人工智能

Lecture 7: 群体智能 (Swarm Intelligence)

赵培海

东华大学
计算机科学与技术学院

2023 年 9 月 6 日

内容组织

1. 遗传算法

2. 差分进化算法

内容组织

进化算法 (Evolutionary Algorithms, EA)

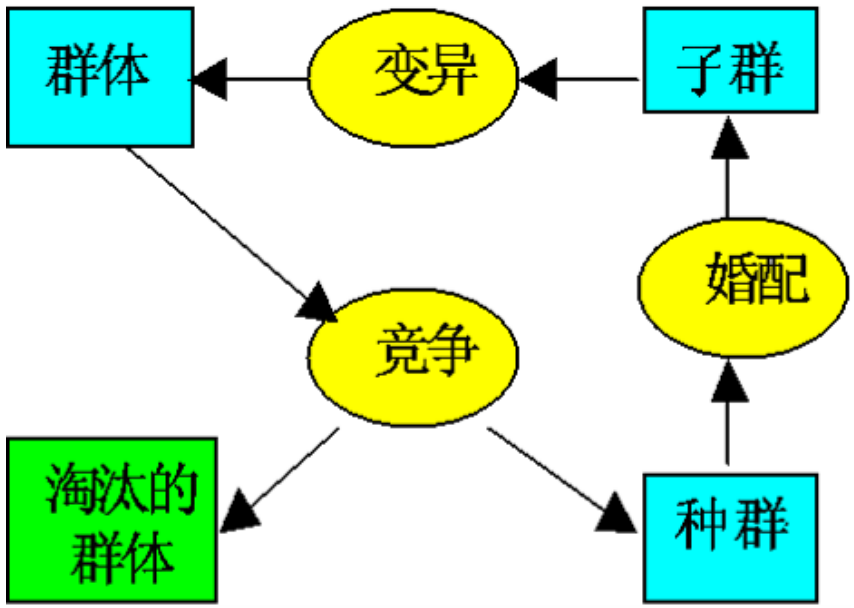
- 进化算法 (Evolutionary Algorithms, EA) 是基于自然选择和自然遗传等生物进化机制的一种搜索算法.
- 进化算法主要通过选择、重组和变异这三种操作实现优化问题的求解.
- 进化算法是一个“算法簇”，包括遗传算法 (GA)、遗传规划、进化策略和进化规划等.
- 进化算法的基本框架是遗传算法所描述的框架.
- 进化算法可广泛应用于组合优化、机器学习、自适应控制、规划设计和人工生命等领域.

进化算法的生物学背景

适者生存：最适合自然环境的群体往往产生了更大的后代群体.

生物进化的基本过程：

- 染色体 (chromosome)：生物的遗传物质的主要载体.
- 基因 (gene)：扩展生物性状的遗传物质的功能单元和结构单位.
- 基因座 (locus)：染色体中基因的位置.
- 等位基因 (alleles)：基因所取的值.



遗传算法 (Genetic Algorithm, GA)

- 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

GA 基本思想

表: 遗传算法基本思想

生物遗传概念	理性地思考的系统遗产算法中的应用
适者生存	目标值比较大的解被选择的可能性大
个体（Individual）	解
染色体（Chromosome）	解的编码（字符串、向量等）
基因（Gene）	解的编码中每一分量
适应性（Fitness）	适应度函数值
群体（Population）	根据适应度值选定的一组解（解的个数为群体的规模）
婚配（Marry）	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
变异（Mutation）	编码的某一分量发生变化的过程

GA 的基本内容

- 编码方案：怎样把优化问题的解进行编码.
- 适应度函数：怎样根据目标函数构建适应度函数.
- 选择策略：优胜劣汰.
- 控制参数：种群的规模、算法执行的最大代数、执行不同遗传操作的概率等.
- 遗传算子：选择、交叉、变异.
- 算法终止准则：规定一个最大的演化代数，或算法在连续多少代以后解的适应值没有改进.

GA 的基本算法

遗传算法的五个基本要素：

1. 参数编码
2. 初始群体的设定
3. 适应度函数的设计
4. 遗传操作设计
5. 控制参数设定

GA 的基本算法

遗传算法的一般步骤

1. 编码
2. 群体设定
3. 适应度函数
4. 选择
5. 交叉
6. 变异

编码

1. 位串编码

位串编码

一维染色体编码方法：将问题空间的参数编码为一维排列的染色体的方法.

- 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
  - 然后在位串空间上进行遗传操作

编码

1. 位串编码

   - 二进制编码

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

1. 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.

$$15 : 01111 \qquad 16 : 10000$$

2. 要先给出求解的精度.
3. 求解高维优化问题的二进制编码串长，算法的搜索效率低.

编码

2. 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.

群体设定

初始种群的产生

- 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.
- 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.

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：若群体规模为 M，则遗传操作可从这 M 个个体中生成和检测 M^3 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

适应度函数

1. 将目标函数映射成适应度函数的方法

- 若目标函数为最大化问题，则 $Fit(f(x)) = f(x)$
- 若目标函数为最小化问题，则 $Fit(f(x)) = \frac{1}{f(x)}$

$\Downarrow$ 将目标函数转换为求最大值的形式，且保证函数值非负

- 若目标函数为最大化问题，则

$$Fit(f(x)) = \begin{cases} f(x) - C_{min}, & f(x) > C_{min} \\ 0, & others \end{cases}$$

- 若目标函数为最小化问题，则

$$Fit(f(x)) = \begin{cases} C_{max} - f(x), & f(x) < C_{max} \\ 0, & others \end{cases}$$

适应度函数

1. 将目标函数映射成适应度函数的方法

⇓ 存在界限值预选估计困难或者不能精确估计的问题

- 若目标函数为最大化问题，则

$$Fit(f(x)) = \frac{1}{1 + c + f(x)} \quad c \geq 0,\ c + f(x) \geq 0$$

- 若目标函数为最小化问题，则

$$Fit(f(x)) = \frac{1}{1 + c - f(x)} \quad c \geq 0,\ c - f(x) \geq 0$$

- c: 目标函数界限的保守估计值.

适应度函数

2. 适应度函数的尺度变换

- 在遗传算法中，将所有妨碍适应度值高的个体产生，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.
  - 适应度函数的尺度变换（fitness scaling）或者定标：对适应度函数值域的某种映射变换.

适应度函数

2. 适应度函数的尺度变换

(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

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

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

适应度函数

2. 适应度函数的尺度变换

(2) 幂函数变换法：

$$f' = f^K$$

(3) 指数变换法：

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

选择

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.

选择

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:

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

选择

1. 个体选择概率分配方法

(2) 排序方法（rank-based model）

a. 线性排序：

- 群体成员按适应值大小从好到坏依次排列：$x_1, x_2, \cdots, x_N$
- 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

选择

1. 个体选择概率分配方法

(2) 排序方法（rank-based model）

b. 非线性排序：

- 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：

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

选择

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 $P = \{x_1, x_2, \cdots, x_M\}$ 且 $f(x_1) \geq f(x_2) \geq \cdots \geq f(x_M)$, 则 p_i 满足 $p_1 \geq p_2 \geq \cdots \geq p_M$
2. $\sum_{i=1}^{M} p_i = 1$

选择

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

选择

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．

选择

2. 选择个体方法

(3) 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 $p > random[0, 1)$ 选择差解，否则选择好解.

(4) 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.

交叉

基本的交叉算子

(1) 一点交叉（single-point crossover）

- 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）

- 随机设置两个交叉点，将两个交叉点之间的码串相互交换.

交叉

修正的交叉方法

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

$$A = 9\ \ 8\ \ 4 \mid 5\ \ 6\ \ 7 \mid 1\ \ 3\ \ 2$$

$$B = 8\ \ 7\ \ 1 \mid 2\ \ 3\ \ 9 \mid 5\ \ 4\ \ 6$$

$$A' = 9\ \ 8\ \ 4 \mid 2\ \ 3\ \ 9 \mid 1\ \ 3\ \ 2$$

$$B' = 8\ \ 7\ \ 1 \mid 5\ \ 6\ \ 7 \mid 5\ \ 4\ \ 6$$

交叉

修正的交叉方法

(2) 顺序交叉 OX：Davis L. (1985)

$$A' = H \quad 8 \quad 4 \;\big|\; 5 \quad 6 \quad 7 \;\big|\; 1 \quad H \quad H$$
$$B' = 8 \quad H \quad 1 \;\big|\; 2 \quad 3 \quad 9 \;\big|\; H \quad 4 \quad H$$

$$A'' = 5 \quad 6 \quad 7 \;\big|\; H \quad H \quad H \;\big|\; 1 \quad 8 \quad 4$$
$$B'' = 2 \quad 3 \quad 9 \;\big|\; H \quad H \quad H \;\big|\; 4 \quad 8 \quad 1$$

$$A''' = 5 \quad 6 \quad 7 \;\big|\; 2 \quad 3 \quad 9 \;\big|\; 1 \quad 8 \quad 4$$
$$B''' = 2 \quad 3 \quad 9 \;\big|\; 5 \quad 6 \quad 7 \;\big|\; 4 \quad 8 \quad 1$$

交叉

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
- 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

交叉

实数编码的交叉方法

(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：

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

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

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

交叉

实数编码的交叉方法

(2) 算术交叉（arithmetical crossover）

- 整体算术交叉：先生成 n 个区间的随机数，则后代分别定义为：

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

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

$$a_1 = a_2 = \cdots = a_n$$

变异

整数编码的变异方法

- 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.
- 互换变异：随机选取染色体的两个基因进行简单互换.
- 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.
- 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.
- 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.
- 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.
- ...

变异

实数编码的变异方法

1. 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

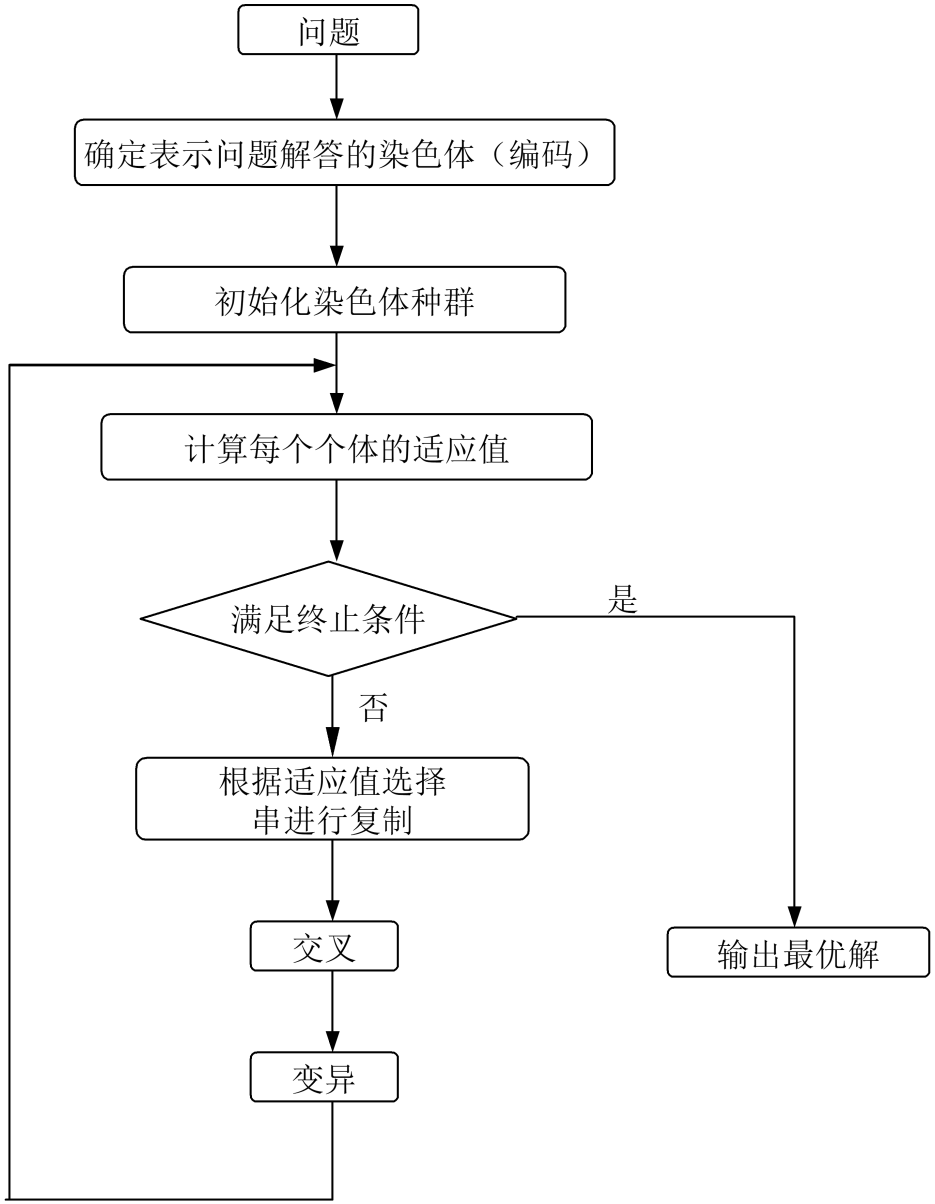
$$s' = \begin{cases} v_i, & i \neq k \\ v_k', & i = k \end{cases}$$

- 即

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

  变异产生的后代：$s' = (v_1, v_2, \cdots, v_k', \cdots, v_n)$

遗传算法的一般步骤


问题
确定表示问题解答的染色体（编码）
初始化染色体种群
计算每个个体的适应值
满足终止条件
是
否
根据适应值选择
串进行复制
交叉
变异
输出最优解

遗传算法的一般步骤

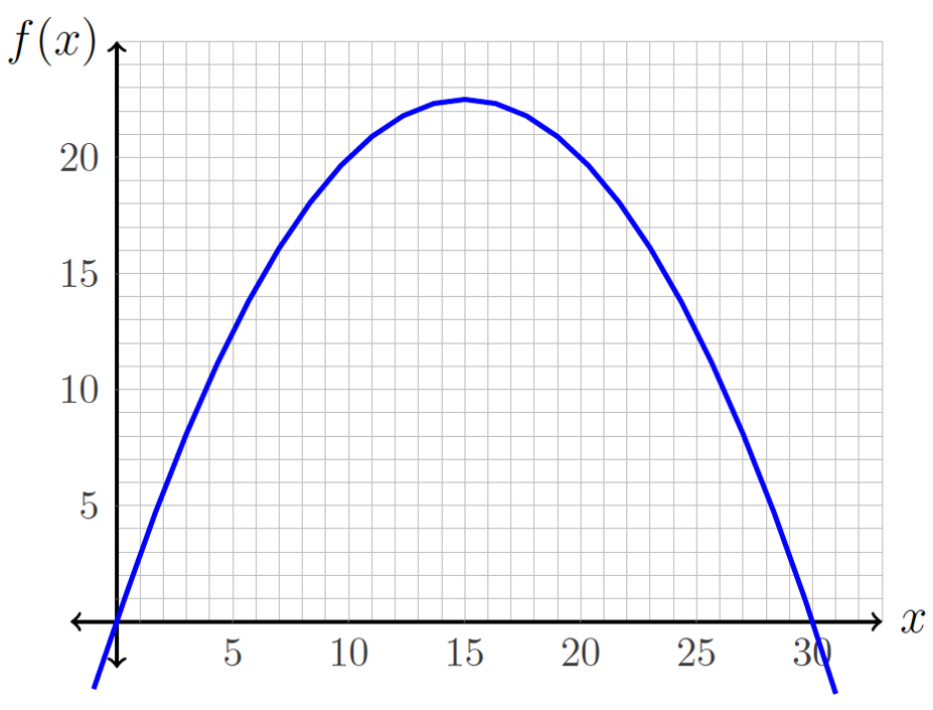
1. 使用随机方法或者其它方法，产生一个有 N 个染色体的初始群 $pop(1)$，$t := 1$;
2. 对群体中的每一个染色体 $pop_i(t)$，计算其适应值 $f_i = fitness(pop_i(t))$;
3. 若满足停止条件，则算法停止;
   - 否则，以概率 $p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$，从 $pop(t)$ 中随机选择一些染色体构成一个新种群 $newpop(t+1) = \{pop_j(t) | j = 1, 2, \cdots, N\}$
4. 以概率 p_c 进行交叉产生一些新的染色体，得到一个新的群体 $crosspop(t+1)$
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 $pop(t) = mutpop(t+1)$
6. 返回步骤 2

遗传算法的特点

- 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高效率搜索.
- 采用群体搜索策略，易于并行化.
- 仅用适应度函数值来评估个体，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.

遗传算法：示例

求函数最大值：$f(x) = \frac{-x^2}{10} + 3x$



遗传算法：示例

求函数最大值：$f(x) = \frac{-x^2}{10} + 3x$

- $x : 0 \sim 31 \quad \Rightarrow \quad 00000 \sim 11111$

Chromosome Number	Initial Population	x Value	Fitness Value $f(x)$	Selection Probability
1	01011	11	20.9	0.1416
2	11010	26	10.4	0.0705
3	00010	2	5.6	0.0379
4	01110	14	22.4	0.1518
5	01100	12	21.6	0.1463
6	11110	30	0	0
7	10110	22	17.6	0.1192
8	01001	9	18.9	0.1280
9	00011	3	8.1	0.0549
10	10001	17	22.1	0.1497
		Sum	147.6	
		Average	14.76	
		Max	22.4	

遗传算法：示例

求函数最大值：$f(x) = \frac{-x^2}{10} + 3x$

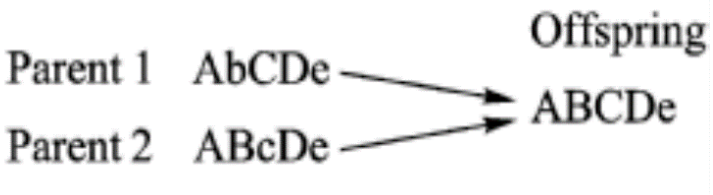
- $x : 0 \sim 31 \quad \Rightarrow \quad 00000 \sim 11111$

Chromosome Number	Mating Pairs	New Population	x Value	Fitness Value $f(x)$
5	01|100	01010	10	20
2	11|010	11100	28	5.6
4	0111|0	01111	15	22.5
8	0100|1	01000	8	17.6
9	0001|1	01010	10	20
2	1101|0	11011	27	8.1
7	10110	10110	22	17.6
4	01110	01110	14	22.4
10	100|01	10001	17	22.1
8	010|01	01001	9	18.9
			Sum	174.8
			Average	17.48
			Max	22.5

GA 的改进算法

1. 双倍体遗传算法

- 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.

GA 的改进算法

双倍体遗传算法设计:

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.

GA 的改进算法

2. 自适应遗传算法

- 基本思想
  - Srinvivas 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.

GA 的改进算法

2. 自适应遗传算法

- 算法步骤

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 $f = \frac{1}{ob}$，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 $\frac{N}{2}$ 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.
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 P_m，随机产生 $R(0,1)$，如果 $R < P_m$ 则对该染色体进行交叉操作.
7. 计算交叉/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；否则转步骤 4.

GA 的改进算法

2. 自适应遗传算法

- 适应的交叉概率与变异概率

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

- 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.

GA 示例：车间生产调度

- 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

GA 示例 1：流水车间调度

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

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

- 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．

GA 示例 1：流水车间调度

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.

GA 示例 1：流水车间调度

流水车间调度问题

- 问题的数学模型：
  - $c(j_i, k)$：工序 j_i 在机器 k 上的加工完工时间；
  - $\{j_1, j_2, \cdots, j_n\}$：工件的调度序列
  - 则 n 个工件、m 台机器的流水车间调度问题有：

$$
\begin{aligned}
&c(j_1, 1) = t_{j_1 1} \\
&c(j_1, k) = c(j_1, k-1) + t_{j_1 k}, \quad k = 2, \cdots, m \\
&c(j_i, 1) = c(j_{i-1}, 1) + t_{j_i 1}, \quad i = 2, \cdots, n \\
&c(j_i, k) = \max\{c(j_{i-1}, k), c(j_i, k-1)\} + t_{j_i k}, \quad i = 2, \cdots, n; k = 2, \cdots, m
\end{aligned}
$$

  - 最大流程时间：$c_{max} = c(j_n, m)$
  - 调度目标：确定 $\{j_1, j_2, \cdots, j_n\}$ 使得 c_{max} 最小

GA 示例 1：流水车间调度

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 .

(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：

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

GA 示例 1：流水车间调度

求解流水车间求解 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

GA 示例 1：流水车间调度

求解流水车间求解 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

穷举法

- 最优解：4-2-5-1-3
- 最优加工时间：213
- 最劣解：1-4-2-3-5
- 最劣加工时间：294
- 平均解的加工时间：265

GA 示例 1：流水车间调度

求解流水车间求解 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

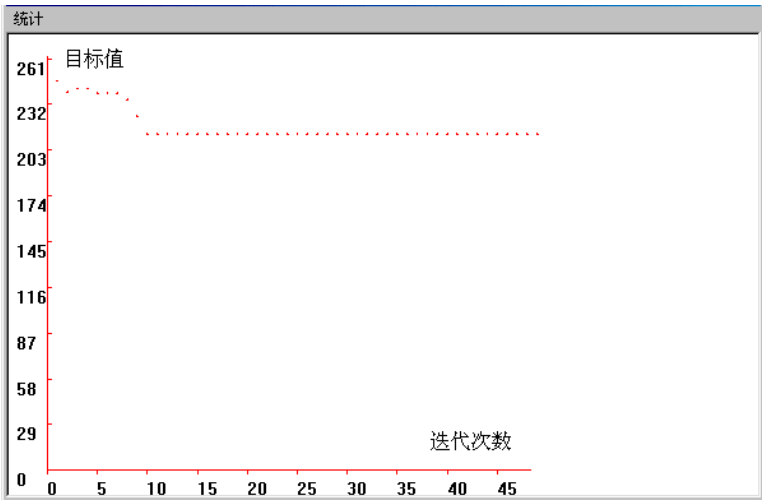
遗传算法

- 选交叉概率 $p_c = 0.6$，选变异概率 $p_m = 0.1$
- 种群规模为 20，迭代次数 $N = 50$

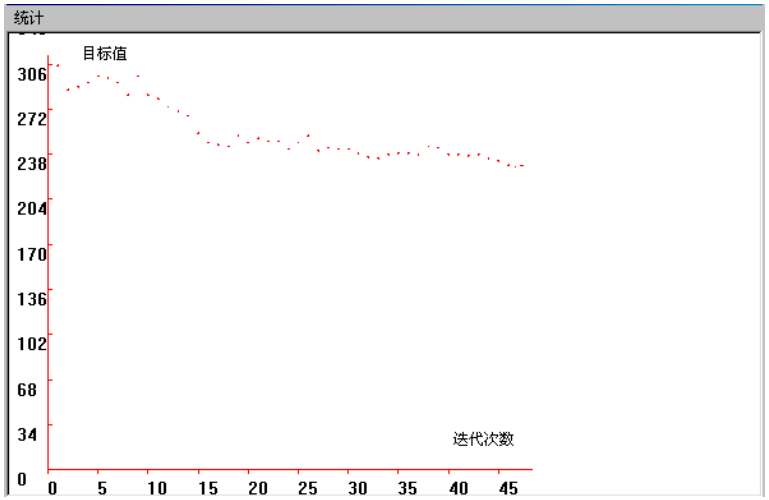
表: 遗传算法运行的结果

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

GA 示例 1：流水车间调度



(a) 最优解收敛图



(b) 平均值收敛图

图: GA 优化结果

GA 示例 2：混合流水车间调度

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

GA 示例 2：混合流水车间调度

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 $M_i, (i = 1, \cdots, S)$. 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

$$A_{S\times N} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{23} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{S1} & a_{S2} & \cdots & a_{SN} \end{bmatrix}$$

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

GA 示例 2：混合流水车间调度

混合流水车间调度问题的遗传算法编码方法

- $i = 1$，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：

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

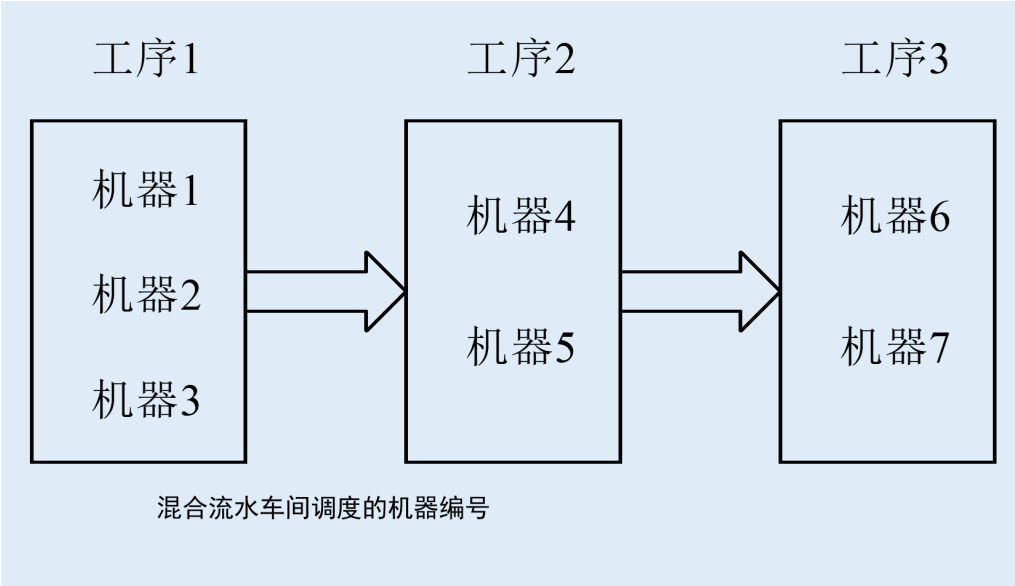
- 染色体的长度：$S \times N + S - 1$

GA 示例 2：混合流水车间调度

混合流水车间调度问题的遗传算法编码方法

- 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.

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



混合流水车间调度的机器编号

- 染色体：

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

GA 示例 2：混合流水车间调度

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）
   a. 种群成员按适应值从好到坏依次排列 $f_1 > f_2 > \cdots > f_N$
   b. 按下式分配复制概率：

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

GA 示例 2：混合流水车间调度

基于遗传算法的求解方法

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. $a'_{ij} = a_{ij} + d \times r$

GA 示例 2：混合流水车间调度

示例：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.

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

工件在每个机器上的加工时间

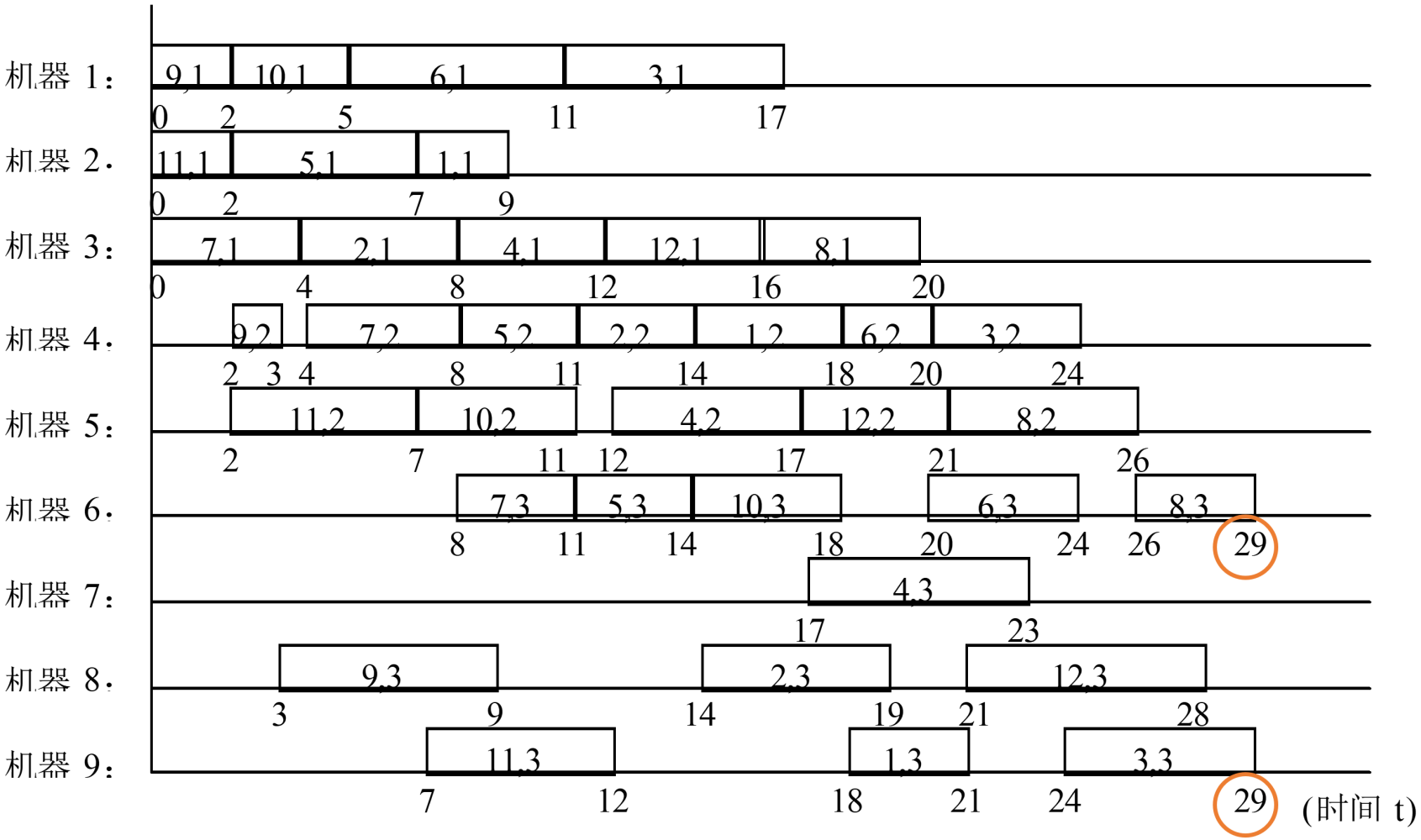
GA 示例 2：混合流水车间调度

- 算法中使用的参数为 $a = 0.07, P_c = 0.80, P_m = 0.01$
- 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

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

GA 示例 2：混合流水车间调度



混合流水车间调度结果甘特图

内容组织

差分进化算法 (Differential Evolution Algorithm)

差分进化算法的概念:

- 差分进化算法 (DE) 或称为差分演化算法、微分进化算法、微分演化算法、差异演化算法.
- 1995 年由 Rainer Storn 和 Kenneth Price 提出.
- DE 是一种采用实数矢量编码在连续空间中进行随机搜索的优化算法.
- 差分进化算法是一种基于实数编码的具有保优思想的贪婪遗传算法.

差分进化算法 (Differential Evolution Algorithm)

差分进化算法的基本要素：

- 参数编码
- 初始群体的产生
- 适应度函数的设计
- 差分操作设计
- 控制参数设置

差分进化算法 (Differential Evolution Algorithm)

参数编码

- DE 采用实数编码方式.
- DE 中的个体 i 表示为: $x_i^t = [x_{i,1}^t, x_{i,2}^t, \cdots, x_{i,D}^t]$

  D 表示问题空间维数

  t 表示进化代数

  $x_{i,j}^t$ 为实数
- 不必进行数制转换，可直接在解的表现型上进行进化操作.

差分进化算法 (Differential Evolution Algorithm)

初始种群的产生

- 根据问题固有知识，把握解参数变量 $x_{i,j}^t$ 的取值范围，然后在此范围内设定初始群体．通常寻找初始种群的一个方法是从给定边界约束内的值中随机选择．
- 设参数变量的界限为 $x_j^{(L)} < x_j < x_j^{(U)}$，则：

$$x_{i,j}^0 = rand[0,1] \times (x_j^{(U)} - x_j^{(L)}) + x_j^{(L)}$$

$$i = 1, 2, \cdots, NP \quad j = 1, 2, \cdots, D$$

- NP 表示种群规模，$rand[0,1]$ 表示在 $[0,1]$ 之间产生的均匀随机数．

差分进化算法 (Differential Evolution Algorithm)

适应度函数的设计

- 在差分进化算法中，差分操作主要通过适应度函数的导向来实现的，它是用来评估个体相对于整个群体的优劣的相对值的大小.
- 通常根据具体问题定义适应度函数，最直观的方法是直接将待求解优化问题的目标函数作为适应度函数.
- 具体设计方法与遗传算法相似.

差分进化算法 (Differential Evolution Algorithm)

差分操作设计：变异

- 对每个目标个体 $x_i^t, i = 1, 2, \cdots, NP$，其变异个体 v_i^{t+1} 的产生方式根据差向量的个数以及父代基向量的选取方式的不同分如下几种：

$$v_i^{t+1} = x_{r_1}^t + F \times (x_{r_2}^t - x_{r_3}^t)$$

$$v_i^{t+1} = x_{best}^t + F \times [(x_{r_1}^t - x_{r_2}^t) + (x_{r_3}^t - x_{r_4}^t)]$$

$$v_i^{t+1} = x_{best}^t + F \times (x_{r_1}^t - x_{r_2}^t)$$

$$v_i^{t+1} = x_i^t + \lambda \times (x_{best}^t - x_i^t) + F \times (x_{r_1}^t - x_{r_2}^t)$$

$$v_i^{t+1} = x_{r_1}^t + F \times [(x_{r_2}^t - x_{r_3}^t) + (x_{r_4}^t - x_{r_5}^t)]$$

  - 其中，r_1, r_2, r_3 和 r_4 表示随机产生的 $[1, NP]$ 之间互异且不等于目标个体序号 i 的自然数．$F \in [0, 2]$ 表示缩放比例因子，是一个实常数因数，控制偏差变量的放大作用．

差分进化算法 (Differential Evolution Algorithm)

差分操作设计：交叉

- 对目标个体 x_i^t 和变异个体 v_i^{t+1} 实施交叉操作生成试验个体 u_i^{t+1} 的过程称为交叉，$u_i^{t+1} = [u_{i,1}^{t+1}, u_{i,2}^{t+1}, \cdots, u_{i,D}^{t+1}]$.
- DE 交叉操作分二项式（bin）交叉和指数（exp）交叉两种交叉方式.
  - 二项式（bin）交叉：

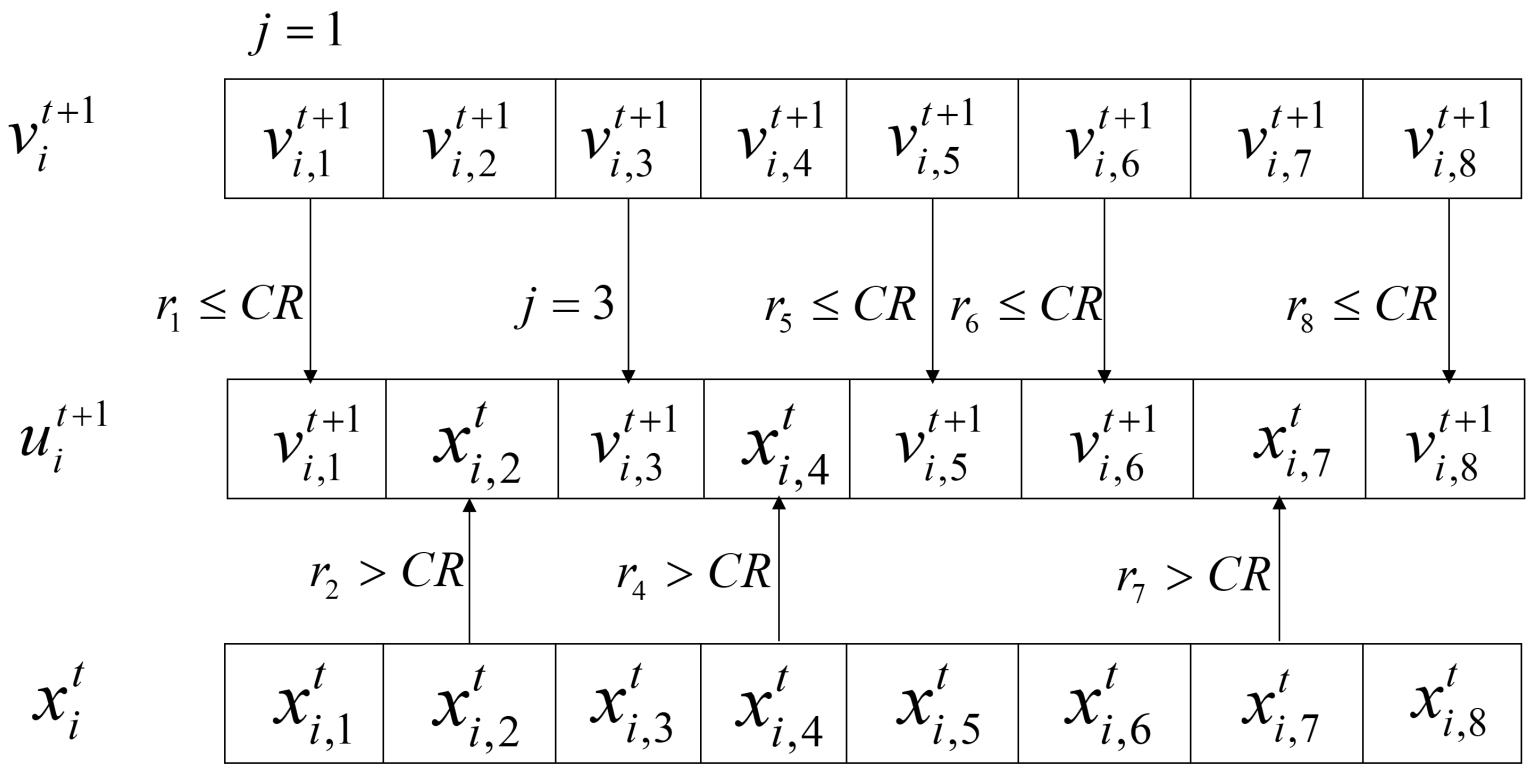
$$u_{i,j}^{t+1} = \begin{cases} v_{i,j}^{t+1}, & if\ r_j \le CR\ or\ j = rnbr\_i \\ x_{i,j}^{t}, & otherwise \end{cases}$$

  - 其中，r_j 为第 j 个 $[0,1]$ 之间的随机数；$rnbr\_i$ 为 $[1,D]$ 之间的随机自然数，它确保了 u_i^{t+1} 至少从 v_i^{t+1} 获得一个参数；CR 为交叉概率，取值范围为 $[0,1]$.

差分进化算法 (Differential Evolution Algorithm)

差分操作设计：交叉

- 二项式交叉（例） $D = 8,\ rnbr\_i = 3$



差分进化算法 (Differential Evolution Algorithm)

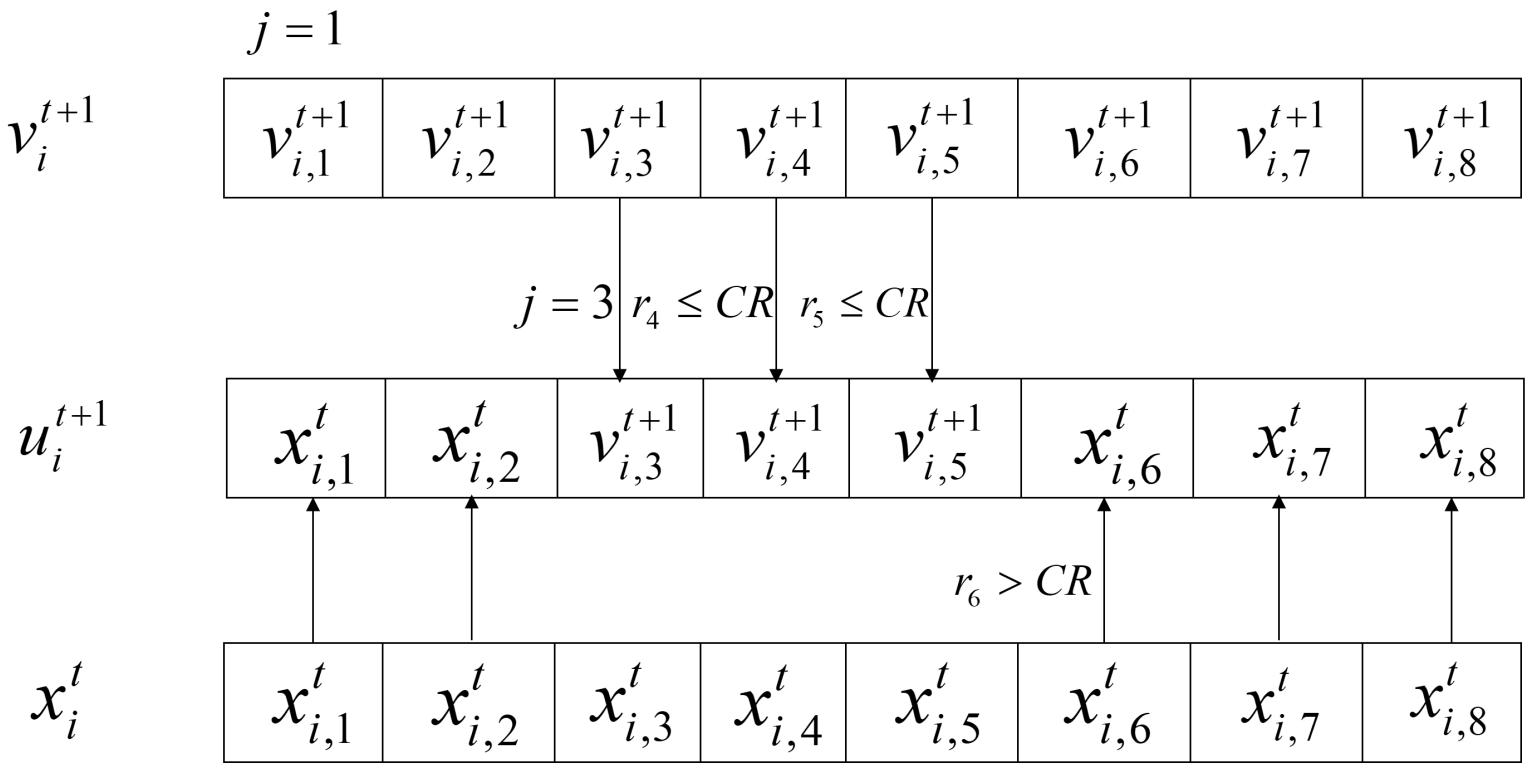
差分操作设计：交叉

- 指数交叉
  - 指数（exp）交叉．对第 j 位参数：
  - 若 $j < rnbr\_i$：取目标个体上的第 j 位参数作为试验个体上的第 j 位参数；
  - 若 $j = rnbr\_i$：取变异个体上的第 j 位参数作为试验个体上的第 j 位参数；
  - 若 $j > rnbr\_i$：随机产生 $[0, 1]$ 之间的随机实数 r_j，若 $r_j \leq CR$，取变异个体上的第 j 位参数作为试验个体上的第 j 位参数，否则，试验个体上剩下的所有参数都从目标个体继承．

差分进化算法 (Differential Evolution Algorithm)

差分操作设计：交叉

- 指数交叉（例） $D = 8,\ rnbr\_i = 3$



差分进化算法 (Differential Evolution Algorithm)

差分操作设计：选择

- DE 采用贪婪准则在 x_i^t 和 u_i^{t+1} 之间进行选择，产生下一代个体 x_i^{t+1}：

$$x_i^{t+1} = \begin{cases} u_i^{t+1}, & if\ f(u_i^{t+1}) > f(x_i^t) \\ x_i^t, & otherwise \end{cases}$$

- $f(\cdot)$ 表示适应度函数．上式是针对最大化问题而言，若是最小化问题，那么选择试验个体作为子代个体的条件是 $f(u_i^{t+1}) < f(x_i^t)$．

差分进化算法 (Differential Evolution Algorithm)

控制参数设置

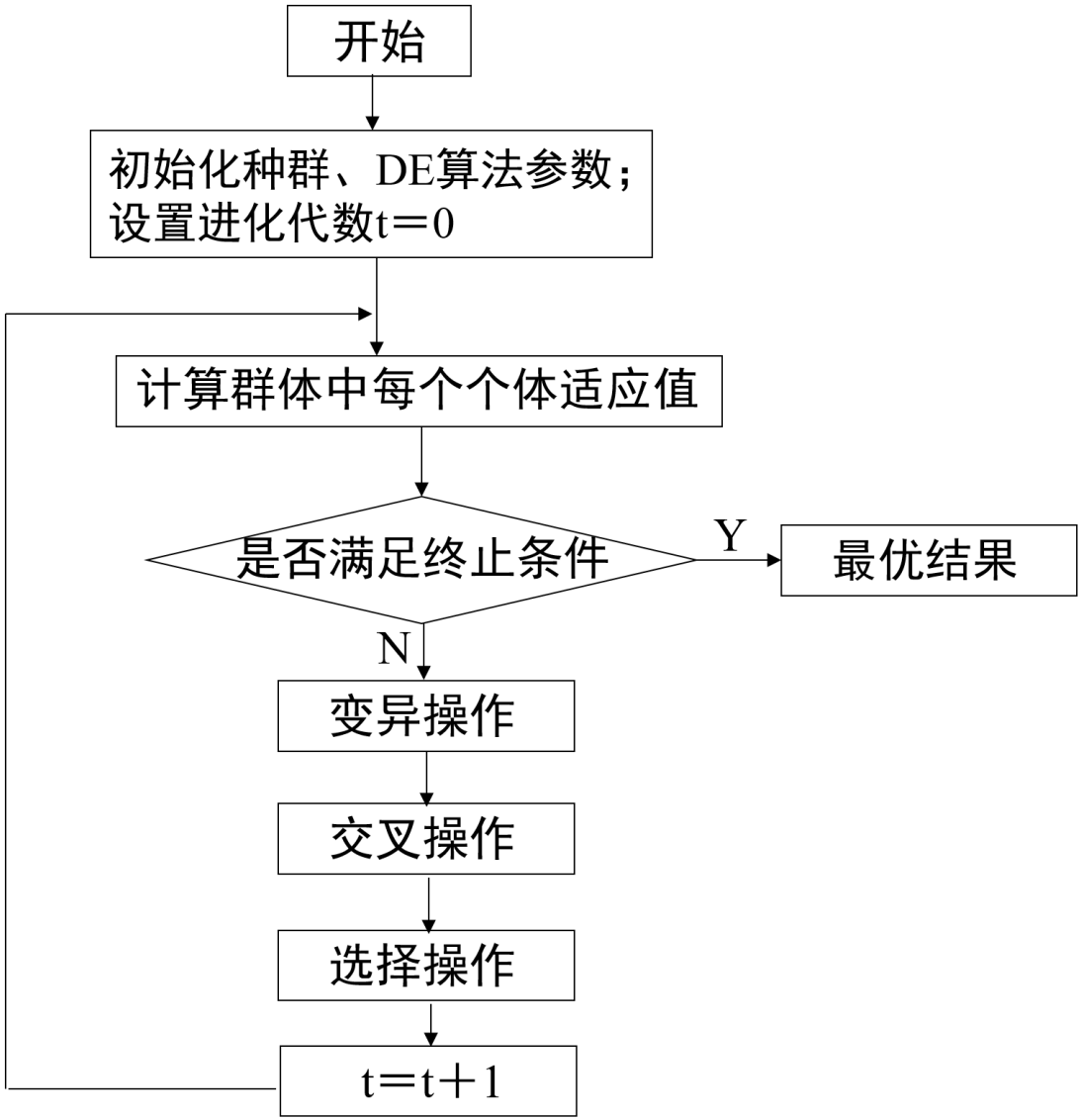
- DE 的搜索性能取决于算法全局探索和局部开发能力的平衡，而这在很大程度上依赖于算法的控制参数的选取.
- 种群规模. 种群规模 NP 必须满足 $NP \geq 4$ 以确保 DE 具有足够的不同的变异向量，根据经验，NP 的合理选择在 $5D \sim 10D$（D 为问题空间的维数）之间 .
- 缩放比例因子. 缩放比例因子 $F \in [0, 2]$ 是一个实常数，它决定偏差向量的放大比例. F 越大，算法更容易逃出局部极小点，更容易收敛到全局最优点. 一般取 0.7.
- 交叉概率. 交叉概率 CR 是一个范围在 $[0, 1]$ 的实数，它控制着一个试验个体参数来自于变异个体的概率. CR 越大，算法更容易收敛，但易发生早熟现象. CR 的一个较好的选择是 0.3.

差分进化算法 (Differential Evolution Algorithm)

控制参数设置

- 最大迭代代数．一般而言，最大迭代次数越大，最优解越精确，但计算时间越长，要根据具体问题设定．
- 终止条件．除最大进化代数可作为 DE 的终止条件外，有时还需要其它判定准则，一般当目标函数值小于阈值时程序终止，阈值常选为 $10 \sim 6$．
- F, CR 与 NP 一样，在搜索过程中是常数，一般 F 和 CR 影响搜索过程的收敛速度和鲁棒性，它们的优化值不仅依赖于目标函数的特性，还与 NP 有关．
- 通常可通过在对不同值做一些试验之后利用试验和结果误差找到 F, CR 与 NP 的合适值．

差分进化算法流程


开始
初始化种群、DE算法参数；
设置进化代数t=0
计算群体中每个个体适应值
是否满足终止条件
Y
最优结果
N
变异操作
交叉操作
选择操作
t=t+1

差分进化算法特点

- 算法原理简单，容易实现；算法通用，不依赖于问题信息.
- 分散搜索，能在整个解空间进行全局搜索.
- 保优搜索，具有记忆个体最优解的能力.
- 协同搜索，具有利用个体局部信息和群体全局信息指导算法进一步搜索的能力.
- 直接对结构对象进行操作，不存在对目标函数的限定（如要求函数可导或连续）.

Question & Answer