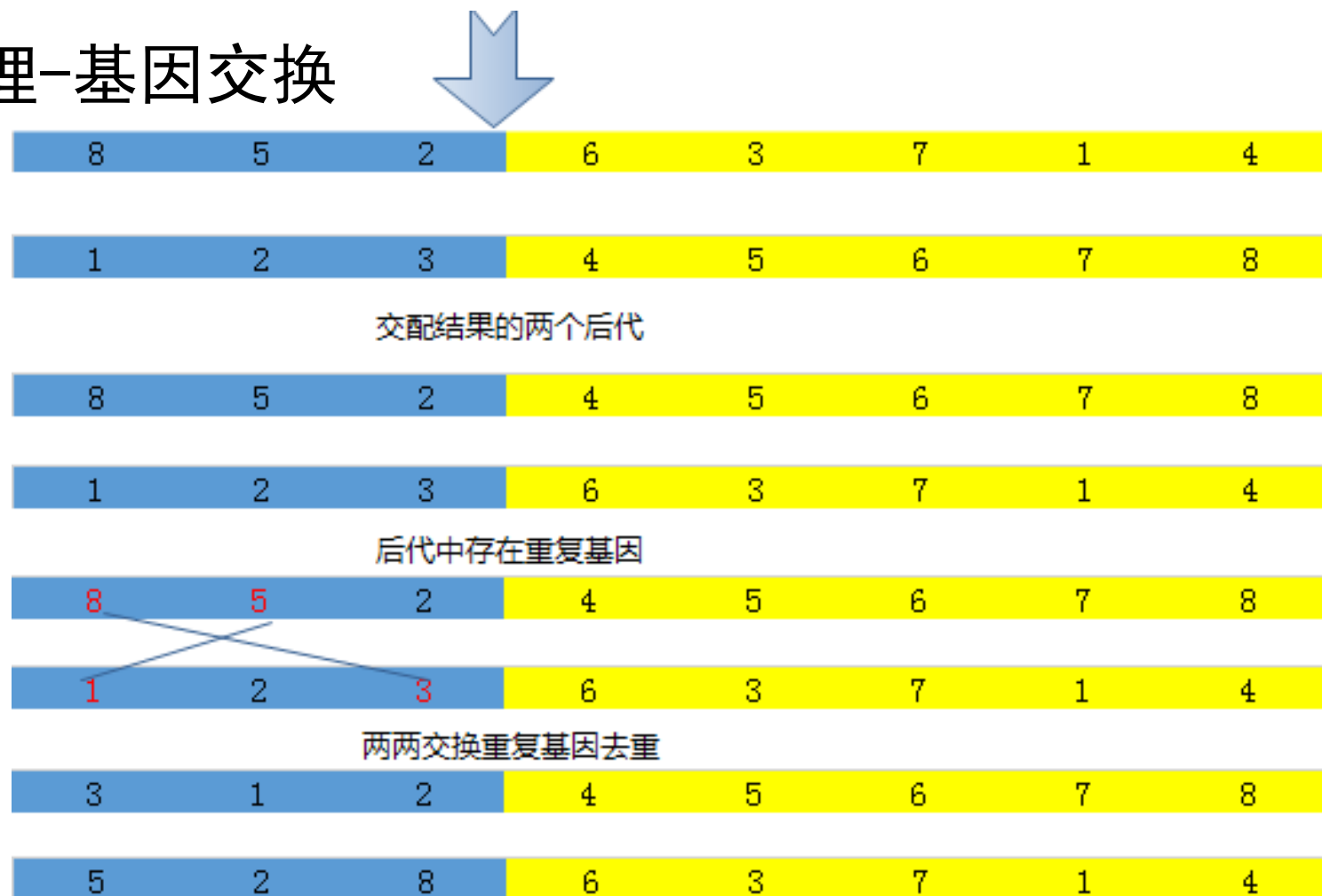


■ 个体1 ■ 个体2 ■ 个体3 ■ 个体4



# 4.交叉操作

- 随机选取、部分交叉、交叉概率
- 基因重复-基因冲突处理-基因交换
- 单点交换:



## 5.变异操作

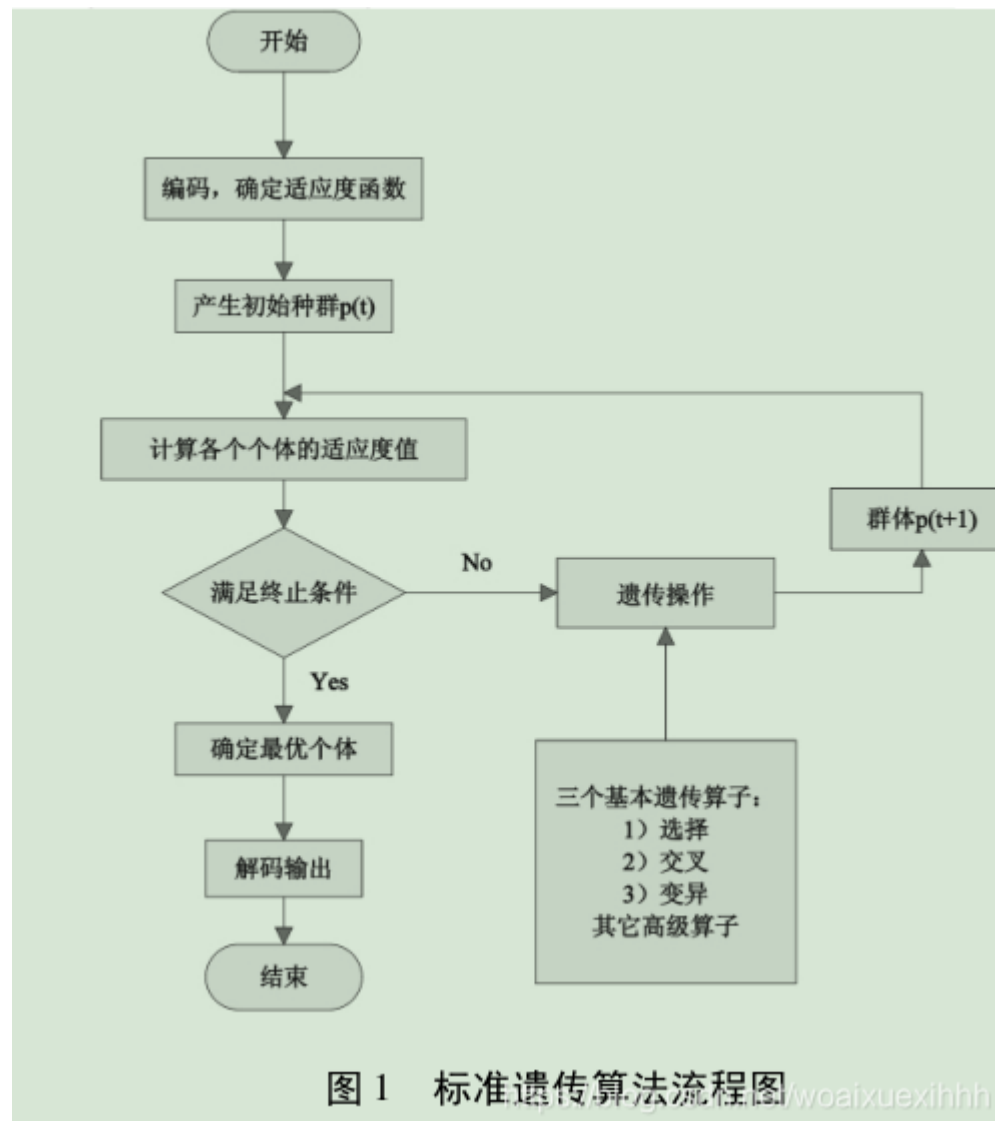
- 多样性，限于局部解的弊病，变异概率
- 二进制编码的变异不适用
- 随机抽取两个城市，交互位置
- 移动，倒序等
- 更新
- 终止条件：循环一定的代数，选出适应度最优的个体作为最终解（解码）



# 关心的数据

- 地图数据
- 编码方式 (染色体长度)
- 适应度评价函数 (曲线)
- 初始群体
- 种群大小 20-160
- 进化代数 (终止条件) 100-1000 饱和
- 交叉概率 0.5-1.0
- 变异概率 0.001-0.1
- 父代保留数量 1/4

# 算法流程图



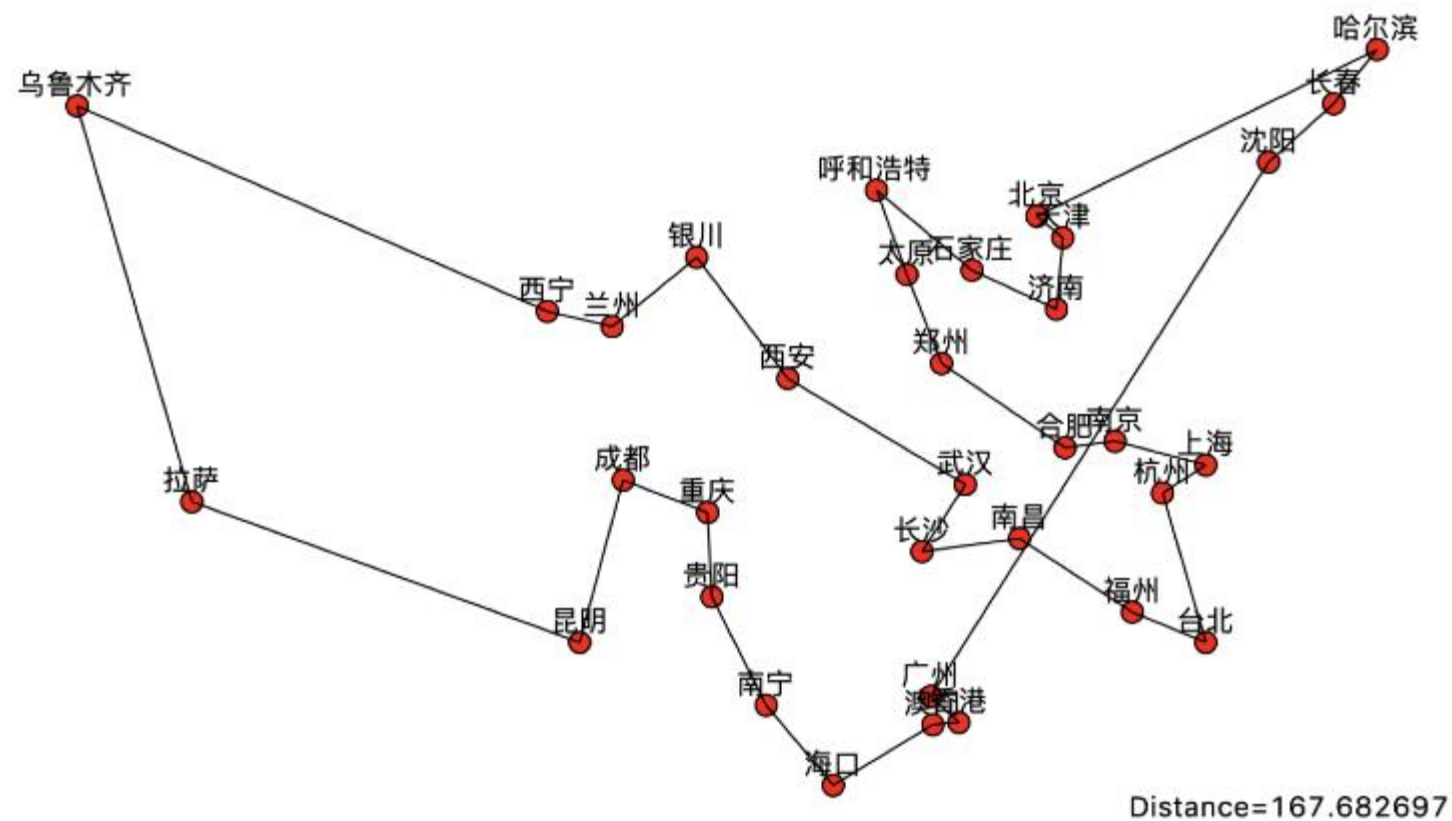
# 地图数据

- 34个城市（23个省会、4个直辖市、5个自治区及2个特别行政区）
- 欧氏距离

	1	2	3	4	5	6	7
种群规模	80	80	80	100	100	100	100
迭代次数	100	100	500	500	饱 和	饱 和	饱 和
交叉概率	0.6	0.6	0.6	0.6	0.65	0.65	0.7
突变概率	0.02	0.03	0.03	0.03	0.035	0.03	0.04
最小开销	271.83	248.33	188.88	182.00	167.68	177.41	188.14

1	北京	116.46	39.92
2	天津	117.2	39.13
3	上海	121.48	31.22
4	重庆	106.54	29.59
5	拉萨	91.11	29.97
6	乌鲁木齐	87.68	43.77
7	银川	106.27	38.47
8	呼和浩特	111.65	40.82
9	南宁	108.33	22.84
10	哈尔滨	126.63	45.75
11	长春	125.35	43.88
12	沈阳	123.38	41.8
13	石家庄	114.48	38.03
14	太原	112.53	37.87
15	西宁	101.74	36.56
16	济南	117	36.65
17	郑州	113.6	34.76
18	南京	118.78	32.04
19	合肥	117.27	31.86
20	杭州	120.19	30.26
21	福州	119.3	26.08
22	南昌	115.89	28.68
23	长沙	113	28.21
24	武汉	114.31	30.52
25	广州	113.23	23.16
26	台北	121.5	25.05
27	海口	110.35	20.02
28	兰州	103.73	36.03
29	西安	108.95	34.27
30	成都	104.06	30.67
31	贵阳	106.71	26.57
32	昆明	102.73	25.04
33	香港	114.1	22.2
34	澳门	113.33	22.13

# 输出的一种解



# 算法函数构成

- Generate[p(n)]; 初始群体, 个数n
- Evaluate[p(h)]; h是种群中的一个个体, 计算每个个体的适应度
- Repeat roof generations times;
- Selectbest(); 适应度高的个体, 直接复制到下一代
- Select p(h) from p(n-1);
- Crossover p(n);
- mutation p(n);
- Evaluate[p(h)] ;
- End;

# 旅行商问题

- 基本要求：实现基于遗传算法的34个城市的旅行商问题
- 选做内容：不同种群规模、交叉概率、变异概率下的结果对比，分析原因，可视化（路径、适应度函数）
- 提交时间：三周
- 提交方式：在线提交（\*实验报告、\*源代码、demo等）
- 迟交问题：扣分
- 抄袭问题：不及格处理！

Q&A

THANKS!

