第5章 智能优化算法

目录

§1 遗传算法

- 遗传算法概念

遗传算法简称GA（Genetic Algorithms）是一种借鉴生物界自然选择和自然遗传机制的自适应全局优化搜索算法。它模拟自然界生物进化的发展规律，在人工系统中实现待定目标的优化。在机器学习、模式识别和控制系统优化等不同领域得到广泛应用。

- 遗传算法发展历程

20世纪60年代，基于人们对自然和人工自适应系统的研究，美国Michigan(密歇根州)大学的J.Holland教授首次提出遗传算法。

70年代，K.A.De Jong 基于遗传算法的思想，进行了大量的纯数值函数优化计算实验，为后续遗传学习的发展及应用奠定了坚实基础。

80年代，D.E.Goldberg总结了遗传算法研究的主要成果，对遗传算法及其应用作了全面而系统的论述。

- 遗传算法基本思想

➢ 基于生物界遗传学说，把问题的参数用基因表示，把问题的解用染色体表示，从而得到一个由具有不同染色体的个体组成的群体。

➢ 该群体在特定环境中生存竞争，更加适应环境的染色体进行复制，并通过交叉、变异过程产生更适应环境的新一代染色体群。

➢ 群体的染色体不断适应环境，不断进化，最终收敛到一组最适应环境的相似个体，即得到问题最优解。

- 遗传算法相关概念

➢ 个体与种群

✓ 个体：模拟生物个体而对问题中的对象的一种称呼，表示问题的一个可行解。

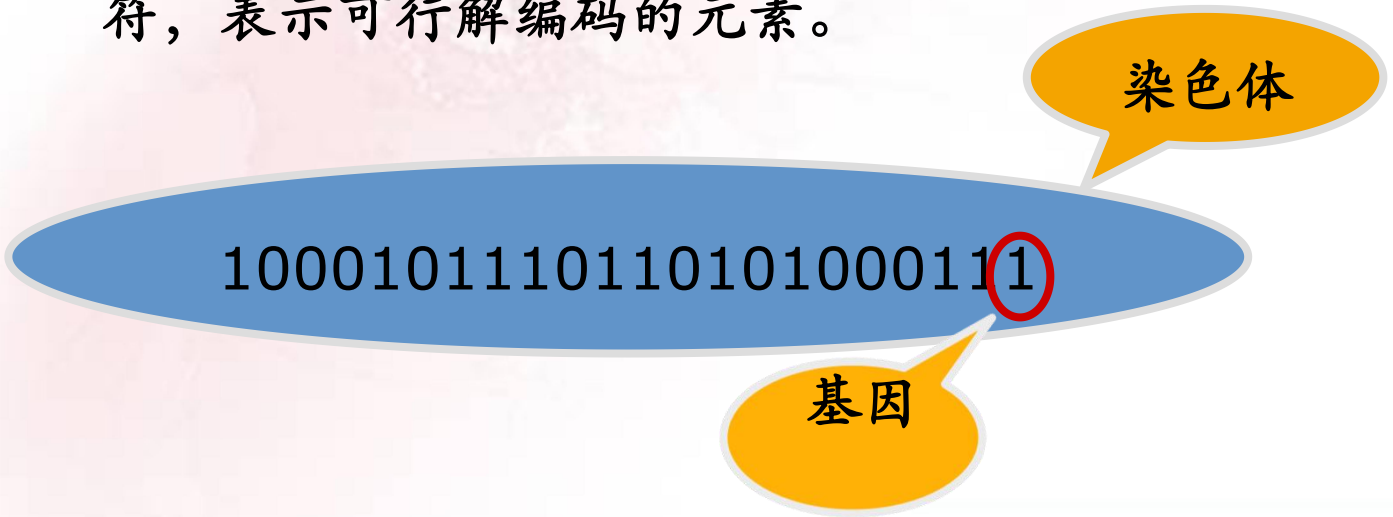
✓ 种群：模拟生物种群而由若干个体组成的群体，表示问题的可行解集。

- 遗传算法相关概念

➢ 染色体与基因

✓ 染色体：个体的某种字符串形式的编码表示，也即是可行解的编码。

✓ 基因：染色体的基本组成单位，也即字符串中的字符，表示可行解编码的元素。



- 遗传算法相关概念

➢ 适应度与适应度函数

✓ 适应度：借鉴生物个体对环境的适应程度，而用来表征问题中的个体对象优劣程度。

✓ 适应度函数：问题中全体个体与其适应度之间的对应关系，是遗传算法中指导搜索的评价函数。

适应度函数的选取直接影响遗传算法的收敛速度以及能否找到最优解。一般而言，适应度函数是由目标函数变换而成。

1.2 遗传算法基本原理

- 遗传算法执行步骤

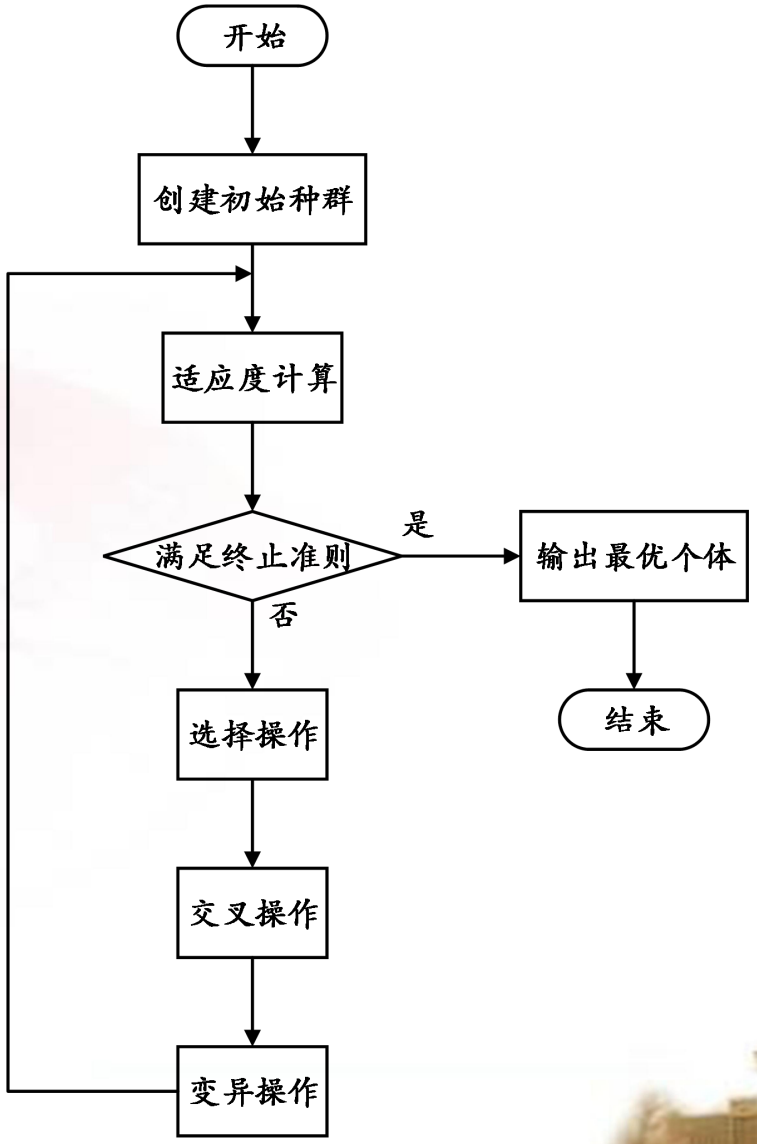
（1）初始化。设置进化代数计数器$n=0$，设置最大进化代数NG，随机生成NP个个体作为初始种群$P(0)$。

（2）个体评价。计算种群$P(0)$中各个个体的适应度。

（3）选择运算。根据选择策略，选择一些优良个体遗传到下一代。

（4）交叉运算。对选中的个体以概率P_c作交叉操作。

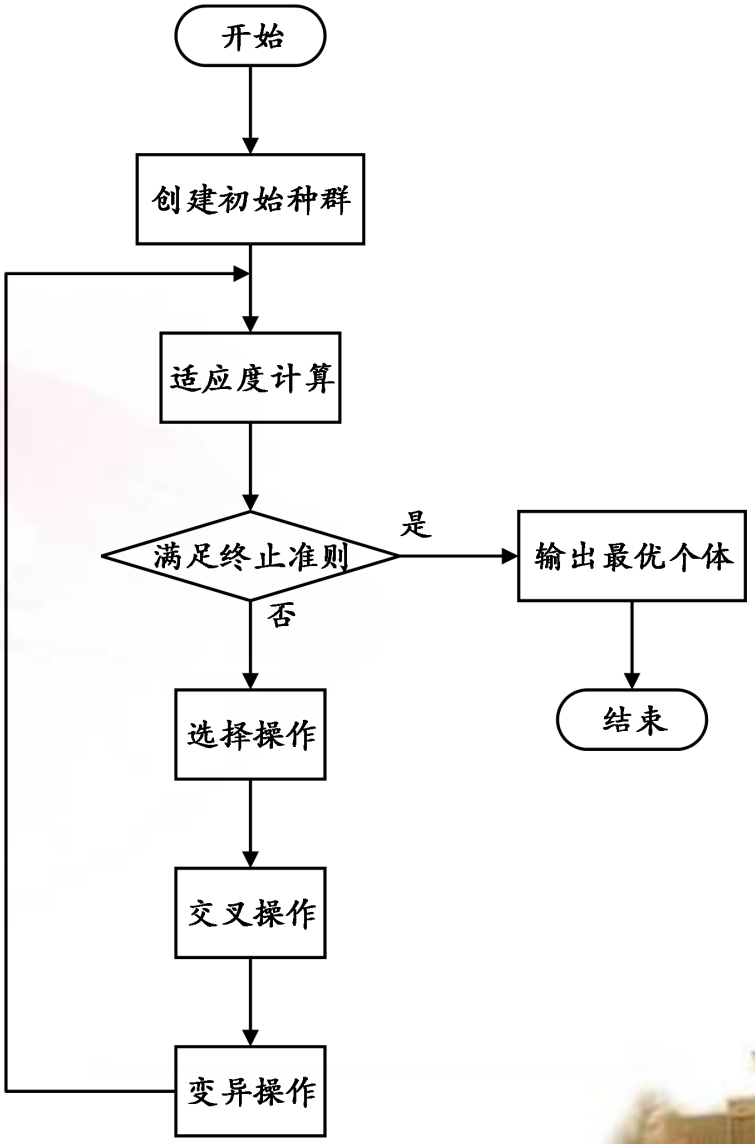
（5）变异运算。对选中的个体以概率P_m作变异操作。



- 遗传算法执行步骤

（6）循环操作。第t代种群$P(t)$经过选择、交叉和变异运算后得到下一代种群$P(t+1)$。计算其中各个个体的适应度，并进行排序，准备进行下一次遗传操作。

（7）停止准则。当$n \leq NG$时，则$n=n+1$，转到步骤（2）；若$n>NG$，则将此进化过程中得到的具有最大适应度的个体作为最优解输出，终止计算。

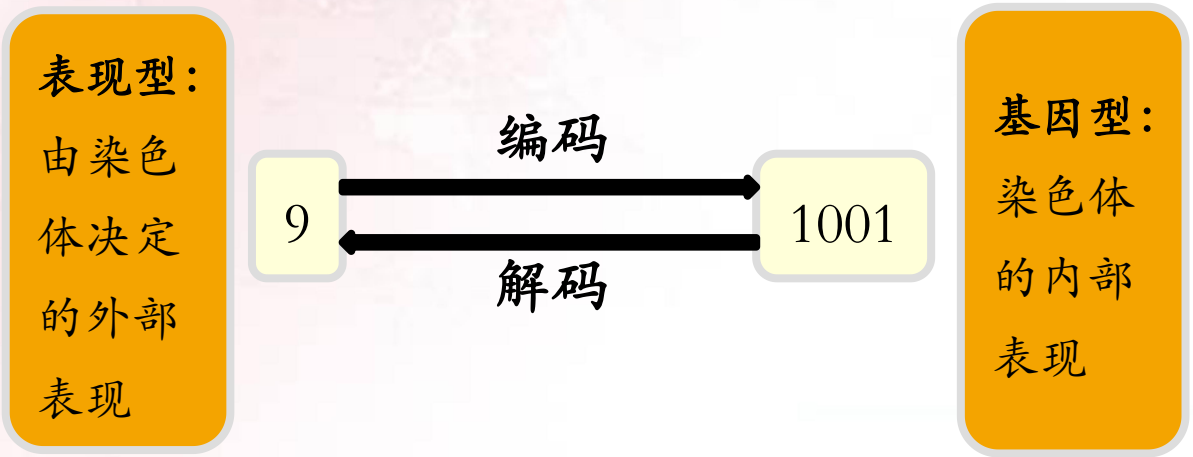


- 遗传算法构成要素

➢ 遗传编码

遗传编码将对象抽象为由特定符号按一定顺序组成的字符串，常用的编码机制有二进制编码、浮点数编码、符号编码等。

✓ 二进制编码方法：



- 遗传算法构成要素

✓ 二进制编码方法精度

假定某参数的取值范围是$[u_{min}, u_{max}]$,用长度为λ的二进制编码串来表示该参数，则共产生2^{λ}种不同的编码，二进制编码的编码精度可以表示为：

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

由上式可以得到，二进制编码位数越长，编码精度越高，意味着所得到的解的质量越高，但也会导致遗传操作的计算量也更大，算法更加耗时，因此解决实际问题时，编码位数需要适当选择。

- 遗传算法构成要素

➢ 遗传操作

遗传操作是优选优良个体的“选择”、个体间交换基因个体的“交叉”、个体基因信息突变而产生新个体的“变异”这三种变换的统称

①选择：根据个体适应度，按照一定的规则或方法，从第t代群体$P(t)$中选择一些优良个体遗传到下一代群体$P(t+1)$中。

②交叉：将群体$P(t)$中的各个个体随机搭配，对每一对个体，以某一概率（交叉概率P_c）交换它们之间的部分染色体。

③变异：对群体$P(t)$中的每个个体，以某一概率（变异概率P_m）将某个或某些基因值改变为其他的等位基因值。

- 遗传算法构成要素

➢ 遗传操作

①选择策略：最常用的选择策略是正比选择(Proportional Selection)策略，即每个个体被选中进行遗传运算的概率为该个体的适应值和群体中所有个体适应值总和的比例.

对于个体i，设其适应值为F_i，种群规模为NP，则该个体的选择概率为：

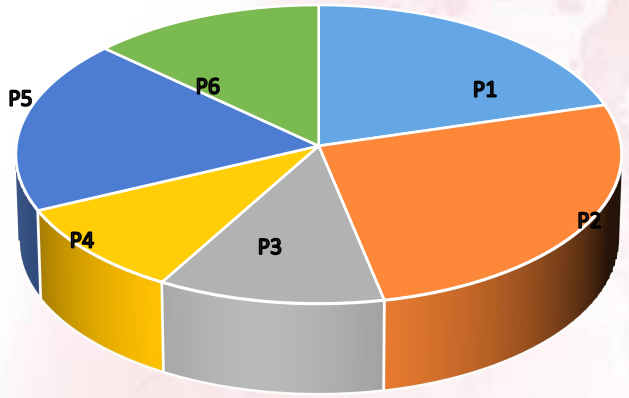
$$P_i = \frac{F_i}{\sum_{i=1}^{NP} F_i}$$

得到选择概率后，采用“轮盘赌”选择法来实现选择操作。

- 遗传算法构成要素

➢ 遗传操作

选择策略—“轮盘赌”选择法：



令$PP_0 = 0$，$PP_i = \sum_{j=1}^{i} P_j$

每次转轮时，随机产生一个$\xi_k \in (0,1)$，当$PP_{i-1} < \xi_k \leq PP_i$，则选择个体$i$。

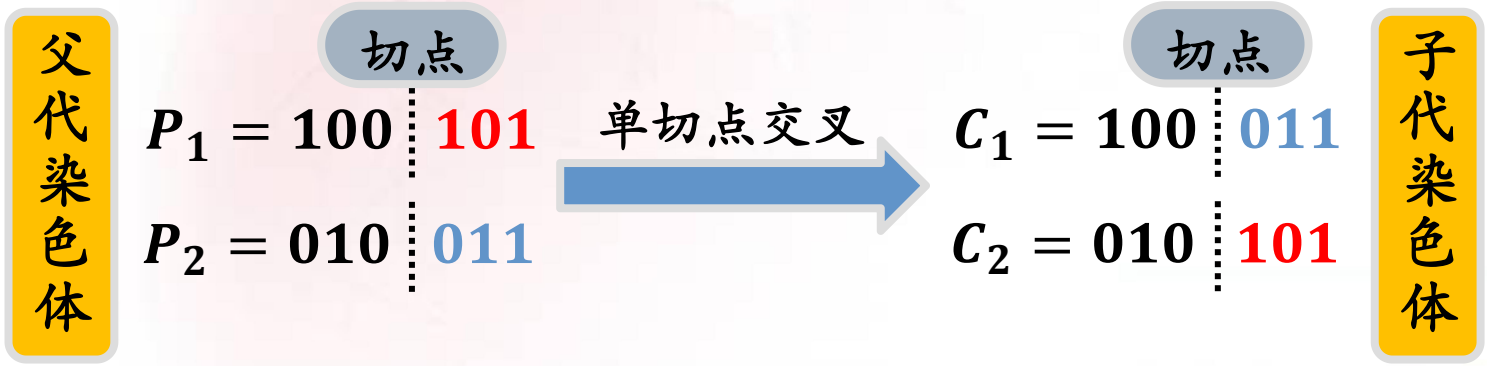
- 遗传算法构成要素

➢ 遗传操作

②交叉操作：从种群中选出两个个体P_1和P_2，随机选取交叉位置，在交叉位置处，以一定的交叉概率P_c相互交换各自的部分基因，从而形成新的一对个体C_1和C_2。

- 单切点交叉:

在父代染色体中随机选择一个切点，将切点右侧的子串分别交换，得到新的两个个体。



- 遗传算法构成要素

➢ 遗传操作

在单切点交叉操作中，交叉点的位置选择可能带来较大偏差，且染色体的末尾基因总是被交换，在实际应用中采用较多的是双切点交叉。

- 双切点交叉：

在父代染色体中随机选择两个切点，将两切点中间的子串分别交换，得到新的两个个体。

- 遗传算法构成要素

➢ 遗传操作

③变异操作：变异是在种群中按照变异概率P_m任选若干基因位改变其位值，对于0-1编码来说，就是反转位值。

类似于自然界中的基因突变，变异实际上是子代因为小概率扰动产生的变化。因此，变异概率一般设定为一个比较小的数。

- 遗传算法参数说明

➢ 种群规模NP

种群规模将影响遗传优化的最终结果以及遗传算法的执行效率，当种群规模NP过小时，遗传优化性能一般不会太好。采用较大的种群规模可以降低遗传优化陷入局部最优解，但较大的种群规模又意味着计算复杂度较高。一般NP取10~200。

➢ 最大进化代数NG

最大进化代数NG是表示遗传算法运行结束条件的一个参数，它表示遗传算法运行到指定进化代数之后就停止运行，并将当前最优个体作为最优解输出。NG的取值一般在100~1000。

- 遗传算法参数说明

➢ 交叉概率P_c

交叉概率P_c控制着交叉操作被使用的频度。较大的交叉概率可以加强遗传算法开辟新的搜索区域的能力，但也会加大高性能个体遭到破坏的可能性。较小的交叉概率可能导致遗传算法陷入迟钝状态。P_c一般取0.25~1.00。

➢ 变异概率P_m

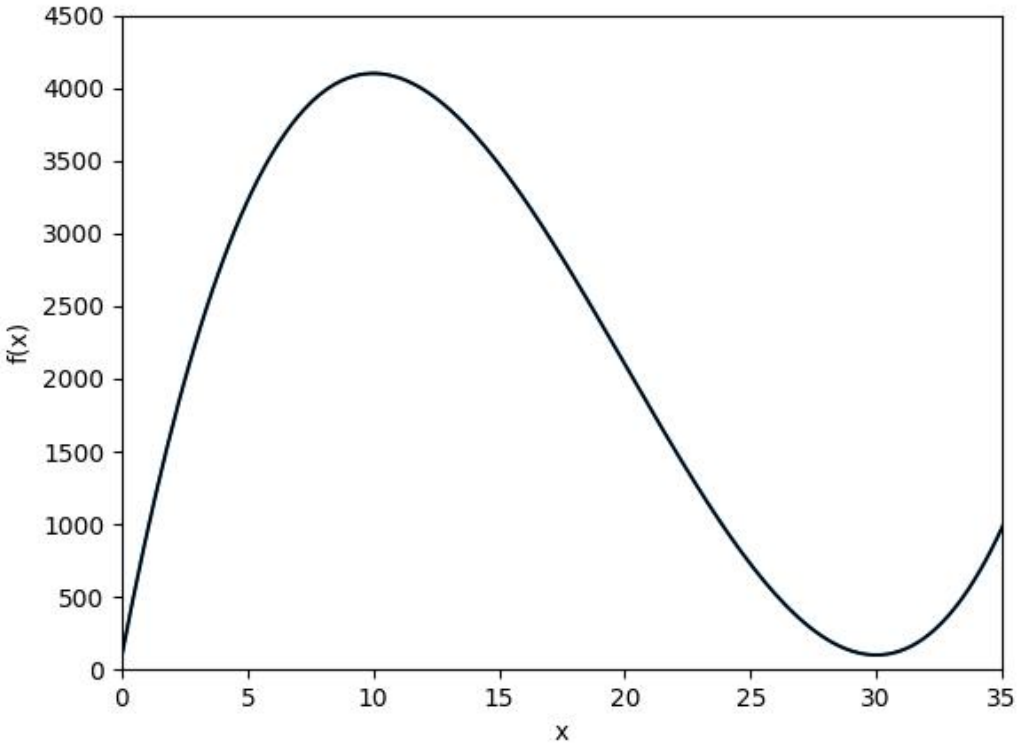
变异在遗传算法中属于辅助性搜索操作，它的主要目的是避免出现交叉操作后的子代适应度不再比父代好，但仍未达到全局最优解的情况（早熟收敛）。但P_m不宜过大，因为可能导致遗传算法搜索方向过于随机，一般取0.001~0.1。

1.3 遗传算法计算实例

- 求解以下无约束优化问题

$$\max \ f(x) = x^3 - 60x^2 + 900x + 100$$

$$x \in [0, 32]$$



利用求导方式得到解析解：

极大值点：$x_1 = 10$

极小值点：$x_2 = 30$

- 求解以下无约束优化问题

$$\max \ f(x) = x^3 - 60x^2 + 900x + 100$$
$$x \in [0, 32]$$

➢ 确定染色体长度：

设染色体长度为L，若编码精度为1，则染色体长度计算方法如下：

$$\frac{32 - 0}{2^L - 1} = 1$$

得$L \approx 5$，取染色体长度为5，可满足精度要求。

➢ 选取适应度函数：

在此问题中，可将目标函数直接作为适应度函数。

1.3 遗传算法计算实例

- 求解以下无约束优化问题

$$\max \ f(x) = x^3 - 60x^2 + 900x + 100$$

$$x \in [0, 32]$$

➢ 初始化：

设种群规模$NP = 4$，最大进化代数$NG = 10$，初始种群如下：

j	编码	x_j	$f(x_j)$	P_j	PP_j
1	10011	19	2399	0.244	0.244
2	00101	5	3225	0.328	0.572
3	11010	26	516	0.053	0.625
4	01110	14	3684	0.375	1

$$S_0 = \sum f(x_j) = 9824, \quad \bar{f}_0 = \frac{S}{NP} = 2456$$

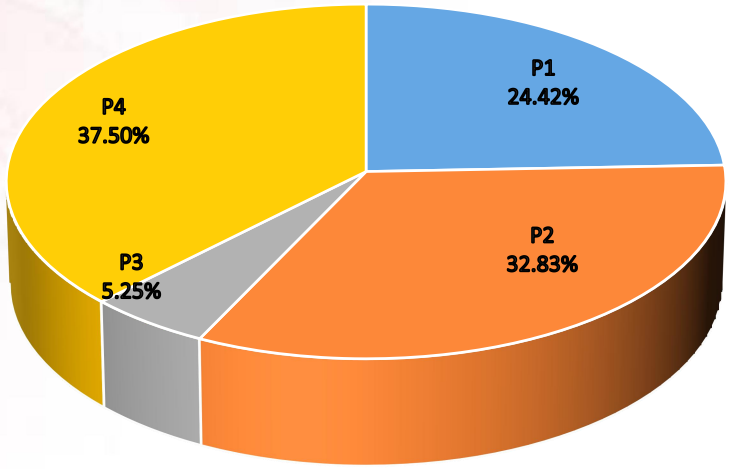
1.3 遗传算法计算实例

- 求解以下无约束优化问题

$$\max \ f(x) = x^3 - 60x^2 + 900x + 100$$
$$x \in [0, 32]$$

➢ 选择运算：

采用“轮盘赌”选择法，每次转轮时，随机产生一个$\xi_k \in (0,1)$，当$PP_{i-1} < \xi_k \leq PP_i$，则选择个体$i$，以相邻选择的两个体作为父代双亲。



1.3 遗传算法计算实例

- 求解以下无约束优化问题

$$\max \ f(x) = x^3 - 60x^2 + 900x + 100$$
$$x \in [0, 32]$$

➢ 交叉与变异运算：

采用单切点交叉操作，交叉概率设为0.9；在此求解过程中，由于种群规模过小，因此不考虑变异。

父代	切点	交叉否	子代	x_j	$f(x_j)$
[4]01110	3	是	01101	13	3857
[2]00101			00110	6	3556
[4]01110	2	是	01011	11	4071
[1]10011			10110	22	1508

$$S_1 = \sum f(x_j) = 12992, \quad \bar{f}_1 = \frac{S}{NP} = 3248$$

1.3 遗传算法计算实例

- 求解以下无约束优化问题

$$\max \ f(x) = x^3 - 60x^2 + 900x + 100$$
$$x \in [0, 32]$$

➢ 第一代种群与初始种群对比：

初始种群

编码	x_j	$f(x_j)$
10011	19	2399
00101	5	3225
11010	26	516
01110	14	3684

$$S_0 = \sum f(x_j) = 9824$$
$$\bar{f}_0 = \frac{S}{NP} = 2456$$

第一代种群

编码	x_j	$f(x_j)$
01101	13	3857
00110	6	3556
01011	11	4071
10110	22	1508

$$S_0 = \sum f(x_j) = 12992$$
$$\bar{f}_0 = \frac{S}{NP} = 3248$$

通过对比可以观察到：经过一代进化后，整个种群的平均适应度在提升，个体在逐渐逼近最优解。

- 求解以下无约束优化问题

$$\max \ f(x) = x^3 - 60x^2 + 900x + 100$$
$$x \in [0, 32]$$

➢ 循环操作：

又以第一代群体作为父代，再次进行选择、交叉、变异等操作生成第二代群体，之后重复进行该操作，直至达到终止条件。

➢ 输出最优个体：

在进行若干代进化后，所有个体会逐渐逼近一个最优解，达到终止条件后输出该最优值。

1.4 遗传算法的特点

- 遗传算法是一种基于概率的搜索技术，其选择、交叉、变异都是以一种概率的方式来进行，从而增加了其搜索过程的灵活性。

- 遗传算法的适应性强，除了需要知道适应性函数外，几乎不需要再知道其他先验知识。

- 遗传算法长于全局搜索，它不受搜索空间的限制，不要求连续性，能以很大概率从离散的、多极值的、含有噪声的高维问题中找到全局最优解。

1.5 遗传算法的改进方向

遗传算法的主要特征在于群体搜索策略和简单的遗传算子，这使得遗传算法具有强大的全局搜索能力，从而成为一种具有良好适应性和可规模化的求解方法。但大量实践和研究表明，标准遗传算法存在局部搜索能力差和“早熟收敛”的缺陷。

- 对遗传算法本身进行优化设计,主要包括编码机制、选择策略、交叉算子、变异算子、和参数设计（种群规模、交叉概率和变异概率）等。

- 将遗传算法与其他算法融合，例如差分进化算法、免疫算法、粒子群算法等，此改进方向可以综合遗传算法和其他算法的优点，提高运行效率和求解质量。

§2 粒子群算法

- 粒子群算法概念

粒子群算法（Particle Swarm Optimization）是一种模拟鸟群捕食行为，基于群体智能的迭代随机搜索算法。在函数优化、神经网络训练、模式识别和模糊系统控制等领域得到了广泛的应用。

- 粒子群算法发展历程

1990年，生物学家Frank Heppner提出了鸟群聚集模型，能够非常接近地模拟出鸟群飞行的现象。

1995年，美国社会心理学家James Kennedy和电气工程师Russell Eberhart共同提出了粒子群算法，该算法的提出是受对鸟类群体行为进行建模与仿真的研究结果的启发。他们的模型和仿真算法主要对Frank Heppner的模型进行了修正，以使粒子飞向解空间并在最优解处降落。

2011年，J. Kennedy与R. Eberhart合著《群体智能》，将粒子群算法的影响进一步扩大。

- 粒子群算法生物学基础—鸟群觅食行为

设想：一群鸟在区域中随机搜索食物，所有鸟都知道自己当前位置离食物有多远，那么什么是最简单有效的搜索策略呢?

➢ 鸟群协作机制：鸟群成员可以通过个体之间的信息交流与共享获得其他成员的发现与飞行经历。这是鸟群觅食的决定优势，对鸟群而言，最简单有效的搜索策略就是搜寻目前离食物最近的鸟的周围区域。

- 粒子群算法基本思想

➢ 将优化问题的搜索空间类比于鸟类的飞行空间，将每只鸟抽象为一个无质量、无体积的粒子，用以表征问题的一个可行解。鸟类离食物的距离即是粒子的适应度，优化问题的最优解等同于鸟类寻找的食物源。

➢ 粒子群算法为每个粒子制定了与鸟类运动类似的行为规则，个体之间同样存在协作和信息共享，使整个粒子群的运动表现出与鸟类觅食相似的特性，从而可以求解复杂的优化问题。

- 粒子群算法相关概念

假设在一个D维的目标搜索空间中，有N个粒子组成的一个群落：

➢ 粒子的位置和速度

✓ 位置：粒子在空间中的位置，即是一个可行解，第i个粒子的位置表示为一个D维的向量：

$$X_i = (x_{i1}, x_{i2}, \cdots, x_{iD}),\ i = 1, 2, \cdots N$$

✓ 速度：粒子的飞行速度，第i个粒子的速度也可表示为一个D维的向量：

$$V_i = (v_{i1}, v_{i2}, \cdots, v_{iD}),\ i = 1, 2, \cdots N$$

- 粒子群算法相关概念

➢ 个体极值和全局极值

✓ **个体极值：** 第i个粒子迄今为止搜索到的最优位置，记为

$$P_{best} = (p_{i1}, p_{i2}, \cdots, p_{iD}),\ i = 1, 2, \cdots N$$

✓ **全局极值：** 整个粒子群迄今为止搜索到的最优位置，记为

$$g_i = (g_1, g_2, \cdots, g_D),\ i = 1, 2, \cdots N$$

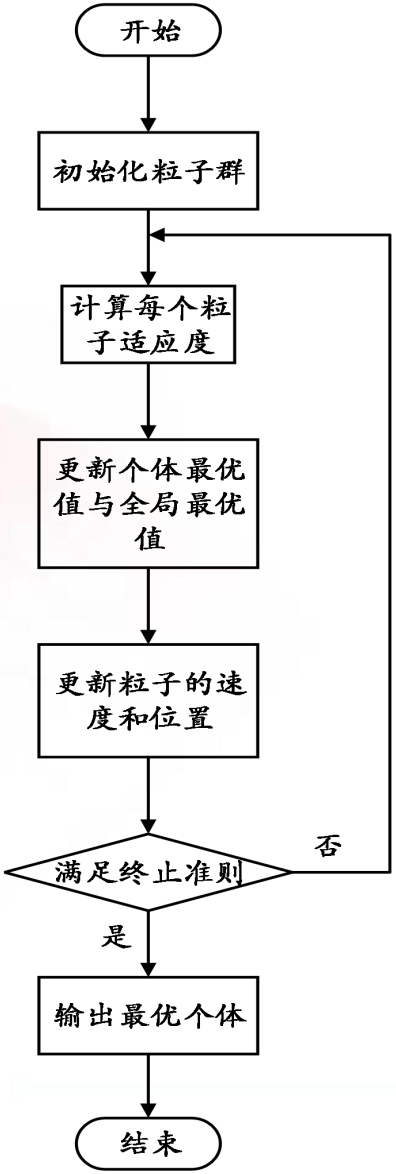
- 粒子群算法执行步骤

（1）初始化粒子群。包括群体规模N，每个粒子的位置x_i和速度v_i。

（2）个体评价。计算群体中各个粒子的适应度$fit[i]$。

（3）更新个体最优值。对每个粒子，将它的适应度值$fit[i]$与个体极值$p_{best}(i)$比较，如果$fit[i]$优于$p_{best}(i)$，则用$fit[i]$替换掉$p_{best}(i)$。

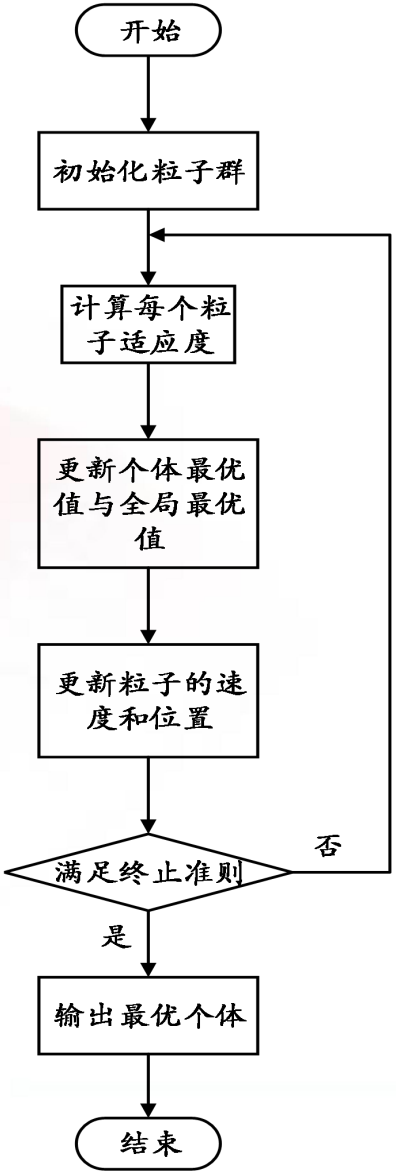
（4）更新全局最优值。对每个粒子，将它的适应度值$fit[i]$与全局极值g_{best}比较，如果$fit[i]$优于g_{best}，则用$fit[i]$替换掉g_{best}。



- 粒子群算法执行步骤

（5）更新粒子。按照迭代公式更新粒子位置x_i和速度v_i。

（6）终止准则。判断算法终止条件是否满足：若满足，则结束算法并输出优化结果；否则返回步骤（2）。



- 粒子群算法构成要素

➢ 粒子速度和位置的更新

在找到个体极值和全局极值后，粒子根据如下的迭代式来更新自己的速度和位置：

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_1(t)[p_{ij}(t) - x_{ij}(t)] + c_2 r_2(t)[p_{gj}(t) - x_{ij}(t)]$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), \quad j = 1, 2, \cdots, D$$

其中，w称为惯性权重，反映了粒子对自身速度的继承权重；c_1和c_2为学习因子，也称加速常数，反映了粒子群之间的信息交流；r_1和r_2为范围$[0, 1]$内的均匀随机数，增加了粒子飞行的随机性；$v_{ij} \in [-v_{max}, v_{max}]$，是粒子的速度，$v_{max}$是常数，由用户设定来限制粒子的速度。

- 粒子群算法构成要素

➢ 粒子速度和位置的更新

粒子的速度更新公式分为以下三部分：

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_1(t)[p_{ij}(t) - x_{ij}(t)] + c_2 r_2(t)[p_{gj}(t) - x_{ij}(t)]$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), \quad j = 1, 2, \cdots, D$$

“惯性部分”：反映了粒子对自身运动状态的信任，代表粒子有维持当前速度的趋势。

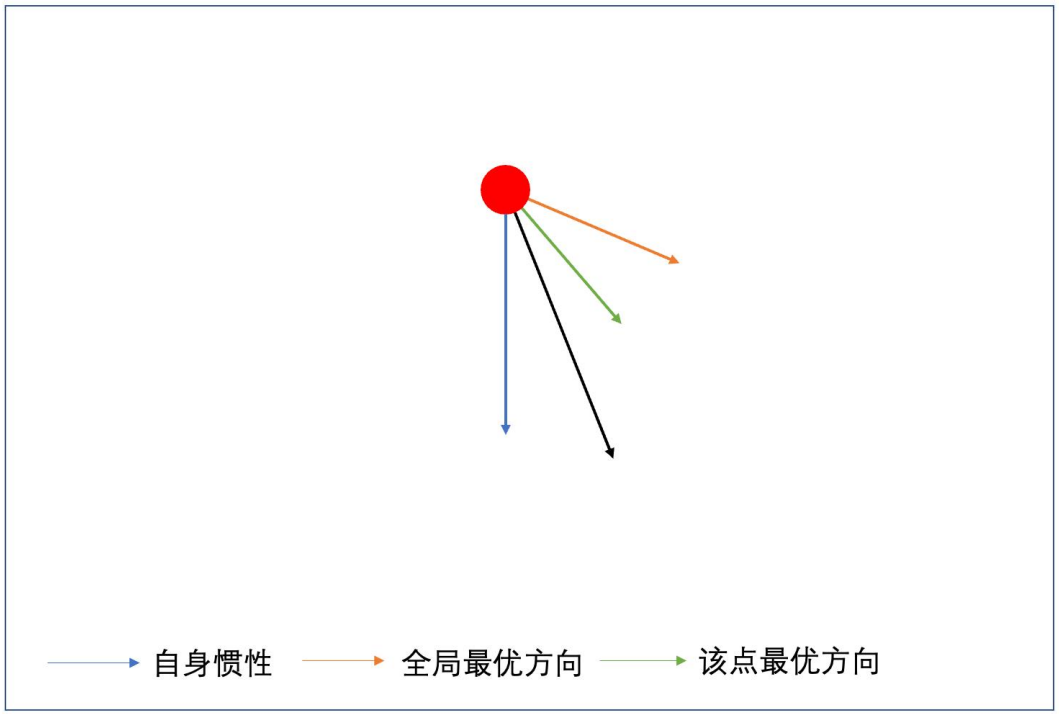
“认知部分”：反映了粒子对自身历史经验的思考，代表粒子有向自身历史最佳位置靠近的趋势。

“社会部分”：反映了粒子对群体中其他优秀粒子经验的思考，代表粒子有向群体历史最佳位置靠近的趋势。

- 粒子群算法构成要素

➢ 粒子速度和位置的更新

粒子的运动受自身惯性、全局最优方向、粒子历史最优方向三部分影响：



- 粒子群算法参数说明

➢ 粒子种群规模N

种群规模将影响粒子群优化算法的最终结果以及执行效率，当种群规模N过小时，优化性能一般不太尽如人意。采用较大的粒子种群规模可以搜索更大的空间范围，也就更容易发现全局最优解，但较大的种群规模又意味着计算复杂度较高。一般N取20~50，对于较复杂问题，可以取到100~200 。

- 粒子群算法参数说明

➢ 惯性权重w

惯性权重w是粒子群算法中非常重要的控制参数，其大小表示了粒子对当前速度继承的多少。当惯性权重较大时，全局寻优能力较强，局部寻优能力较弱；当惯性权重较小时，全局寻优能力较弱，局部寻优能力较强。惯性权值的取值通常有固定权重和时变权重两种。

①固定权值：在迭代过程中为一常数，一般取值范围为[0.8，1.2]。

②时变权值：在迭代过程中以某种方式逐步减小，也即是越接近最优值，每个粒子探索的步长越小。

- 粒子群算法参数说明

➢学习因子c_1和c_2

学习因子c_1和c_2分别调节向p_{best}和g_{best}方向探索的最大步长，它们分别决定粒子个体经验和群体经验对粒子运动轨迹的影响，反映了粒子群之间的交流。如果学习因子设置的很小，则粒子群的运动缓慢；反之亦然。它们的数值通常取$c_1 = c_2 = 1.5$，这说明个体经验和群体经验对粒子的运动具有同样的重要性，使得最后的最优解更加精确。

- 粒子群算法参数说明

➢ 粒子的最大速度v_{max}

用来对粒子的速度进行限制，使粒子速度控制在范围$[-v_{max}, v_{max}]$内，该值一般由用户设定。若该值设定过大，则粒子可能飞过优秀解区域；若设置过小，则粒子可能无法跳出局部最优解，达到更佳的位置。当粒子达到最大速度或最大位置时，可在取值范围内随机产生一个数值替代。

➢ 终止准则

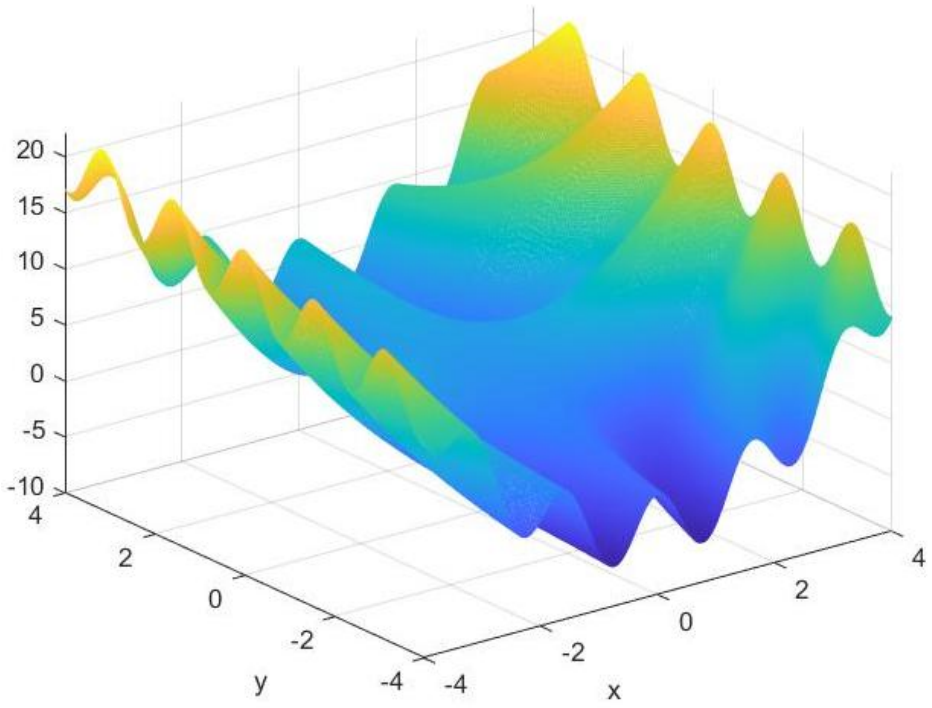
用最大迭代次数以及最小误差要求作为终止准则，不同的优化问题可能有不同的终止条件。

2.3 粒子群算法计算实例

- 求解以下无约束优化问题

$$\min \ f(x, y) = 3\cos(xy) + x + y^2$$

$$x \in [-4, 4], y \in [-4, 4]$$



观察左侧函数图像可知，该函数具有多个局部极值。

- 求解以下无约束优化问题

$$min\ \ f(x,\ y) = 3cos(xy) + x + y^2$$

$$x \in [-4,\ 4],\ y \in [-4,\ 4]$$

➢ 参数设置：

设置群体粒子数量$N = 4$，粒子维度D$= 2$；最大迭代次数T$= 200$；学习因子$c_1 = c_2 =$1.5；惯性权重最大值$w_{max} = 0.8$，最小值$w_{min} = 0.4$；粒子位置最大值$x_{max} = 4$，位置最小值$x_{min} = -4$；粒子速度最大值$v_{max} = 1$，速度最小值$v_{min} = -1$。

➢ 选取适应度函数：

在此问题中，可将目标函数直接作为适应度函数。

2.3 粒子群算法计算实例

- 求解以下无约束优化问题

$$min\ \ f(x, y) = 3cos(xy) + x + y^2$$

$$x \in [-4, 4], y \in [-4, 4]$$

➢ 初始化：

随机初始化种群粒子的位置x_i^0和速度v_i^0如下：

i	位置	速度	$f(x_i^0)$
1	$[0.139, -3.656]$	$[-0.668, -0.167]$	16.51
2	$[3.963, -0.071]$	$[-0.279, 0.815]$	6.97
3	$[1.661, -0.427]$	$[0.761, -0.811]$	4.84
4	$[-3.356, -0.106]$	$[-0.489, 0.637]$	-0.34

初始种群全局最优位置为x_4^0：$[-3.356, -0.106]$

全局最佳适应度为$f(x_4^0) = -0.34$

- 求解以下无约束优化问题

$$\min \ f(x, y) = 3\cos(xy) + x + y^2$$

$$x \in [-4, 4], y \in [-4, 4]$$

➢ 更新粒子速度和位置：

按照以下公式更新粒子的速度和位置：

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_1(t)[p_{ij}(t) - x_{ij}(t)] + c_2 r_2(t)[p_{gj}(t) - x_{ij}(t)]$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), \quad j = 1, 2, \cdots, D$$

此处的惯性权值采用时变权值，即其数值随着迭代次数i增加而逐渐减小，更新方法如下：

$$w = w_{max} - (w_{max} - w_{min}) * (i/T)$$

- 求解以下无约束优化问题

$$\min \ f(x, y) = 3\cos(xy) + x + y^2$$

$$x \in [-4, 4], y \in [-4, 4]$$

➢ 边界条件处理：

当粒子的位置和速度达到边界时，即：

$$v_{ij} > v_{max} \ 或 \ v_{ij} < v_{min}$$

$$x_{ij} > x_{max} \ 或 \ x_{ij} < x_{min}$$

按照以下公式进行处理：

$$v_{ij} = \text{rand} * (v_{max} - v_{min}) + v_{min}$$

$$x_{ij} = \text{rand} * (x_{max} - x_{min}) + x_{min}$$

其中，rand是一个范围为[0，1]的随机数

2.3 粒子群算法计算实例

- 求解以下无约束优化问题

$$\min \ f(x, y) = 3\cos(xy) + x + y^2$$

$$x \in [-4, 4], y \in [-4, 4]$$

➢ 第一代种群：

更新后的第一代种群粒子的位置x_i^1和速度v_i^1如下：

i	位置	速度	$f(x_i^1)$
1	$[0.365, -2.192]$	$[0.226, -0.499]$	8.17
2	$[3.286, 0.558]$	$[-0.677, 0.629]$	6.60
3	$[3.247, -0.737]$	$[0.793, -0.310]$	6.79
4	$[-3.746, 0.402]$	$[-0.390, 0.508]$	-0.59

初始种群全局最优位置为x_4^1： $[-3.746, 0.402]$

全局最佳适应度为$f(x_4^1) = -0.59$

- 求解以下无约束优化问题

$$\min \ f(x, y) = 3\cos(xy) + x + y^2$$

$$x \in [-4, 4], y \in [-4, 4]$$

➢ 循环操作：

又重新更新个体历史最优位置以及全局最优位置，判断是否满足终止条件，若不满足，则继续进行迭代优化；满足终止条件则输出全局最优值。

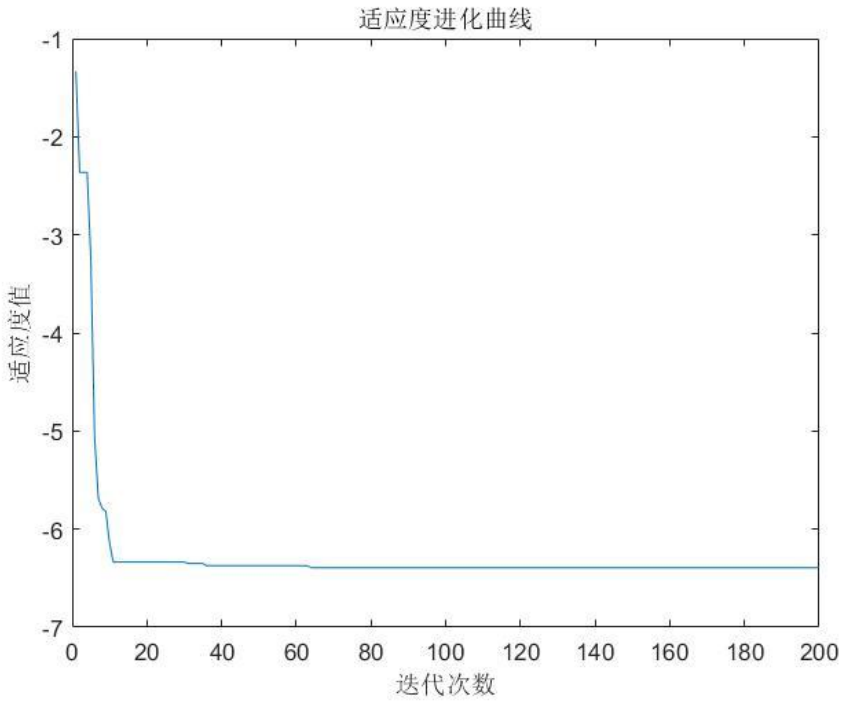
2.3 粒子群算法计算实例

- 求解以下无约束优化问题

$$\min \ f(x, y) = 3\cos(xy) + x + y^2$$

$$x \in [-4, 4], y \in [-4, 4]$$

➢ 输出最优解：



该问题迭代200次的适应度进化曲线如下，优化后的结果为：在 $x = -3.9894$, $y = -0.7528$ 时函数取得最小值-6.394。

2.4 粒子群算法的特点

- 粒子群算法是一种基于群智能理论的优化算法，通过群体中粒子间的交流与合作产生的智能优化搜索。

- 粒子群算法中的每个粒子在算法结束后仍保留其个体极值，因此粒子群算法除了可以找到问题的最优解外，还会得到若干的次优解。

- 粒子群算法特有的记忆使其可以动态地跟踪当前搜索情况并调整其搜索策略。另外，粒子群算法对种群的大小不敏感，即使种群数目下降，粒子群算法的性能下降也不会很明显。

➢ 带收缩因子的粒子群算法

Cleck等人提出利用约束因子来控制系统行为的最终收敛，该方法可以有效搜索不同的区域，并且得到高质量的解。压缩因子法的速度更新公式为：

$$v_{ij}(t+1)=\lambda\{v_{ij}(t)+c_1r_1(t)[p_{ij}(t)-x_{ij}(t)]+c_2r_2(t)[p_{gj}(t)-x_{ij}(t)]\}$$

式中，λ是压缩因子，其表达式为

$$\lambda=\frac{2}{|2-c-\sqrt{c^2-4c}|},\quad c=c_1+c_2$$

实验结果表明：与使用惯性权重的粒子群优化算法相比，使用带有约束因子的粒子群算法具有更快的收敛速度。

§3 禁忌搜索算法

- 禁忌搜索算法概念

禁忌搜索算法（Tabu Search or Taboo Search）是一种模拟人类的记忆功能，模拟人类智能的记忆机制，采用禁忌表封锁刚刚搜索的区域以避免迂回搜索，同时引入特赦(破禁)准则来释放一些被禁忌的优良状态,以保证搜索过程的有效性和多样性，从而达到全局最优。禁忌搜索算法在组合优化、神经网络和机器学习领域得到广泛应用。

- 禁忌搜索算法发展历程

1986年，Glover教授首次提出禁忌搜索算法，通过禁止某些行动的策略来超越局部最优。

此后，Glover教授在1986年和1990年发表的两篇标题为Tabu search的论文，提出了现在大家所熟知的禁忌搜索的大部分原理。

1997年，Glover与Laguna合著的第一本禁忌搜索专著正式出版，标志着禁忌搜索的相关研究日趋完善。

- 禁忌搜索算法基础—局部邻域搜索

禁忌搜索算法是基于局部邻域搜索的一种算法，在其基础上引入禁忌表，以此改进局部邻域搜索容易陷入局部最优点的不足。

➢ 局部邻域搜索

局部邻域搜索是基于贪婪准则持续地在当前的邻域中进行搜索，其算法简单，易于实现，容易理解。简单的邻域搜索算法可以描述为：

① 选定一个初始可行解；
② 在其邻域内选择更优解移动，若这样的解不存在，则将该可行解作为最优解输出，算法停止；
③ 若邻域中存在更优解，则将作为当前解，重复第2步，继续搜索。

- 禁忌搜索算法基础—局部邻域搜索

➢ 局部邻域搜索

虽然局部邻域搜索算法简单，易于实现。但从其实现过程可以看出，其搜索性能完全依赖于邻域结构和初始解，容易陷入局部最优解而无法保证全局最优解。

- 禁忌搜索算法基本思想

➢ 建立禁忌表

为了避免陷入局部最优值，禁忌搜索算法采用了一种灵活的“记忆方式”，即对已经进行的优化过程进行记录，指导下一步的搜索方向，这就是禁忌表的建立。

凡是处在禁忌表的移动，在当前迭代过程中是禁忌进行的，这样可以防止算法重复访问已访问过的解，从而防止了循环，帮助算法摆脱局部最优解。

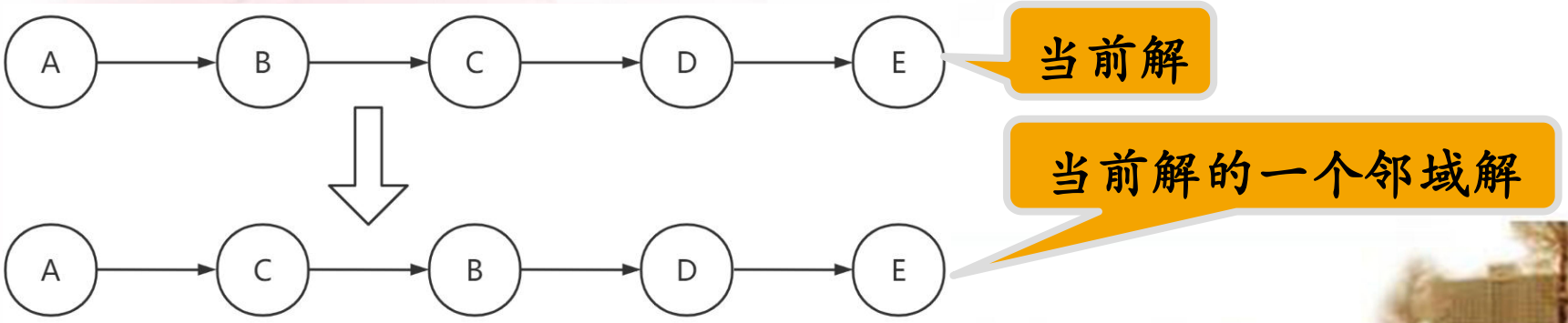
随着迭代的进行，禁忌表不断更新，最早进入禁忌表的移动就从禁忌表中解禁退出。另外，为了尽可能不错过产生最优解的移动，禁忌搜索还采用“藐视准则”的策略。

- 禁忌搜索算法相关概念

➢ 邻域结构

邻域结构是指从一个解通过“移动”到另一个解的途径，它是保证产生好的解和算法搜索速度的最重要因素之一。

通过移动，目标函数值将产生变化，移动前后的目标函数值之差，称之为移动值。如果移动值是非负的，则称此移动为改进移动;否则称作非改进移动。最常见的一种邻域搜索结构为互换操作（SWAP），即随机交换两个点的位置。



- 禁忌搜索算法相关概念

➢ 禁忌对象

所谓禁忌对象，就是被置入禁忌表中的那些变化元素。禁忌的目的是为了尽量避免迂回搜索而多搜索解空间的其他区域。归纳而言，禁忌对象通常选取状态本身或状态分量。

➢ 禁忌表

禁忌表是用来存放禁忌对象的一个容器，放入禁忌表中的禁忌对象在解禁之前不能被再次搜索，它的主要目的是阻止搜索过程中出现循环和避免陷入局部最优。禁忌表在数据结构上是一个具有一定长度的先进先出的队列。禁忌表是禁忌搜索算法的核心，禁忌表的大小在很大程度上影响着搜索速度和解的质量。如果选择的好，可有助于识别出曾搜索过的区域。

- 禁忌搜索算法执行过程示例

第一步：随机选取初始解（2 5 7 3 4 6 1），其适配度为10，禁忌表被置空。

2	5	7	3	4	6	1

初始解适配度：10

	2	3	4	5	6	7
1						
2						
3						
4						
5						
6						

禁忌表

- 禁忌搜索算法执行过程示例

第二步：对初始解进行互换操作得到一系列状态，并选取其中适配度最佳的5个状态作为候选解，据此更新当前解和禁忌表：初始解进行互换操作后得到最佳5个候选解中（5 4）互换后得到的适配度为16，因此将当前解更新为（2 4 7 3 5 6 1）。并把（5 4）加入禁忌表中，用“*”标注入选的交换对，用“T”标注被禁忌的对，同时标记其在禁忌表中的“任期”为3，表示它将被禁忌3步。

（5 ,4）	6*
（7, 4）	4
（3, 6）	2
（2,3）	0
（4,1）	-1

候选解适配度增量

2	4	7	3	5	6	1

当前解适配度：16

	2	3	4	5	6	7
1						
2						
3						
4		3				
5						
6						

禁忌表

- 禁忌搜索算法执行过程示例

第三步：对当前解再次进行互换操作，根据适配值增量更新当前解和禁忌表。当前解更新为（2 4 7 1 5 6 3），禁忌表中（4 5）对应的值减为2，（3 1）对应的值变为3。

（3,1）	2*
（2,3）	1
（3,6）	-1
（7,1）	-2
（6,1）	-4

候选解适配度增量

2	4	7	1	5	6	3

当前解适配度：18

	2	3	4	5	6	7
1		3				
2						
	3					
			4	2		
				5		
					6	

禁忌表

- 禁忌搜索算法执行过程示例

第四步：对当前解再次进行互换操作，由于当前解的5个最佳候选解均不能使适配值得到提高，而禁忌搜索却无视这一点（这也是算法实现局部解突破的关键点）。同时由于（1 3）和（4 5）是禁忌对象，所以算法在候选解集中选择非禁忌的最佳候选解为下一当前解，即（2 4）互换得到的解，其适配度降到14，并把（2 4）加入禁忌表。

（1,3）	-2T
（2,4）	-4*
（7,6）	-6
（4,5）	-7T
（5,3）	-9

候选解适配度增量

4	2	7	1	5	6	3

当前解适配度：14

	2	3	4	5	6	7
1		2				
2			3			
		3				
			4	1		
				5		
					6	

禁忌表

- 禁忌搜索算法执行过程示例

第五步：当前解的适配值为14，之前的最优解的适配值为18，对当前解再次进行互换操作，所得的5个最佳候选解中虽然（4 5）互换是禁忌对象，但由于它导致的适配值为20，优于“Best So Far”状态，因此算法根据藐视准则对它解禁，并将它重新加入禁忌表，并更新当前解。之后算法按照相同机理进行操作直至算法终止条件成立。

（4,5）	6T*
（5,3）	2
（7,1）	0
（1,3）	-3T
（2,6）	-6

候选解适配度增量

5	2	7	1	4	6	3

当前解适配度：20

	2	3	4	5	6	7
1		1				
2			2			
	3					
			4	3		
				5		
					6	

禁忌表

3.2 禁忌搜索算法基本原理

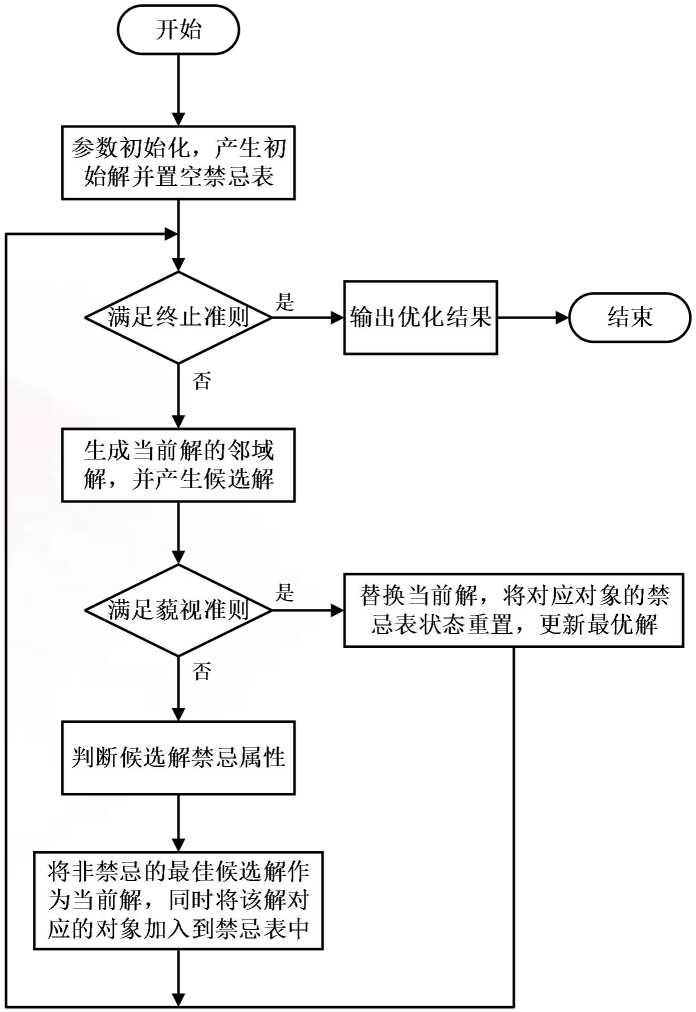
- 禁忌搜索算法执行步骤

（1）初始化。给定算法参数，随机产生初始解x，置禁忌表为空。

（2）判断算法终止条件。若满足，则结束算法并输出优化结果；否则，继续以下步骤。

（3）确定候选解。利用当前解的邻域函数产生其所有（或若干）邻域解，并从中确定若干候选解。

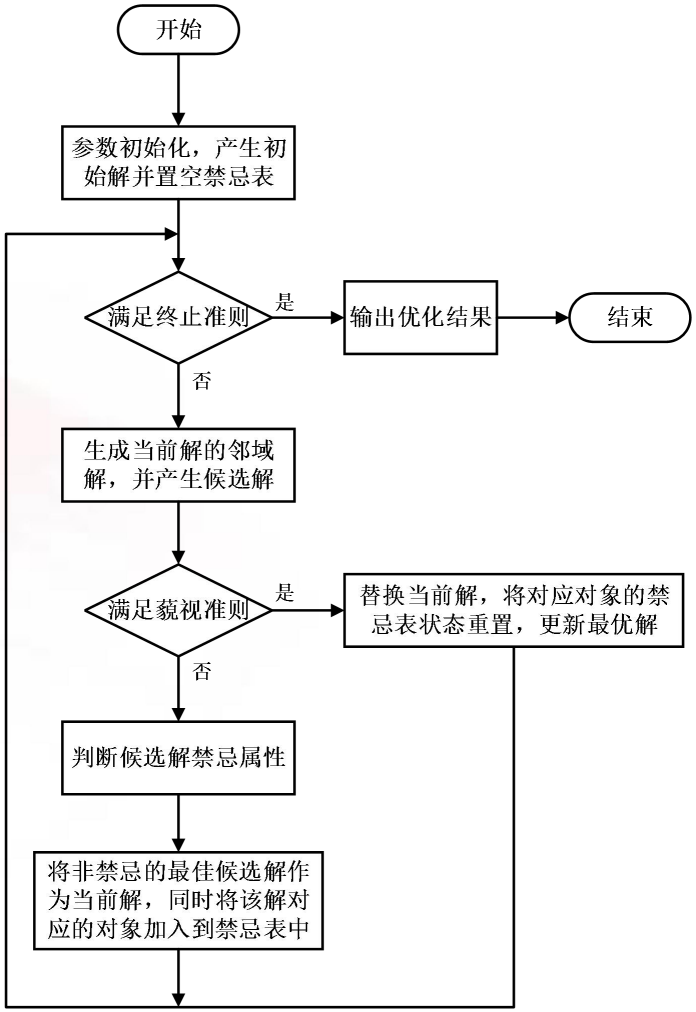
（4）判断藐视准则。对候选解判断藐视准则是否满足：若成立，则用满足藐视准则的最佳状态y替代x成为新的当前解，即x=y，并将对应对象的禁忌表状态重置，同时用y替换“best so far”（当前最优解）状态，然后转步骤6；否则，继续以下步骤。



- 禁忌搜索算法执行步骤

（5）判断候选解对应的各对象的禁忌属性，选择候选解集中非禁忌对象对应的最佳状态为新的当前解，同时将该解对应的对象加入到禁忌表中。

（6）转步骤（2）



- 禁忌搜索算法构成要素

➢候选解选择策略

一般来说局部搜索并不需要在每次迭代中评价邻域中所有解，而只是其中一个子集，这个子集就是候选集。候选集的确定对搜索速度于算法性能影响都很大，选取较大将需要大量的计算，而选取较少则容易造成“早熟收敛”。

一个好的选择策略应该是既保证解的质量又保证计算速度。当前采用最广泛的两类策略是最好解优先策略和第一个改进解优先策略。

最好改进解优先策略：对当前邻域中选择移动值最好的移动产生的解，作为下一次迭代的开始。

第一个改进解优先策略：搜索邻域移动时选择第一个改进当前解的邻域移动产生的解作为下一次迭代的开始。

- 禁忌搜索算法构成要素

➢ 藐视准则

在禁忌搜索算法中，可能出现当前邻域候选解全部被禁忌，或者存在一个优于当前最优解状态的禁忌候选解，为了不漏掉这个解，此时特赦准则则将某些状态解禁，以实现更高效的优化性能。

藐视准则的常用方法有：

（1）基于评价值的规则。若出现一个候选解的目标值好于当前最优解，可特赦。

（2）基于最小错误的规则。若所有的候选解均被禁忌，且没有解优于当前最优解，则特赦一个当前候选解中最佳的解。

（3）基于搜索方向的规则。若禁忌对象上次被禁忌使得目标值有所改善，并且该禁忌对象的目标值好于当前解，则特赦该禁忌对象。

- 禁忌搜索算法构成要素

➢ 终止准则

终止准则表示算法在何种条件下停止搜索过程，在禁忌搜索中终止规则通常有如下几种：

（1）给定最大迭代数，当禁忌搜索算法运行到指定的迭代步数后，则终止搜索；

（2）若在给定数目的迭代内所发现的最好解无法改进或无法跳出它时，算法终止；

（3）设置适配值的偏离幅度。首先估计问题的下界，当算法最优解的适配值与下界的偏离值小于某规定阈值时，终止搜索。

- 禁忌搜索算法参数说明

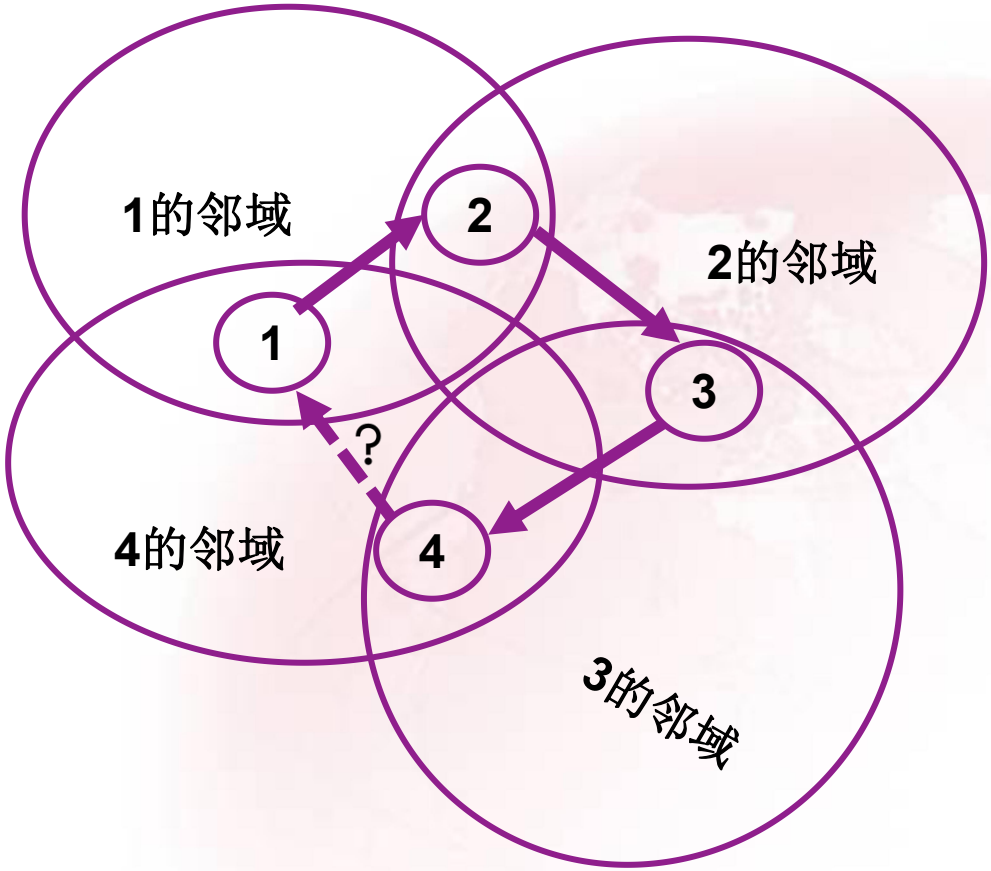
➢初始解

由于禁忌搜索主要基于邻域搜索，算法对初始解有较强的依赖性，好的初始解可使禁忌搜索在解空间搜索到好的解，而较差的初始解则会降低禁忌搜索的收敛速度。

初始解的构造可以随机产生，但效果往往不够理想，常用方法是基于问题的特征信息，借助启发式方法（如遗传算法和粒子群算法等）先进行搜索，确定禁忌搜索的初始解，然后再用禁忌搜索算法进行搜索。

- 禁忌搜索算法参数说明

➢ 禁忌长度



设禁忌表长度为2

规则：不得接受与禁忌表中相同的解

禁忌表的变化：

第一步搜索时{　　}

第二步搜索时{①　}

第三步搜索时{①, ②}

第四步搜索时{②, ③}

在第四步搜索时，①已经从禁忌表中释放，因此可能会出现重复搜索①的情况。可见若禁忌长度过小可能会导致搜索的循环。