智能搜索（优化）算法

优化问题

- 很多实际问题可建模成优化问题
- 非常多的人工智能问题，最后都归结为优化问题
- 优化问题的分界线应该是凸与非凸，而不是线性与非线性

凸集和凸函数

凸集:

$S \subset R^n$.称S是凸集，如果对任意$x_1, x_2 \subset S$和任意的$\lambda \in [0, 1]$

都有　　$\lambda x_1 + (1-\lambda) x_2 \in S.$

凸函数:

$S \subset R^n$.称S是非空凸集, f是定义在 S 上的函数．称函数f是凸函数，

如果对任意$x_1, x_2 \subset S$和任意的$\lambda \in [0, 1]$

都有　　$f(\lambda x_1 + (1-\lambda) x_2) < \lambda f(x_1) + (1-\lambda) f(x_2).$

凸函数的极小：

若问题有局部极小，则这个局部解是全局最小.

凸优化和非凸优化

The general form of an *optimization problem* (also referred to as a *mathematical programming problem* or *minimization problem*) is to find some $x^* \in \mathcal{X}$ such that

$$f(x^*) = \min\{f(x) : x \in \mathcal{X}\},$$

for some feasible set $\mathcal{X} \subset \mathbb{R}^n$ and objective function $f(x) : \mathbb{R}^n \to \mathbb{R}$. The optimization problem is called a convex optimization problem if $\mathcal{X}$ is a convex set and $f(x)$ is a convex function defined on $\mathbb{R}^n$.

Alternatively, an optimization problem of the form

$$\begin{aligned} &\textbf{minimize} && f(x) \\ &\textbf{subject to} && g_i(x) \leq 0, \quad i = 1, \ldots, m \end{aligned}$$

is called convex if the functions $f, g_1 \ldots g_m : \mathbb{R}^n \to \mathbb{R}$ are all convex functions.

- 约束优化问题常转化为无约束优化问题求解
- 凸优化有很好的确定性解法，可参见 Stephen Boyd and Lieven Vandenberghe，*Convex Optimization*，Cambridge University Press
- 非凸优化可采用一些启发式算法求解

4.1.4 进化计算

进化计算（Evolutionary Computation,EC）是在达尔文（Darwin）的进化论和孟德尔（Mendel）的遗传变异理论的基础上产生的一种模拟自然界生物进化过程与机制的问题求解技术。它主要包括

遗传算法（Genetic Algorithm，GA）

进化策略（Evolutionary Strategy,ES）

进化规划（Evolutionary Programming,EP）

遗传规划（Genetic Programming,GP）

四大分支。

其中，第一个分支是进化计算中最初形成的一种具有普遍影响的模拟进化优化算法。因此我们主要讨论遗传算法。

4.1.4 进化计算概述

(1) 什么是进化计算

进化计算是一种模拟自然界生物进化过程与机制进行问题求解的随机搜索技术。它以达尔文进化论的“物竞天择、适者生存”作为算法的进化规则，并结合孟德尔的遗传变异理论，将生物进化过程中的

繁殖（Reproduction）

变异（Mutation）

竞争（Competition）

选择（Selection）

引入到了算法中。

4.1.4 进化计算概述

(2) 进化计算的生物学基础

自然界生物进化过程是进化计算的生物学基础，它主要包括遗传(Heredity)、变异(Mutation)和进化(Evolution)理论。

① 遗传理论

遗传是指父代利用遗传基因将自身的基因信息传递给下一代，使子代能够继承其父代的特征或性状的这种生命现象。

在自然界，构成生物基本结构与功能的单位是细胞（Cell）。

细胞中含有一种包含着所有遗传信息的复杂而又微小的丝状化合物，人们称其为染色体（Chromosome）。

在染色体中，遗传信息由基因（Gene）所组成，基因决定着生物的性状，是遗传的基本单位。

染色体的形状是一种双螺旋结构，构成染色体的主要物质叫做脱氧核糖核酸(DNA)，每个基因都在DNA长链中占有一定的位置。

一个细胞中的所有染色体所携带的遗传信息的全体称为一个基因组。

4.1.4 进化计算概述

② 变异理论

变异是指子代和父代之间，以及子代的各个不同个体之间产生差异的现象。变异是一种随机、不可逆现象，是生物多样性的基础。

引起变异的主要原因：

杂交，是指有性生殖生物在繁殖下一代时两个同源染色体之间的交配重组。

复制差错，是指在细胞复制过程中因DNA上某些基因结构的随机改变而产生出新的染色体。

4.1.4 进化计算概述

③ 进化论

进化是指在生物延续生存过程中，逐渐适应其生存环境，使得其品质不断得到改良的这种生命现象。遗传和变异是生物进化的两种基本现象，优胜劣汰、适者生存是生物进化的基本规律。

达尔文的自然选择学说：在生物进化中，一种基因有可能发生变异而产生出另一种新的基因。这种新基因将依据其与生存环境的适应性而决定其增殖能力。通常，适应性强的基因会不断增多，而适应性差的基因则会逐渐减少。通过这种自然选择，物种将逐渐向适应于生存环境的方向进化，甚至会演变成为另一个新的物种，而那些不适应于环境的物种将会逐渐被淘汰。

4.1.4 进化计算概述

2. 进化计算的产生与发展

进化计算自20世纪50年代以来，其发展过程大致可分为三个阶段。

(1) 萌芽阶段

这一阶段是从20世纪50年代后期到70年代中期。20世纪50年代后期，一些生物学家在研究如何用计算机模拟生物遗传系统中，产生了遗传算法的基本思想，并于1962年由美国密执安（Michigan）大学霍兰德（Holland）提出。1965年德国数学家雷切伯格（Rechenberg）等人提出了一种只有单个个体参与进化，并且仅有变异这一种进化操作的进化策略。同年，美国学者弗格尔（Fogel）提出了一种具有多个个体和仅有变异一种进化操作的进化规划。1969年美国密执安（Michigan）大学的霍兰德（Holland）提出了系统本身和外部环境相互协调的遗传算法。至此，进化计算的三大分支基本形成。

4.1.4 进化计算概述

2. 进化计算的产生与发展

(2) 成长阶段

这一阶段是从20世纪70年代中期到80年代后期。1975年，霍兰德出版专著《自然和人工系统的适应性（Adaptation in Natural and Artificial System）》，全面介绍了遗传算法。同年，德国学者施韦费尔（Schwefel）在其博士论文中提出了一种由多个个体组成的群体参与进化的，并且包括了变异和重组这两种进化操作的进化策略。1989年，霍兰德的学生戈尔德伯格（Goldberg）博士出版专著《遗传算法----搜索、优化及机器学习（Genetic Algorithm in Search Optimization and Machine Learning）》，使遗传算法得到了普及与推广。

4.1.4 进化计算概述

2. 进化计算的产生与发展

(3) 发展阶段

这一阶段是从20世纪90年代至今。1989年，美国斯坦福（Stanford）大学的科扎（Koza）提出了遗传规划的新概念，并于1992年出版了专著《遗传规划----应用自然选择法则的计算机程序设计（Genetic Programming : On the Programming of Computer by Means of Natural Selection）》该书全面介绍了遗传规划的基本原理及应用实例，标志着遗传规划作为进化计算的一个分支已基本形成。

进入20世纪90年代以来，进化计算得到了众多研究机构和学者的高度重视，新的研究成果不断出现、应用领域不断扩大。

4.1.4 进化计算概述

3. 进化计算的基本过程

进化计算尽管有多个重要分支，但它们却有着共同的进化框架。

若假设P为种群(Population，或称为群体)，t为进化代数， P(t)为第t代种群，则进化计算的基本结构可粗略描述如下：

```
{ 确定编码形式并生成搜索空间；
  初始化各个进化参数，并设置进化代数t=0；
  初始化种群P(0);
  对初始种群进行评价（即适应度计算）；
  while（不满足终止条件）do
     {
     t=t+1;
     利用选择操作从P(t-1)代中选出P(t)代群体；
     对P(t)代种群执行进化操作；
     对执行完进化操作后的种群进行评价（即适应度计算）；
     }
}
```

可以看出，上述基本结构包含了生物进化中所必需的选择操作、进化操作和适应度评价等过程。

4.5 遗传算法

4.5.1 遗传算法的基本概念

遗传算法的基本思想是从初始种群出发，采用优胜劣汰、适者生存的自然法则选择个体，并通过杂交、变异来产生新一代种群，如此逐代进化，直到满足目标。遗传算法涉及的基本概念主要有以下几个：

种群（Population）：种群是指用遗传算法求解问题时，给定的多个解的集合。种群由个体组成。

个体（Individual）

染色体（Chromos）：染色体是指对个体进行编码后所得到的编码串，其中的每1位称为基因。

适应度（Fitness）函数：适应度函数是一种用来对种群中各个个体的环境适应性进行度量的函数。

遗传操作（Genetic Operator）：遗传操作是指作用于种群而产生新的种群的操作。标准的遗传操作包括以下3种基本形式：

选择（Selection）

交叉（Crosssover）

变异（Mutation）

4.5 遗传算法

4.5.2 遗传算法的基本过程

遗传算法主要由染色体编码、初始种群设定、适应度函数设定、遗传操作设计等几大部分所组成，其算法主要内容和基本步骤可描述如下：

(1) 选择编码策略，将问题搜索空间中每个可能的点用相应的编码策略表示出来，即形成染色体；

(2) 定义遗传策略，包括种群规模N，交叉、变异方法，以及选择概率P_r、交叉概率P_c、变异概率P_m等遗传参数；

(3) 令t=0，随机选择N个染色体初始化种群P(0)；

(4) 定义适应度函数f（f>0）；

(5) 计算P(t)中每个染色体的适应值；

(6) t=t+1；

(7) 运用选择算子，从P(t-1)中得到P(t)；

(8) 对P(t)中的每个染色体，按概率P_c参与交叉；

(9) 对染色体中的基因，以概率P_m参与变异运算；

(10) 判断群体性能是否满足预先设定的终止标准，若不满足则返回(5)。

4.5 遗传算法

遗传算法流程如图4-18所示：

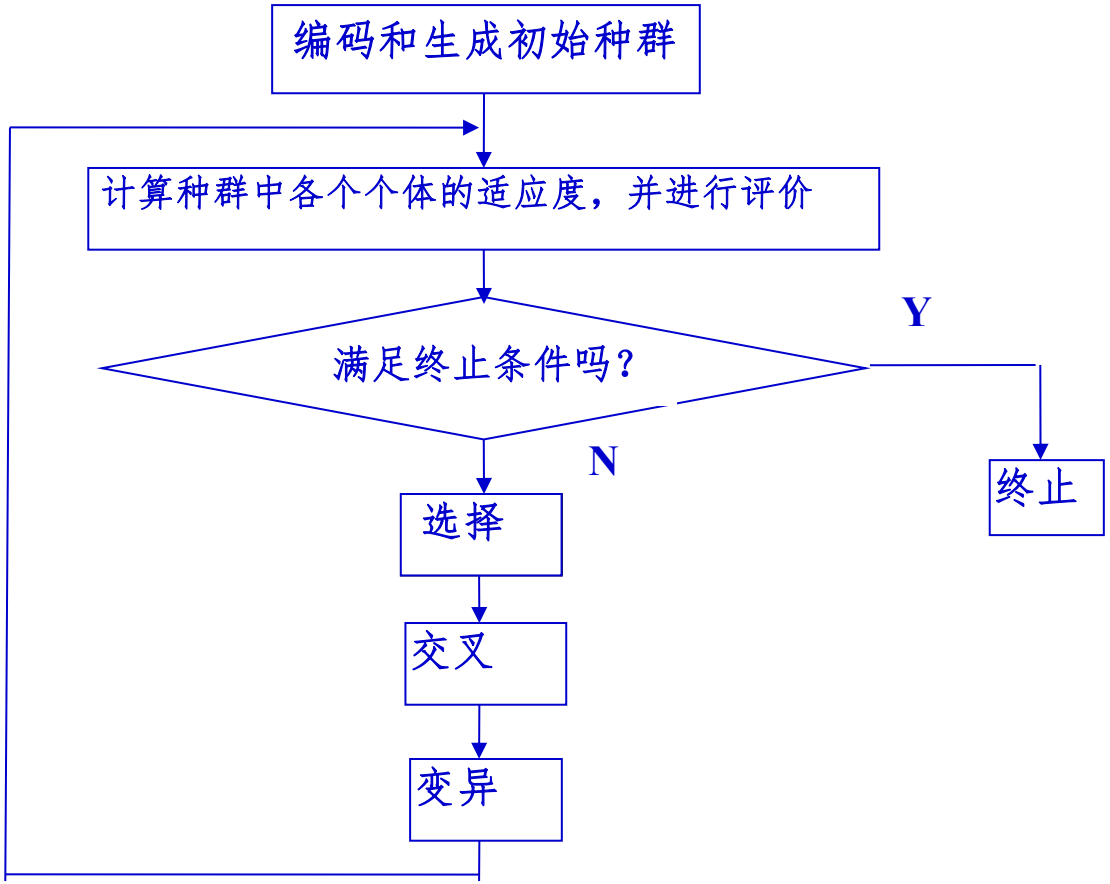


图4-18 基本遗传算法的算法流程图

4.5 遗传算法

4.5.3 遗传编码

常用的遗传编码算法有

(1)二进制编码（Binary encoding）

二进制编码是将原问题的结构变换为染色体的位串结构。在二进制编码中，首先要确定二进制字符串的长度l，该长度与变量的定义域和所求问题的计算精度有关。

例4.10 假设变量x的定义域为[5，10]，要求的计算精度为10^{-5}，则需要将[5，10]至少分为600000个等长小区间，每个小区间用一个二进制串表示。于是，串长至少等于20，原因是：

$$524288=2^{19}<600000<2^{20}=1048576$$

这样，对应于区间[5，10]内满足精度要求的每个值x，都可用一个20位编码的二进制串$<b_{19},b_{18},\ldots,b_0>$来表示。

二进制编码存在的主要缺点是汉明（Hamming）悬崖。

例如，7和8的二进制数分别为0111和1000，当算法从7改进到8时，就必须改变所有的位。

4.5 遗传算法

4.5.3 遗传编码

(2) 格雷编码（Gray encoding）

格雷编码是对二进制编码进行变换后所得到的一种编码方法。这种编码方法要求两个连续整数的编码之间只能有一个码位不同，其余码位都是完全相同的。它有效地解决了汉明悬崖问题.其基本原理如下：

设有二进制串$\mathbf{b_1,b_2,\ldots,b_n}$，对应的格雷串为$\mathbf{a_1,a_2,\ldots,a_n}$，则从二进制编码到格雷编码的变换为：

$$a_i = \begin{cases} b_i & i = 1 \\ b_{i-1} \oplus b_i & i > 1 \end{cases} \qquad \textbf{(4.9)}$$

其中，$\oplus$ 表示模2加法。而从一个格雷串到二进制串的变换为：

$$b_i = \sum_{j=1}^{i} a_i \pmod 2 \qquad \textbf{(4.10)}$$

例4.11 十进制数7和8的二进制编码分别为0111和1000，而其格雷编码分别为0100和1100。

4.5 遗传算法

4.5.3 遗传编码

(3) 实数编码（Real encoding）

实数编码是将每个个体的染色体都用某一范围的一个实数（浮点数）来表示，其编码长度等于该问题变量的个数。

这种编码方法是将问题的解空间映射到实数空间上，然后在实数空间上进行遗传操作。由于实数编码使用的是变量的真实值，因此这种编码方法也叫做真值编码方法。

实数编码适应于那种多维、高精度要求的连续函数优化问题。

4.5 遗传算法

4.5.4 适应度函数

适应度函数是一个用于对个体的适应性进行度量的函数。通常，一个个体的适应度值越大，它被遗传到下一代种群中的概率也就越大。

(1) 常用的适应度函数

在遗传算法中，有许多计算适应度的方法，其中最常用的适应度函数有以下两种：

① 原始适应度函数

它是直接将待求解问题的目标函数**f(x)**定义为遗传算法的适应度函数。例如，在求解极值问题

$$\max_{x\in[a,b]} f(x)$$

时，**f(x)**即为**x**的原始适应度函数。

采用原始适应度函数的优点是能够直接反映出待求解问题的最初求解目标，其缺点是有可能出现适应度值为负的情况。

4.5 遗传算法

4.5.4 适应度函数

② 标准适应度函数

在遗传算法中，一般要求适应度函数非负，这就往往需要对原始适应函数进行某种变换，将其转换为标准的度量方式，以满足进化操作的要求，这样所得到的适应度函数被称为标准适应度函数$\mathbf{f_{normal}(x)}$。例如下面的极小化和极大化问题：

极小化问题

对极小化问题，其标准适应度函数可定义为

$$f_{normal}(x)=\begin{cases} f_{\max}(x)-f(x) & \text{当} f(x)<f_{\max}(x) \\ 0 & \text{否则} \end{cases} \tag{4.11}$$

其中，$\mathbf{f_{max}(x)}$是原始适应函数**f(x)**的一个上界。如果$\mathbf{f_{max}(x)}$未知，则可用当前代或到目前为止各演化代中的**f(x)**的最大值来代替。可见，$\mathbf{f_{max}(x)}$是会随着进化代数的增加而不断变化的。

4.5 遗传算法

4.5.4 适应度函数

极大化问题

对极大化问题，其标准适应度函数可定义为

$$f_{nomal}(x)=\begin{cases} f(x)-f_{\min}(x) & \text{当} f(x)>f_{\min}(x) \\ 0 & \text{否则} \end{cases} \tag{4.12}$$

其中，$f_{min}(x)$是原始适应函数$f(x)$的一个下界。如果$f_{min}(x)$ 未知，则可用当前代或到目前为止各演化代中的$f(x)$的最小值来代替。

4.5 遗传算法

4.5.4 适应度函数

(2) 适应度函数的加速变换

在某些情况下，如极值点附近，需要对适应度函数进行加速变换。

适应度函数的加速变换有两种基本方法

线性加速的适应度函数的定义如下：

$$f'(x)=\alpha f(x)+\beta$$

其中，f(x)是加速转换前的适应度函数； f'(x)是加速转换后的适应度函数；α和β是转换系数,它们应满足如下条件：

① 变化后得到的新的适应度函数平均值要等于原适应度函数的平均值。即

$$\alpha \bullet \frac{\sum_{i=1}^{n} f(x_i)}{n} + \beta = \frac{\sum_{i=1}^{n} f(x_i)}{n} \qquad (4.13)$$

其中，$x_i(i=1,\ldots,n)$为当前代中的染色体。

4.5 遗传算法

4.5.4 适应度函数

② 变换后所得到的新的种群个体所具有的最大适应度要等于其平均适应度的指定倍数。即有关系：

$$\alpha \bullet \max_{1\le i\le n}\{f(x_i)\}+\beta = M \bullet \frac{\sum_{i=1}^{n} f(x_i)}{n} \qquad (4.14)$$

式中，$x_i(i=1,\ldots,n)$ 为当前代中的染色体，M是指将当前的最大适应度放大为平均值的M倍。目的是通过M拉开不同染色体适应度值的差距。

非线性加速

幂函数变换方法

$$f'(x)=f(x)^k \qquad (4.15)$$

指数变换方法

$$f'(x)=\exp(-\beta f(x)) \qquad (4.16)$$

4.5 遗传算法

4.5.5 基本遗传操作

遗传算法中的基本遗传操作包括选择、交叉和变异3种，而每种操作又包括多种不同的方法，下面分别对它们进行介绍。

(1). 选择操作

选择（Selection）操作是指根据选择概率按某种策略从当前种群中挑选出一定数目的个体，使它们能够有更多的机会被遗传到下一代中。

常用的选择策略可分为比例选择、排序选择和竞技选择三种类型。

① 比例选择

比例选择方法（Proportional Model）的基本思想是：各个个体被选中的概率与其适应度大小成正比。

4.5 遗传算法

4.5.5 基本遗传操作

轮盘赌选择

在这种方法中，个体被选中的概率取决于该个体的相对适应度：

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

其中，$P(x_i)$是个体x_i被选中的概率；$f(x_i)$是个体x_i的原始适应度。

轮盘赌选择算法的基本思想是：根据每个个体的选择概率$P(x_i)$将一个圆盘分成N个扇区，其中第i个扇区的中心角为：

$$2\pi\frac{f(x_i)}{\sum_{j=1}^{N} f_i(x_j)}=2\pi\,p(x_i)$$

再设立一个移动指针，将圆盘的转动等价为指针的移动。选择时，假想转动圆盘，若静止时指针指向第i个扇区，则选择个体i。其物理意义如图5-19所示。

4.5 遗传算法

4.5.5 基本遗传操作

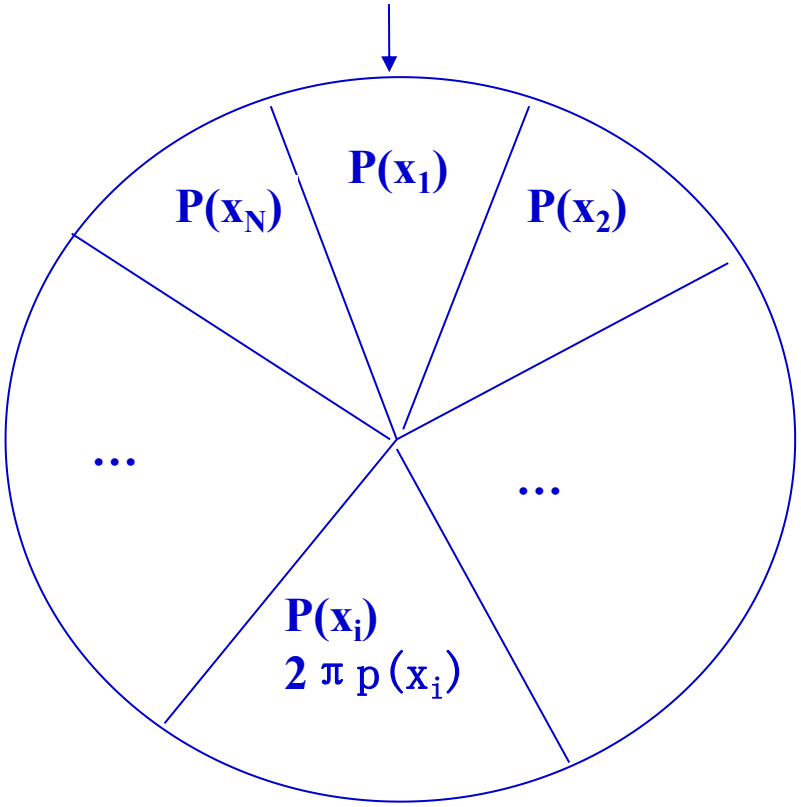


图4.25 轮盘赌选择

个体的适应度值越大，其对应的扇区的面积越大，被选中的可能性也越大。这种方法有点类似于发放奖品使用的轮盘，并带有某种赌博的意思，因此亦被称为轮盘赌选择。

程序中，可用累加概率+随机数 实现

4.5 遗传算法

4.5.5 基本遗传操作

(2)交叉操作

交叉（Crossover）操作是指按照某种方式对选择的父代个体的染色体的部分基因进行交配重组，从而形成新的个体。交配重组是自然界中生物遗传进化的一个主要环节，也是遗传算法中产生新的个体的最主要方法。根据个体编码方法的不同，遗传算法中的交叉操作可分为二进制交叉和实值交叉两种类型。

①二进制交叉

二进制交叉（Binary Valued Crossover）是指二进制编码情况下所采用的交叉操作，它主要包括单点交叉、两点交叉、多点交叉和均匀交叉等方法。

4.5 遗传算法

4.5.5 基本遗传操作

单点交叉

先在两个父代个体的编码串中随机设定一个交叉点，然后对这两个父代个体交叉点前面或后面部分的基因进行交换，并生成子代中的两个新的个体。假设两个父代的个体串分别是：

$X=x_1\ x_2 \cdots x_k\ x_{k+1} \cdots x_n$

$Y=y_1\ y_2 \cdots y_k\ y_{k+1} \cdots y_n$

选择第k位为交叉点，交叉后生成的两个新的个体是：

$X'=x_1\ x_2 \cdots x_k\ y_{k+1} \cdots y_n$

$Y'=y_1\ y_2 \cdots y_k\ x_{k+1} \cdots x_n$

例4.12 设有两个父代的个体串A=0 0 1 1 0 1 和B=1 1 0 0 1 0，若随机交叉点为4，则交叉后生成的两个新的个体是：

A'= 0 0 1 1 1 0

B'= 1 1 0 0 0 1

4.5 遗传算法

4.5.5 基本遗传操作

两点交叉

先在两个父代个体的编码串中随机设定两个交叉点，然后再按这两个交叉点进行部分基因交换，生成子代中的两个新的个体。

假设两个父代的个体串分别是：

$X=x_1\ x_2 \cdots x_i \cdots x_j \cdots x_n$

$Y=y_1\ y_2 \cdots y_i \cdots y_j \cdots, y_n$

选定i、j为交叉点（其中i<j<n），交叉后生成的两个新的个体是：

$X'=x_1\ x_2 \cdots x_i\ y_{i+1} \cdots y_j\ x_{j+1} \cdots x_n$

$Y'=y_1\ y_2 \cdots y_i\ x_{i+1} \cdots x_j\ y_{j+1} \cdots y_n$

例4.13 设有两个父代的个体串A= 0 0 1 1 0 1 和B= 1 1 0 0 1 0，若随机交叉点为3和5，则交叉后的两个新的个体是：

A'= 0 0 1 0 1 1

B'= 1 1 0 1 0 0

4.5 遗传算法

4.5.5 基本遗传操作

多点交叉

多点交是指先随机生成多个交叉点，然后再按这些交叉点分段地进行部分基因交换，生成子代中的两个新的个体。

假设两个父代的个体串分别是$X=x_1\ x_2 \cdots x_i \cdots x_j \cdots x_k \cdots x_n$和$Y=y_1\ y_2 \cdots y_i \cdots y_j \cdots y_k \cdots y_n$，随机设定第i、j、k位为三个交叉点（其中i<j<k<n），则将构成两个交叉段。交叉后生成的两个新的个体是：

$X'= x_1\ x_2 \cdots x_i\ y_{i+1} \cdots y_j\ x_{j+1} \cdots x_k\ y_{k+1} \cdots y_n$

$Y'= y_1\ y_2 \cdots y_i\ x_{i+1} \cdots x_j\ y_{j+1} \cdots y_k\ x_{k+1} \cdots x_n$

例4.14 设有两个父代的个体串A= 0 0 1 1 0 1 和B= 1 1 0 0 1 0，若随机交叉点为1、3和5，则交叉后的两个新的个体是：

A'= 0 1 0 1 0 0

B'= 1 0 1 0 1 1

4.5 遗传算法

4.5.5 基本遗传操作

均匀交叉

均匀交叉（Uniform Crossover）是先随机生成一个与父串具有相同长度，并被称为交叉模版（或交叉掩码）的二进制串，然后再利用该模版对两个父串进行交叉，即将模版中1对应的位进行交换，而0对应的位不交换，依此生成子代中的两个新的个体。事实上，这种方法对父串中的每一位都是以相同的概率随机进行交叉的。

例4.15 设有两个父代的个体串A=001101和B=110010，若随机生成的模版T=010011，则交叉后的两个新的个体是A'=011010和B'=100101。即

A：　0 0 1 1 0 1

B：　1 1 0 0 1 0

T：　0 1 0 0 1 1

A'：0 1 1 1 1 0

B'：1 0 0 0 0 1

4.5 遗传算法

4.5.5 基本遗传操作

②实值交叉

在实数编码情况下所采用的交叉操作，主要包括离散交叉和算术交叉，下面主要讨论离散交叉（部分离散交叉和整体离散交叉）。

部分离散交叉是先在两个父代个体的编码向量中随机选择一部分分量，然后对这部分分量进行交换，生成子代中的两个新的个体。

整体交叉则是对两个父代个体的编码向量中的所有分量，都以1/2的概率进行交换，从而生成子代中的两个新的个体。

以部分离散交叉为例，若选择对第k个分量以后的所有分量进行交换，则生成的两个新的个体向量是：

$X' = x_1\ x_2 \cdots x_k\ y_{k+1} \cdots y_n$

$Y' = y_1\ y_2 \cdots y_k\ x_{k+1} \cdots x_n$

例4.16 设有两个父代个体向量A=20 16 19 32 18 26和B=36 25 38 12 21 30，若随机选择对第3个分量以后的所有分量进行交叉，则交叉后两个新的个体向量是：

A'= 20 16 19 12 21 30

B'= 36 25 38 32 18 26

4.5 遗传算法

4.5.5 基本遗传操作

(3) 变异操作

变异（Mutation）是指对选中个体的染色体中的某些基因进行变动，以形成新的个体。变异是生物遗传和自然进化中的一种基本现象，它可增强种群的多样性。遗传算法中的变异操作增加了算法的局部随机搜索能力。根据个体编码方式的不同，变异操作可分为二进制变异和实值变异两种类型。

① 二进制变异

当个体的染色体采用二进制编码表示时，其变异操作应采用二进制变异方法。该变异方法是先随机地产生一个变异位，然后将该变异位置上的基因值由“0”变为“1”，或由“1”变为“0”，产生一个新的个体。

例4.17 设变异前的个体为A=0 0 1 1 0 1，若随机产生的变异位置是2，则该个体的第2位由“0”变为“1”。

变异后的新的个体是A’= 0 1 1 1 0 1 。

4.5 遗传算法

4.5.5 基本遗传操作

②实值变异

当个体的染色体采用实数编码表示时，其变异操作应采用实值变异方法。该方法是用另外一个在规定范围内的随机实数去替换原变异位置上的基因值，产生一个新的个体。最常用的实值变异操作有：

基于位置的变异方法：该方法是先随机地产生两个变异位置，然后将第二个变异位置上的基因移动到第一个变异位置的前面。

例4.18 设选中的个体向量C=20 16 19 12 21 30，若随机产生的两个变异位置分别时2和4，则变异后的新的个体向量是：

C'= 20 12 16 19 21 30

基于次序的变异：该方法是先随机地产生两个变异位置，然后交换这两个变异位置上的基因。

例4.19 设选中的个体向量D=20 12 16 19 21 30，若随机产生的两个变异位置分别时2和4，则变异后的新的个体向量是：

D'= 20 19 16 12 21 30

4.5 遗传算法

4.5.6 遗传算法应用简例

例4.20 用遗传算法求函数 $f(x)=x^2$ 的最大值，x为[0，31]间的整数。

解：这个问题本身比较简单，其最大值很显然是在x=31处。但作为一个例子，它有着较好的示范性和可理解性。

按照遗传算法，其求解过程如下：

(1) 编码

由于x的定义域是区间[0，31]上的整数，由5位二进制数即可全部表示。因此，可采用二进制编码方法，其编码串的长度为5。

例如，用二进制串00000来表示x=0,11111来表示x=31等。其中的0和1为基因值。

(2) 生成初始种群

若假设给定的种群规模N=4，则可用4个随机生成的长度为5的二进制串作为初始种群。再假设随机生成的初始种群（即第0代种群）为：

s_{01}=0 1 1 0 1　　s_{02}=1 1 0 0 1

s_{03}=0 1 0 0 0　　s_{04}=1 0 0 1 0

4.5 遗传算法

4.5.6 遗传算法应用简例

(3) 计算适应度

要计算个体的适应度，首先应该定义适应度函数。由于本例是求**f(x)**的最大值，因此可直接用**f(x)**来作为适应度函数。

初始种群中各个个体的适应值及其所占比例如表**4-5**所示。

表**4-5** 初始种群情况表

编号	个体串（染色体）	x	适应值	百分比%	累计百分比%	选中次数
S_{01}	01101	13	169	14.30	14.30	1
S_{02}	11001	25	625	52.88	67.18	2
S_{03}	01000	8	64	5.41	72.59	0
S_{04}	10010	18	324	27.41	100	1

可以看出，在**4**个个体中S_{02}的适应值最大，是当前最佳个体。

4.5 遗传算法

4.5.6 遗传算法应用简例

(4) 选择操作

假设采用轮盘赌方式选择个体，且依次生成的4个随机数（相当于轮盘上指针所指的数）为0.85、0.32、0.12和0.46，经选择后得到的新的种群为：

S'_{01}=10010

S'_{02}=11001

S'_{03}=01101

S'_{04}=11001

其中，染色体11001在种群中出现了2次，而原染色体01000则因适应值太小而被淘汰

4.5 遗传算法

4.5.6 遗传算法应用简例

(5) 交叉

假设交叉概率P_i为50%，则种群中只有1/2的染色体参与交叉。若规定种群中的染色体按顺序两两配对交叉，且有S'_{01}与S'_{02}交叉，S'_{03}与S'_{04}不交叉，则交叉情况如表4-6所示。

编号	个体串（染色体）	交叉对象	交叉位	子代	适应值
S'_{01}	10010	S'_{02}	3	10001	289
S'_{02}	11001	S'_{01}	3	11010	676
S'_{03}	01101	S'_{04}	N	01101	169
S'_{04}	11001	S'_{03}	N	11001	625

可见，经交叉后得到的新的种群为：

S''_{01}=10001

S''_{02}=11010

S''_{03}=01101

S''_{04}=11001

4.5 遗传算法

4.5.6 遗传算法应用简例

(6) 变异

变异概率P_m一般都很小，假设本次循环中没有发生变异，则变异前的种群即为进化后所得到的第1代种群。即：

S_{11}=10001

S_{12}=11010

S_{13}=01101

S_{14}=11001

然后，对第1代种群重复上述(4)-(6)的操作

4.5 遗传算法

4.5.6 遗传算法应用简例

表4-7 第1代种群的选择情况表

编号	个体串（染色体）	x	适应值	百分比%	累计百分比%	选中次数
S_{11}	10001	17	289	16.43	16.43	1
S_{12}	11010	26	676	38.43	54.86	2
S_{13}	01101	13	169	9.61	64.47	0
S_{14}	11001	25	625	35.53	100	1

其中若假设按轮盘赌选择时依次生成的4个随机数为0.14、0.51、0.24和0.82，经选择后得到的新的种群为：

S'_{11}=10001

S'_{12}=11010

S'_{13}=11010

S'_{14}=11001

可见，染色体11010被选择了2次，而原染色体01101则被淘汰。

4.5 遗传算法

4.5.6 遗传算法应用简例

表4-8 第1代种群的交叉情况表

编号	个体串（染色体）	交叉对象	交叉位	子代	适应值
S'_{11}	10001	S'_{12}	3	10010	324
S'_{12}	11010	S'_{11}	3	11001	625
S'_{13}	11010	S'_{14}	2	11001	625
S'_{14}	11001	S'_{13}	2	11010	675

可见，经杂交后得到的新的种群为：

S''_{11}=10010

S''_{12}=11001

S''_{13}=11001

S''_{14}=11010

可以看出，第3位基因均为0，已经不可能通过交配达到最优解。这种过早陷入局部最优解的现象称为早熟。为解决这一问题，需要采用变异操作。

4.5 遗传算法

4.5.6 遗传算法应用简例

表4-9 第1代种群的变异情况表

编号	个体串（染色体）	是否变异	变异位	子代	适应值
S''_{11}	10010	N		10010	324
S''_{12}	11001	N		11001	625
S''_{13}	11001	N		11001	625
S''_{14}	11010	Y	3	11110	900

它是通过对S''_{14}的第3位变异来实现的。变异后所得到的第2代种群为：

S_{21}=10010

S_{22}=11001

S_{23}=11001

S_{24}=11110

接着，再对第2代种群同样重复上述(4)-(6)的操作：

4.5 遗传算法

4.5.6 遗传算法应用简例

表4-10 第2代种群的选择情况表

编号	个体串（染色体）	x	适应值	百分比%	累计百分比%	选中次数
S_{21}	10010	18	324	23.92	23.92	1
S_{22}	11001	25	625	22.12	46.04	1
S_{23}	11001	25	625	22.12	68.16	1
S_{24}	11110	30	900	31.84	100	1

其中若假设按轮盘赌选择时依次生成的4个随机数为0.42、0.15、0.59和0.91，经选择后得到的新的种群为：

S'_{21}=11001

S'_{22}=10010

S'_{23}=11001

S'_{24}=11110

4.5 遗传算法

4.5.6 遗传算法应用简例

图4.11 第2代种群的交叉情况

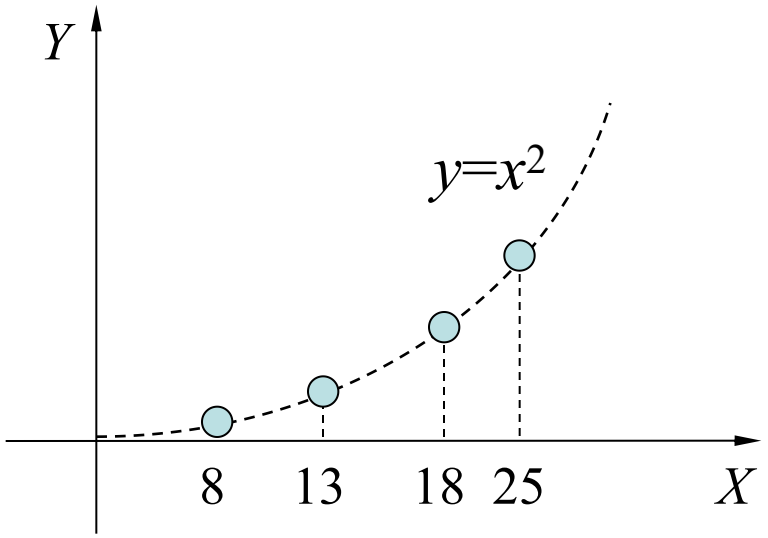
编号	个体串（染色体）	交叉对象	交叉位	子代	适应值
S'_{21}	11001	S'_{22}	3	11010	676
S'_{22}	10010	S'_{21}	3	10001	289
S'_{23}	11001	S'_{24}	4	11000	576
S'_{24}	11110	S'_{23}	4	11111	961

这时，函数的最大值已经出现，其对应的染色体为11111，经解码后可知问题的最优解是在点x=31处。

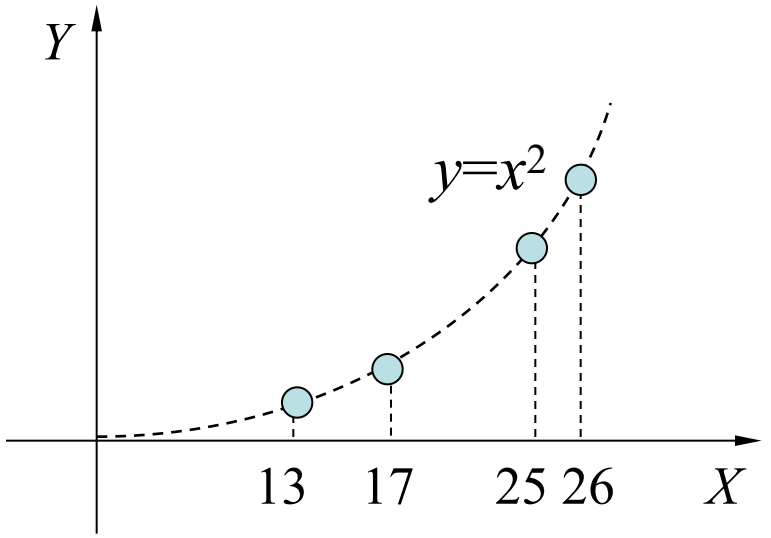
假设不发生变异，此时子带即为第3代种群

求解过程结束。

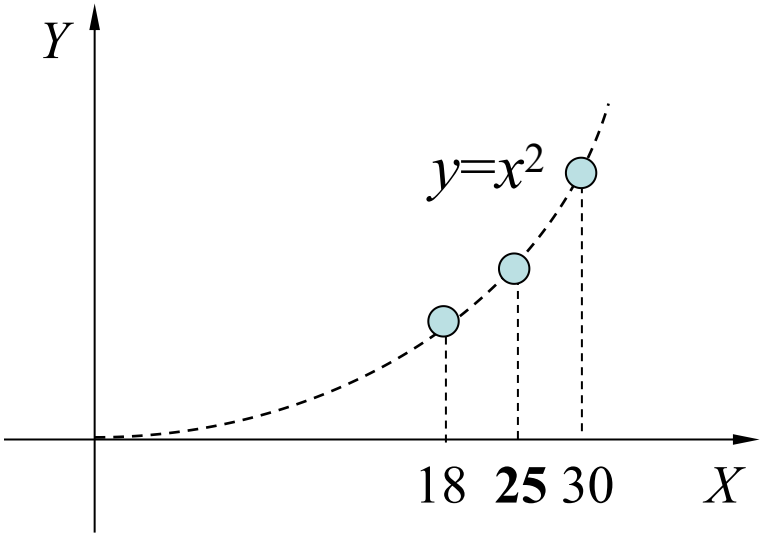
种群演化情况



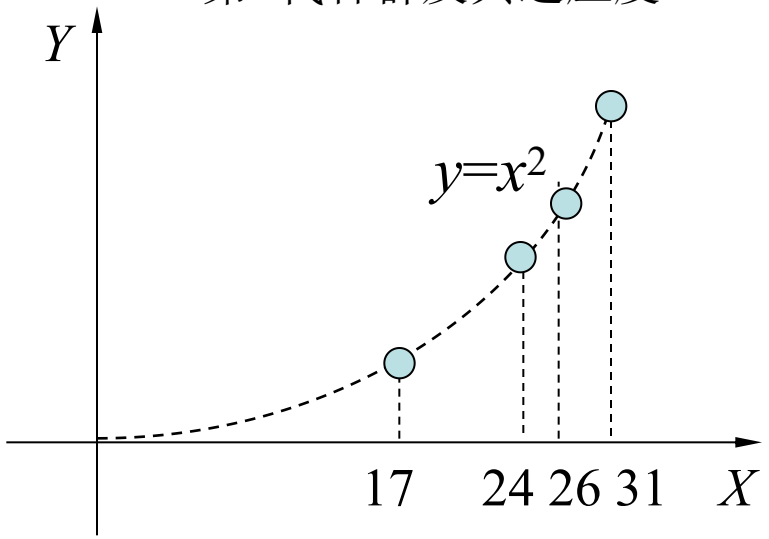
第0代种群及其适应度



第1代种群及其适应度



第2代种群及其适应度



第3代种群及其适应度

4.5 遗传算法

- 遗传算法的特点 （P124-125）
- 遗传算法的收敛性

遗传算法要实现全局收敛，要求任意初始种群经有限步都能到达全局最优解；算法必须有保优操作来防止最优解的遗失。与算法收敛性有关的因素主要包括

- 种群规模
- 选择操作
- 交叉概率
- 变异概率

影响算法收敛性的因素

- 种群规模：通常，种群太小则不能提供足够的采样点，以致算法性能很差；种群太大，尽管可以增加优化信息，阻止早熟收敛的发生，但无疑会增加计算量，造成收敛时间太长，表现为收敛速度缓慢。
- 选择操作：选择操作使高适应度个体能够以更大的概率生存，从而提高了遗传算法的全局收敛性。如果在算法中采用最优保存策略，即将父代群体中最佳个体保留下来，不参加交叉和变异操作，使之直接进入下一代，最终可使遗传算法以概率1收敛于全局最优解。
- 交叉操作：交叉操作用于个体对，产生新的个体，实质上是在解空间中进行有效搜索。交叉概率太大时，种群中个体更新很快，会造成高适应度值的个体很快被破坏掉；概率太小时，交叉操作很少进行，从而会使搜索停滞不前，造成算法的不收敛。
- 变异操作：变异操作是对种群模式的扰动，有利于增加种群的多样性 。但是，变异概率太小则很难产生新模式，变异概率太大则会使遗传算法成为随机搜索算法。

A Jumping Genes Paradigm: Theory, Verification, and Applications

Wallace K.S. Tang, Sam T.W. Kwong, and Kim F. Man

1. Introduction

Engineering designs are usually multi-objective or tated. The technological advantage of adopting ev tionary computing is its uniqueness in solving a ra of objective functions in a simultaneous manner. Unlike sin objective optimization, which only aims for a global solut the multiobjective optimization (MO) is to find a set of r dominated solutions via the Pareto-optimal methodology satisfy a range of specific design requirements [22], [23] There are two technically desirable goals for achiev

Abstract:

A new evolutionary computing algorithm on the basis of "jumping genes" phenomenon is presented in this article. It emulates the gene transposition in the genome that was discovered by Nobel Laureate Dr. Barbara McClintock from her work on maize chromosome. The principle of jumping genes, adopted for evolutionary computing, is outlined and the procedures for executing the computational optimization are provided. Mathematical derivation of the Schema Theorem is briefly discussed, which is established to demonstrate the searching capacity of the newly proposed algorithm, in terms of convergence and diversity. The algorithm is found to be robust and provides outcomes in speed and accuracy, while the solutions are widely spread along the Pareto-optimal front when a multiobjective problem is tackled. To further reinforce the jumping genes proposition, some typical engineering design problems are included. The obtained results have indicated that this new algorithm is indeed capable of searching multiobjective solutions including the extreme solutions at both ends of the Pareto-optimal front.

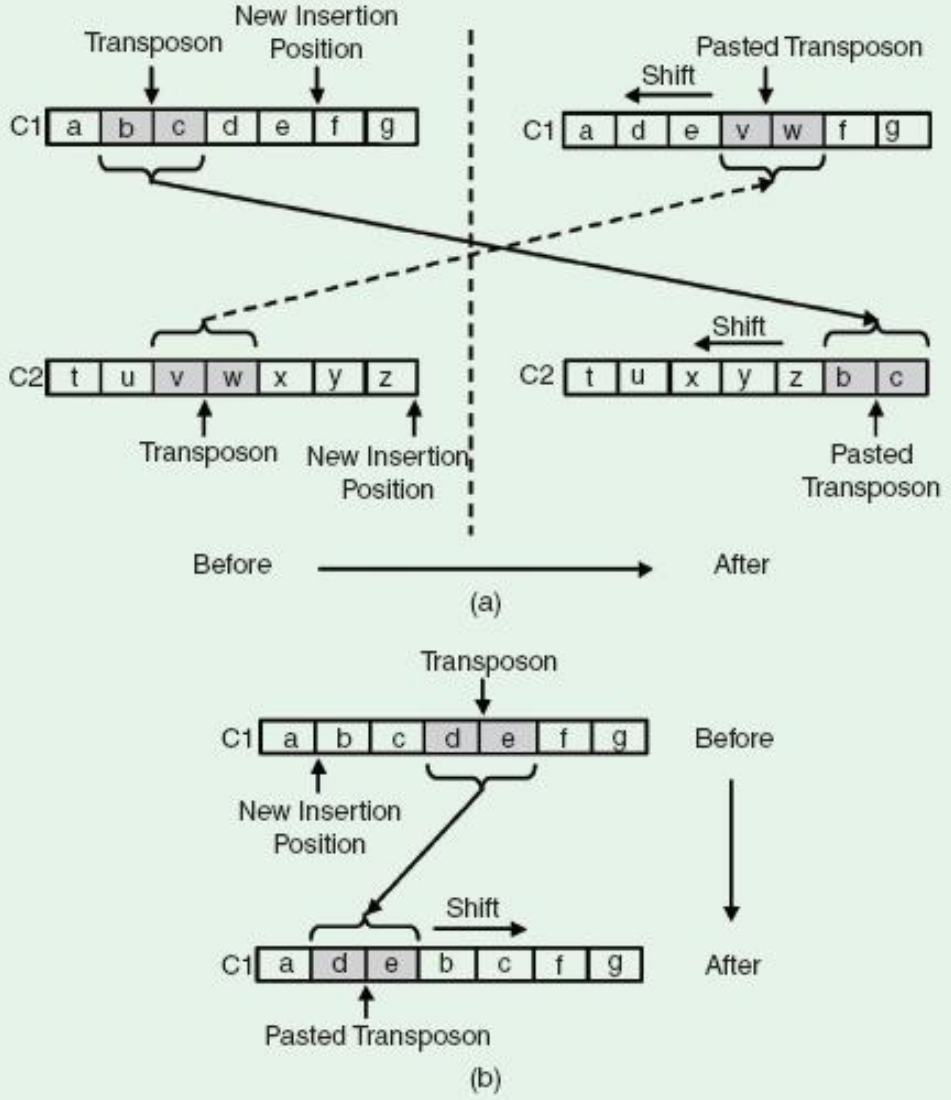


Figure 2. Cut-and-paste operation on (a) two different chromosomes and (b) the same chromosome.

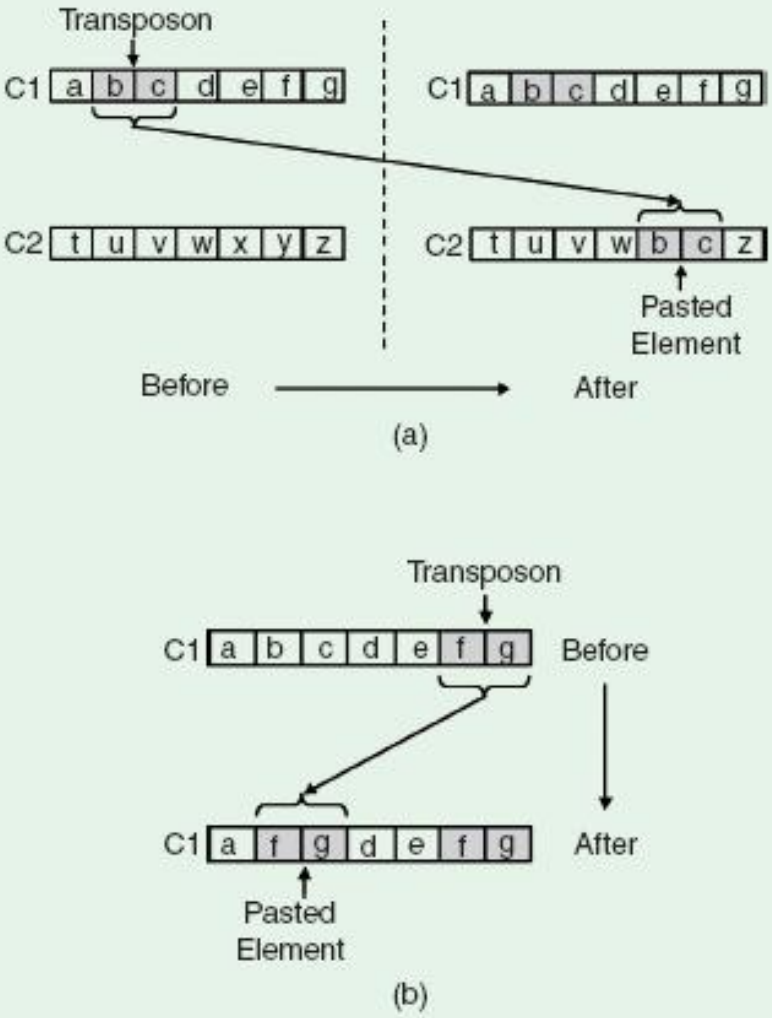


Figure 3. Copy-and-paste operation on (a) two different chromosomes and (b) the same chromosome.

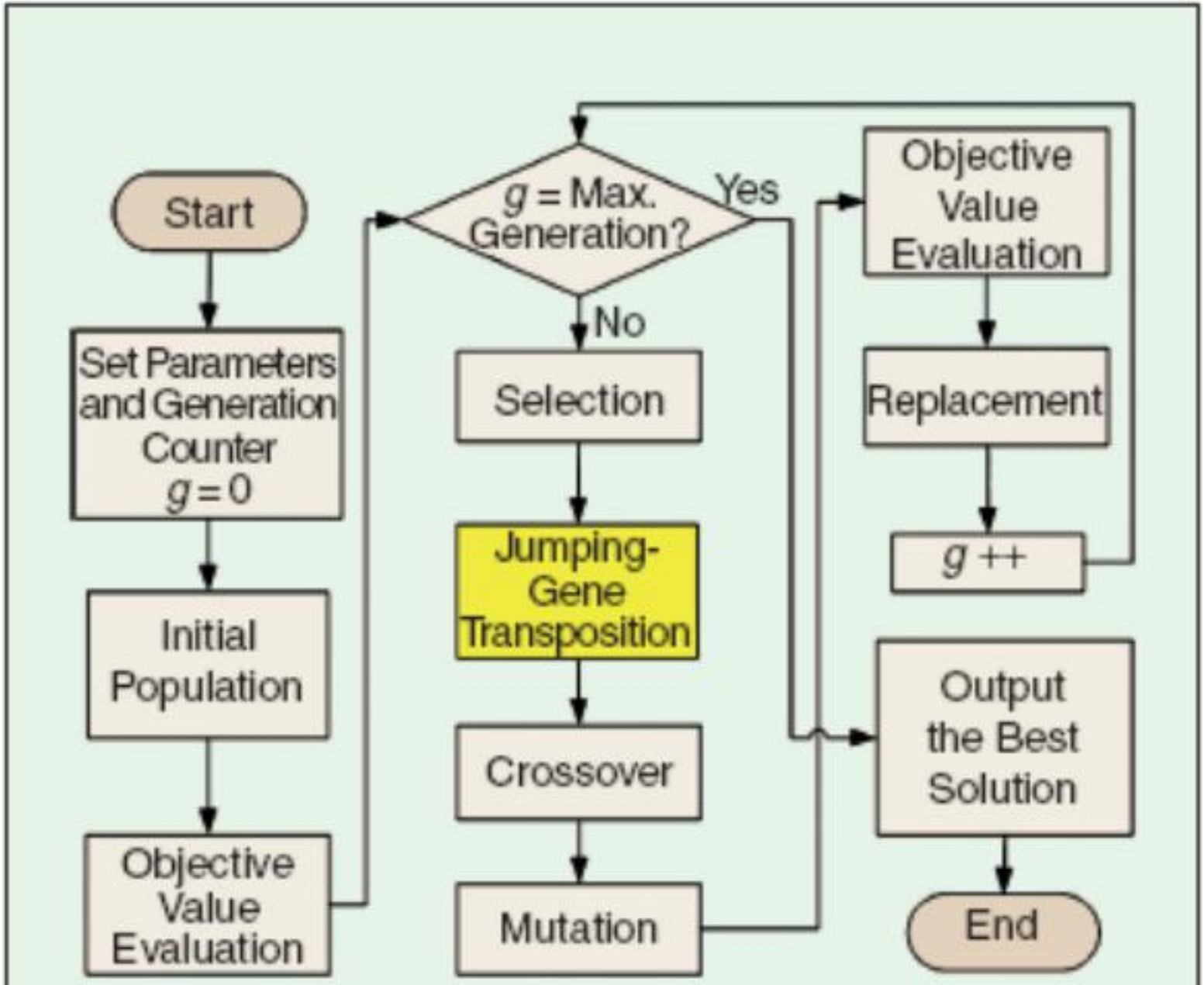


Figure 6. Genetic cycle of JGEA.

Publications

- **S Y Zhang, S H Yeung, W S Chan, K F Man and K S Tang, 2009:** *Design of Broadband Hybrid Coupler with Tight Coupling Using Jumping Gene evolutionary Algorithm,* **IEEE Tran. on Industrial Electronics,vol.56,No.8, pp2987-2991, 2009.**
- **X S Yang, S H Yeung, K T Ng and K F Man, 2008:** *Jumping Genes Multiobjective Optimization of Plannar Monopole Ultra-Wide Band Antenna*, **IEEE Tran. on Antennas and Propagation, vol.56,No.2,pp3659-3666, Dec., 2008.**
- **W K S Tang, S T W Kwong, and K F Man:** *A Jumping Genes Paradigm: Theory, Verification and Applications,* **IEEE Circuits and Systems Magazine, vol.8 issue 4, pp18-36, Dec.,2008**
- **H.M. Ma, K T Ng and K F Man, 2008:** *Multi-objective Coordinated Power Voltage Control Using Jumping Genes Paradigm*, **IEEE Tran. on Industrial Electronics, vol.55, No.11, pp4075-4084, Nov., 2008.**
- **S H Yeung, K F Man and W S Chan, 2008:** *Optimized design of an ISM band Antenna Using a Jumping Genes Methodology*, **IET Microwaves Antennas and propagation, Vol.2, No: 3, pp259-267, June 2008.**

Publications

- **S.H. Yeung, H K Ng and K F Man, 2008:** *Multi-criteria Design Methodology of a Dielectric Resonator Antenna with Jumping Genes Evolutionary Algorithm*, **Int. J. of Electronics and Communications, vol.62, No:4, pp266-276, April 2008.**
- **S H Yeung, K F Man, K M Luk and C H Chan, 2008:** *A Trapeizform U-slot Folded Patch Feed Antenna Design Optimized with Jumping Genes Evolutionary Algorithm,* **IEEE Tran. on Antennas and propagation, vol.56,No:2, pp571-577, Feb., 2008.**
- **T M Chan, K F Man, S Kwong and K S Tang, 2008** *A Jumping Gene paradigm for Evolutionary Multiobjective Optimization*, **IEEE Tran. on Evolutionary Computation, vol.12 No:2, pp143-159, April 2008.**
- **K S Ripon, S Kwong and K F Man, 2007:** ***A Real-coding Jumping Gene Genetic Algorithm (RJJGA) for Multiobjective Optimization*, Information Science, vol.177, No:2, pp632-654, January 2007.**
- **T M Chan, K F Man, K S Tang and S Kwong, 2007:** *A Jumping Genes paradigm for Optimizing Factory WLAN Network* **IEEE Tran. on Industrial Informatics, vol.3, No:1, pp33-43, Feb.2007.**

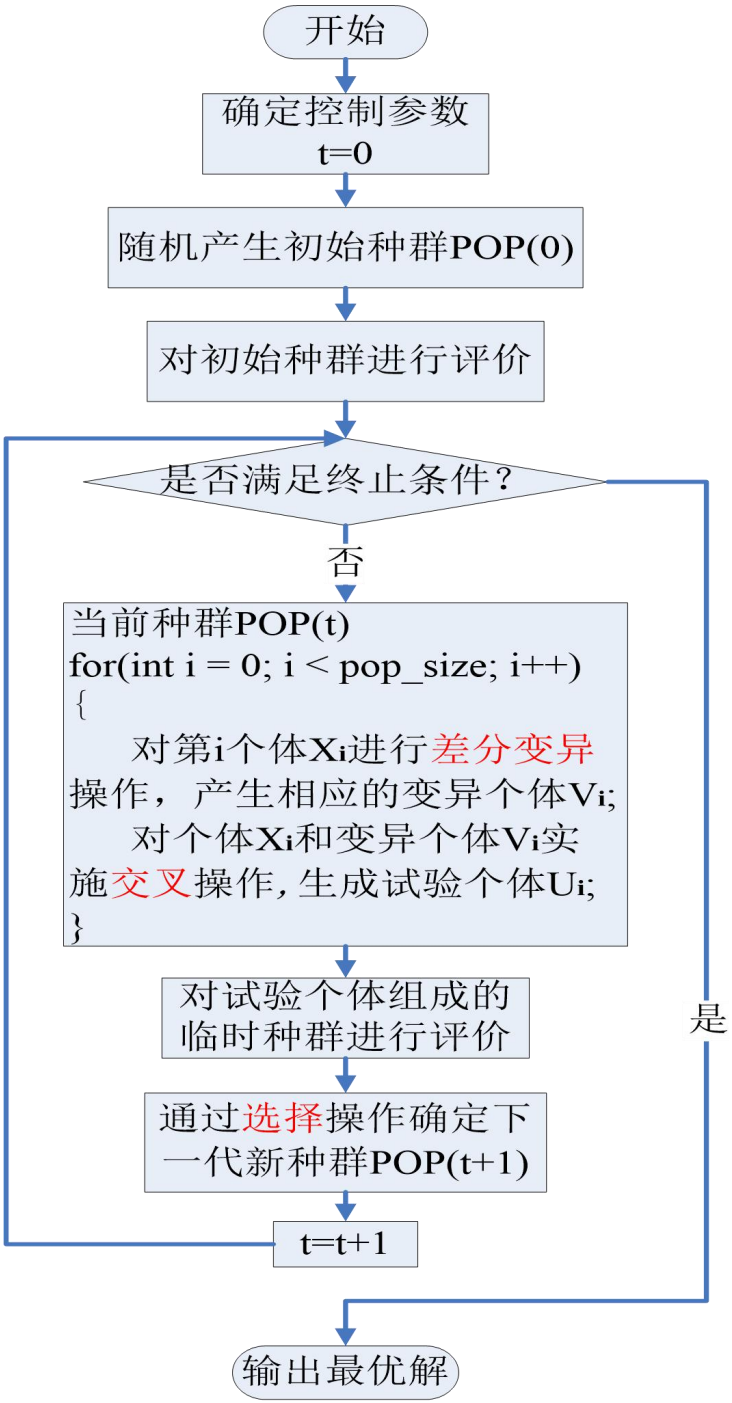
差分演化(Differential Evolution)算法

- **Rainer Storn 和Kenneth Price在1996 年为求解切比雪夫多项式而提出；**
- **在当年IEEE演化计算大赛中表现超群，随后在多个领域成功应用。**
- **DE是一种随机的并行直接搜索算法,它可对非线性不可微连续空间函数进行最小化**
- **具有易用性、稳健性和强大的全局寻优能力；**

标准DE流程图

DE算法：

- 基于实数编码；
- 整体结构类似于遗传算法；
- 变异操作是基于染色体的差异向量进行的；



基本原理

求解函数f (x)的最小值问题, 令 $x_i(t)$ 是第t代的第i个染色体, 满足:

$$x_i(t)=\left[x_{i,1}(t),x_{i,2}(t),\cdots,x_{i,n}(t)\right]$$

$$i=1,2,\cdots,M;\quad t=1,2,\cdots t_{\max}.$$

且

$$x_{ij}^{L}\le x_{ij}\le x_{ij}^{U}\qquad j=1,2,\cdots n$$

其中,n 是染色体的长度,即变量的个数,M为群体规模, $t_{\max}$ 是最大的进化代数。

基本原理

(1) 生成初始种群

在**n** 维空间里随机产生满足约束条件的**M** 个染色体：

$$x_{i,j}(0) = x_{ij}^{L} + rand_{ij}(0,1)\left(x_{ij}^{U} - x_{ij}^{L}\right),$$

$$i = 1,2,\cdots,M;\ j = 1,2,\cdots n$$

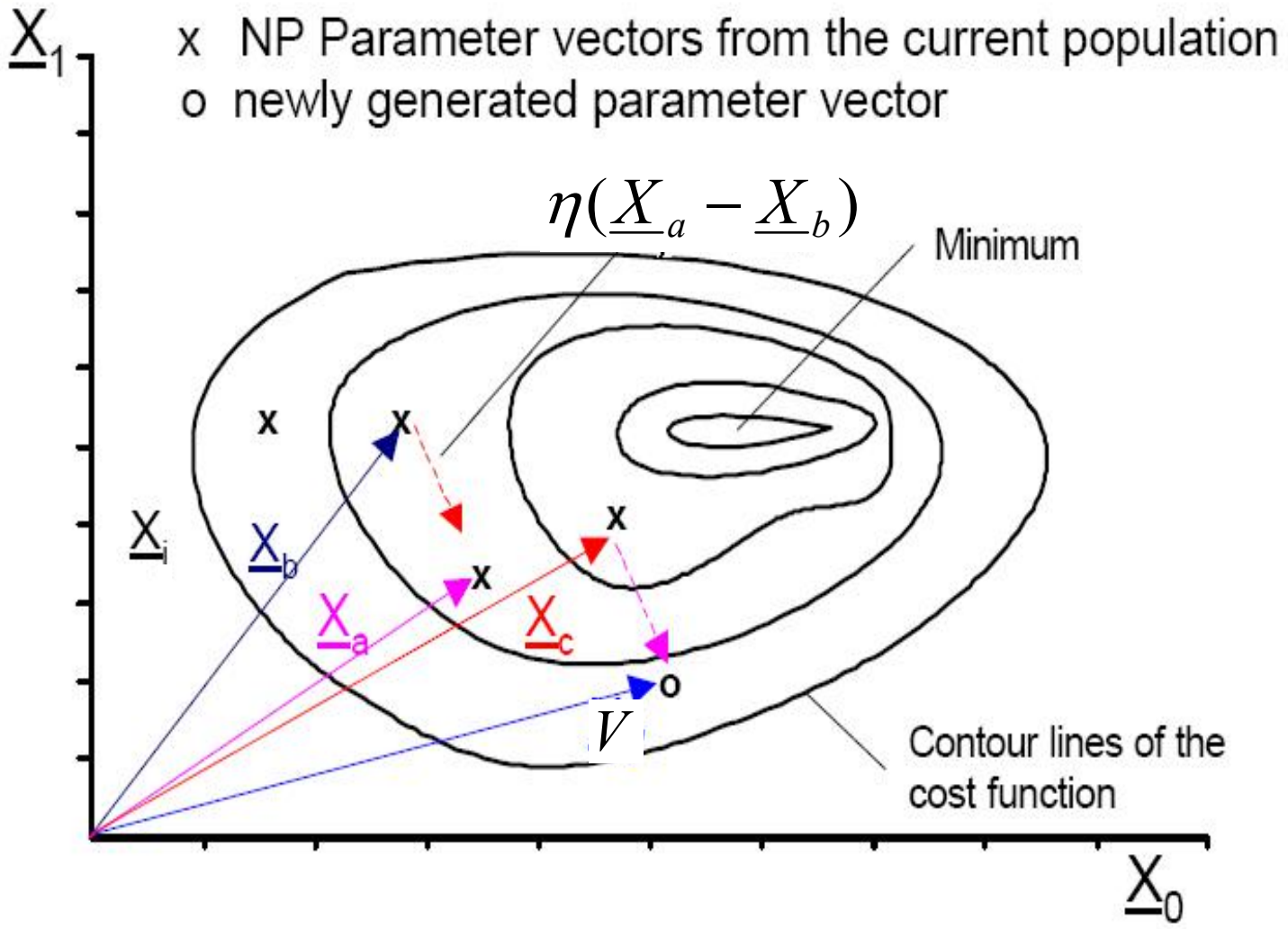
(2) 变异操作

从群体中随机选择**3** 个染色体 x_{p1}，x_{p_2}，x_{p_3}，且 **(i≠p1≠p2≠p3)**，$x_i(t)$是目标向量，则变异向量

$$v_{ij}(t+1) = x_{p_1 j}(t) + \eta\left(x_{p_2 j}(t) - x_{p_3 j}(t)\right)$$

$x_{p_2 j}(t) - x_{p_3 j}(t)$为差异化向量，$\eta$为缩放因子。

基本原理


X1
x NP Parameter vectors from the current population
o newly generated parameter vector
η(Xa − Xb)
Minimum
Xi
Xb
Xa
Xc
V
Contour lines of the cost function
X0

基本原理

(3) 交叉操作

交叉操作是为了增加群体的多样性, 具体操作如下:

$$u_{ij}(t+1)=\begin{cases} v_{ij}(\mathrm{t}+1) & rand1_{ij} \le CR \text{或} j = rand(\mathrm{i}) \\ x_{\mathrm{ij}}(\mathrm{t}) & rand1_{ij} > CR \text{且} j \ne rand(\mathrm{i}) \end{cases}$$

$rand1_{ij}$ 是在[0, 1]之间的随机小数, CR为交叉概率, CR∈[0, 1], rand(i)在[1, n]之间的随机整数, 这种交叉策略可确保u_i (t+ 1) 至少有一分量由v_i (t+1) 的相应分量贡献。

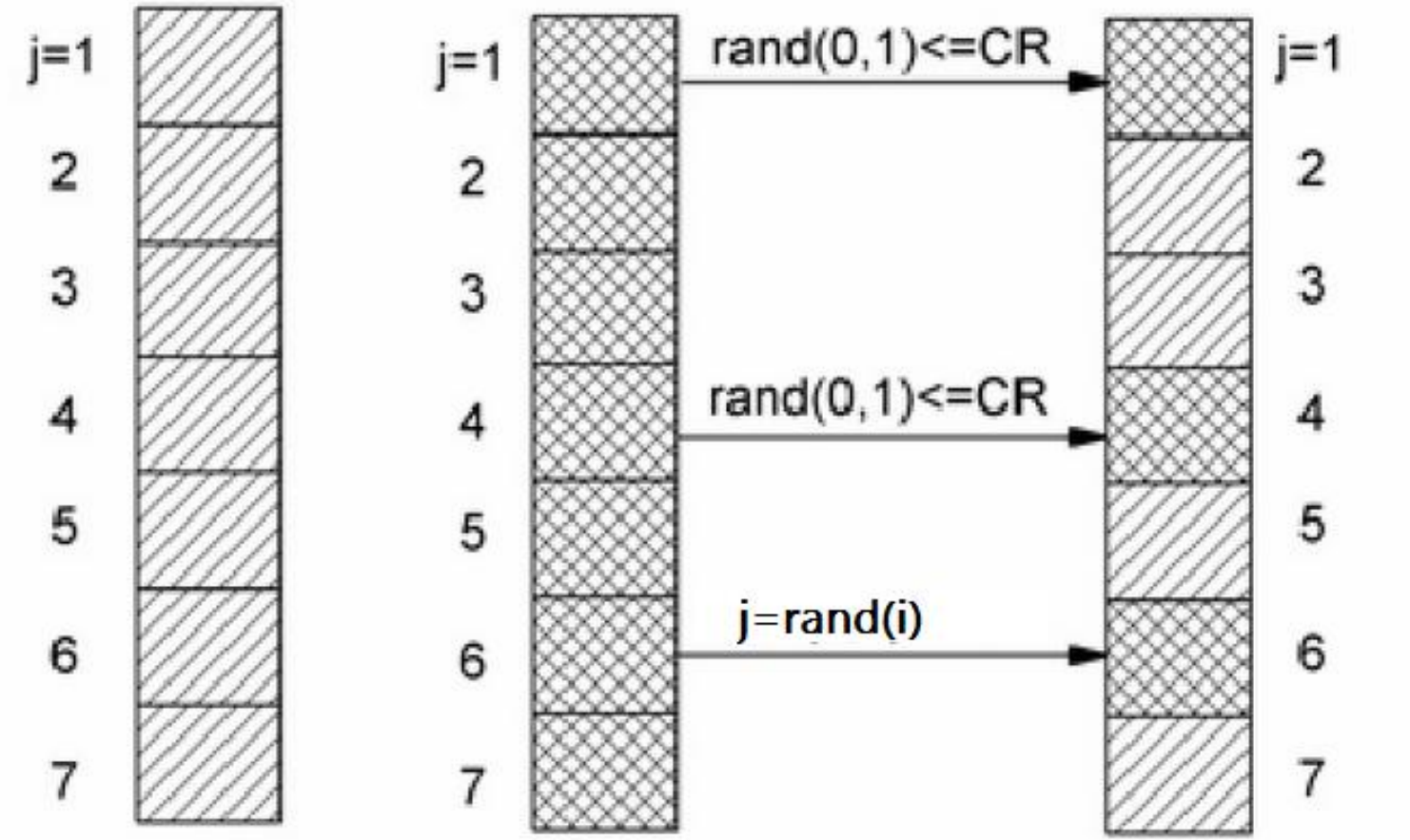
(4) 选择操作

比较试验向量 $u_i(t+1)$ 和目标向量 $x_i(t)$ 的评价函数:

$$x_i(t+1)=\begin{cases} u_i(\mathrm{t}+1) & f(u_i(t+1)) \le f(x_i(t)) \\ x_i(t) & otherwise \end{cases} \quad i=1,2,\cdots,M$$

反复执行(2)至(4)操作, 直至达到最大的进化代数 tmax.

基本原理



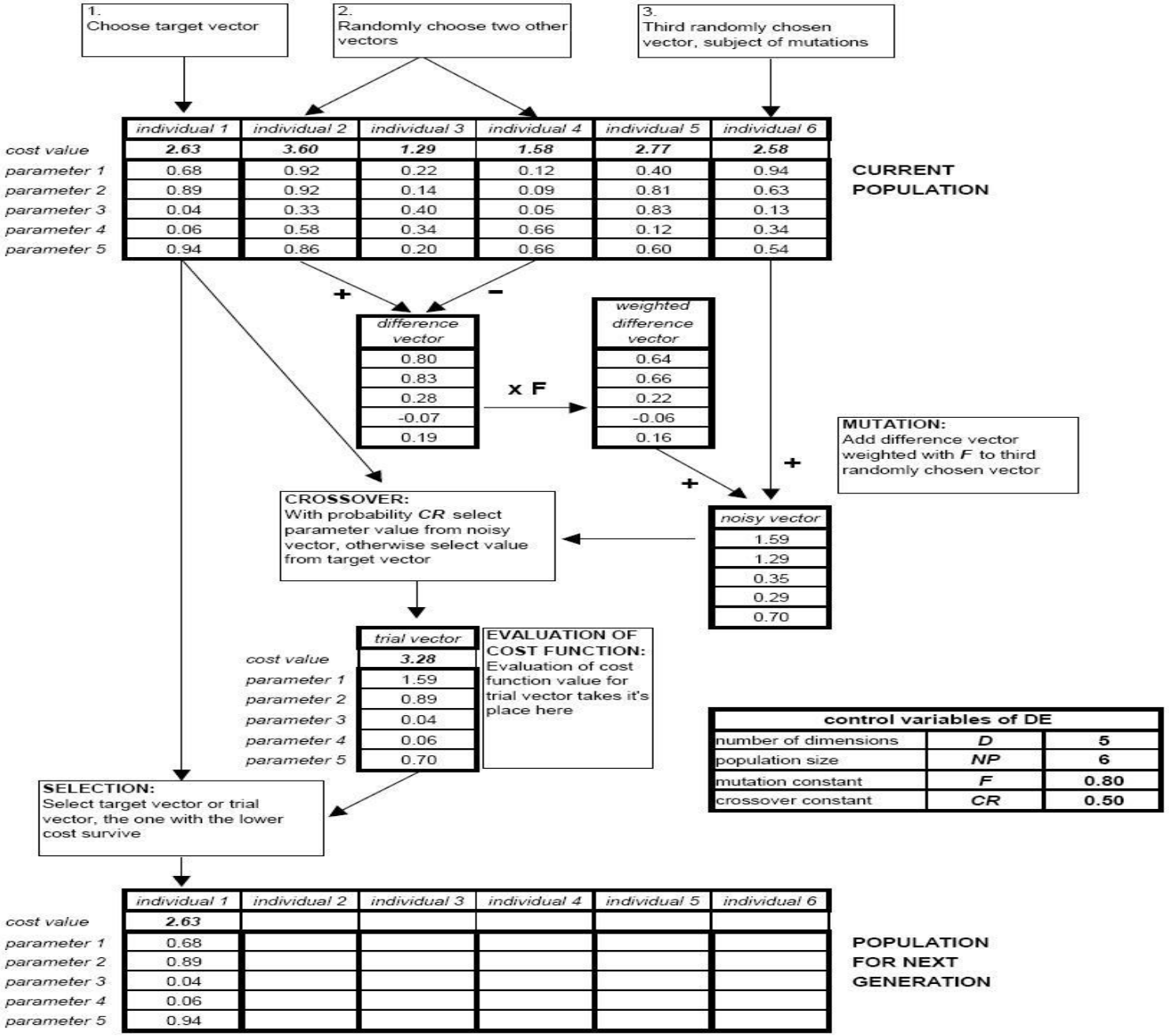
目标个体 $x_i(t)$ (含 7 个变量)

变异向量 $v_i(t+1)$

假设 rand(i)=6

试用个体 $u_i(t+1)$

$f(X) = x_1 + x_2 + x_3 + x_4 + x_5$ (variables bounded within the range [0,1])



差分演化算法的参数选取

差分演化算法主要涉及群体规模M，缩放因子η，以及交叉概率CR三个参数的设定。

M：一般介于**5×n** 与**10×n** 之间, 但不能少于**4**, 否则无法进行变异操作;

η：一般在**[0, 2]**之间选择, 通常可取**0. 5**;

CR：在**[0, 1]**之间选择, 比较好的选择应在**0. 3** 左右。

差分演化算法的优缺点

和其它进化算法相比, 差异演化具有以下优点:

- 差异演化在求解非凸、多峰、非线性函数优化问题表现极强的稳健性。
- 在同样的精度要求下, 差异演化算法收敛的速度快。
- 差异演化算法尤其擅长求解多变量的函数优化问题。
- 操作简单, 易编程实现。

缺点:

由于差分演化的关键步骤变异操作是基于群体的差异向量信息来修正各个体的值, 随着进化代数的增加, 各个体之间的差异化信息在逐渐缩小, 以至于后期收敛速度变慢, 甚至有时会陷入局部极值点。

为提高DE的寻优能力、加快收敛速度、克服启发式算法常见的早熟收敛现象，许多学者对DE算法进行了改进。

DE的变形

DE算法的多种变形形式常用符号DE /x/y/z以示区分，其中：

X——限定当前被加差异的向量是“随机的”或“最佳的”；

Y——是所利用的差异向量的个数;

Z——指示交叉模式，有指数模式和二项式模式。

DE/rand/1: $\mathbf{V}_{i,G} = \mathbf{X}_{r_1,G} + F \cdot \left(\mathbf{X}_{r_2,G} - \mathbf{X}_{r_3,G}\right)$

DE/best/1: $\mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F \cdot \left(\mathbf{X}_{r_1,G} - \mathbf{X}_{r_2,G}\right)$

DE/current-to-best/1: $\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + F \cdot \left(\mathbf{X}_{best,G} - \mathbf{X}_{i,G}\right) + F \cdot \left(\mathbf{X}_{r_1,G} - \mathbf{X}_{r_2,G}\right)$

DE/rand/2: $\mathbf{V}_{i,G} = \mathbf{X}_{r_1,G} + F \cdot \left(\mathbf{X}_{r_2,G} - \mathbf{X}_{r_3,G} + \mathbf{X}_{r_4,G} - \mathbf{X}_{r_5,G}\right)$

DE/best/2: $\mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F \cdot \left(\mathbf{X}_{r_1,G} - \mathbf{X}_{r_2,G} + \mathbf{X}_{r_3,G} - \mathbf{X}_{r_4,G}\right)$