第四章 几种典型优化算法的对比

4.1 引言

各种典型的智能优化算法常用于求解使某问题 $f(\mathbf{x})$ 取最小值时的参数 $\mathbf{x}$：

$$\text{Min}\, f(\mathbf{x}), \mathbf{x}=[x_1, x_2, ..., x_d, ..., x_D]^{\mathrm{T}}, x_d \in [R_{\mathrm{l},d}, R_{\mathrm{u},d}] \tag{1.1}$$

式中，$f(\mathbf{x})$ 为目标函数，一般为某种形式的数学函数或误差函数表达式；$\mathbf{x}$ 为待优化参数向量，也就是待求解问题的解，如果 $\mathbf{x}$ 能使 $f(\mathbf{x})$ 最小，则 $\mathbf{x}$ 被称为最优参数；x_i 为待优化参数向量 $\mathbf{x}$ 的某个维度的元素；D 是解向量的维度；$R_{\mathrm{l},d}$ 和 $R_{\mathrm{u},d}$ 分别表示第 d 维参数的下限和上限；T 表示对解行向量进行转置操作变为列向量；Min 代表欲通过使 $f(\mathbf{x})$ 最小化得到待优化参数 $\mathbf{x}$。如果 $f(\mathbf{x})$ 是某种形式的数学函数，那么可以通过智能优化算法直接最小化函数 $f(\mathbf{x})$ 得到待优化参数 $\mathbf{x}$。如果要利用智能优化算法根据曲线拟合的方式反演本构模型参数，$f(\mathbf{x})$ 可以设置为本构模型预测值与室内试验测量值之间的某种误差函数，只要使误差函数最小就可以得到本构模型参数 $\mathbf{x}$。从这儿可以看出，最小化某数学函数时目标函数是 $f(\mathbf{x})$ 的函数值本身，而曲线拟合时目标函数 $f(\mathbf{x})$ 是某种形式的误差函数，误差函数的类型通常会影响待优化参数 $\mathbf{x}$ 的结果。

智能优化算法是建立在生物智能或自然现象基础上的一种随机搜索算法，其主要思想是模拟自然界一些群居物种觅食、竞争和繁殖等行为，将以上各种行为抽象为可量化的关键指标，形成数学模型用于求解各类问题[1]。求解的各类问题统称优化问题，优化问题是指在满足一定条件下，在众多参数方案中寻找最优参数方案，以使得某个或多个指标达到最优，或使系统的某些性能指标达到最大值或最小值。优化问题广泛地存在于信号处理、图像处理、生产调度、任务分配、模式识别、自动控制和机械设计等众多领域[2]。优化问题中很多问题的目标函数表达式未知，也无法给出导数信息，只能根据离散的自变量取值得到对应的目标函数值，属于黑盒问题。智能优化算法在优化时有自己的学习模式，不需要导数等信息，十分适合解决这类黑盒问题的优化。土的本构模型参数反演问题也属于

黑盒问题，也适合用智能优化算法解决。

如图 1 所示，智能优化算法大致可分为单一解优化算法和基于种群的元启发式优化算法[3]。需要注意的是，单一解元启发式优化算法并不是只能优化一个参数，而是优化时仅利用单个候选解向量，并利用全局和局部搜索对该解向量进行改进。著名的基于单一解的元启发式算法有模拟退火（Simulated Anneaing，SA）[4]、状态转移算法（State Transition Algorithm，STA）[5]、禁忌搜索(Tabu Search or Taboo Search，TS)[6]、天牛须算法（Beetle Antennae search algorithm, BAS）[7]、微正则退火(Microcanonical Annealing，MA)[8]、引导局部搜索(Guided Local Search，GLS)[9]等。大量的研究表明基于个体的局部搜索方法往往能够得到更好的性能和收敛速度，但缺点是这类算法对初始解和邻域的依赖较大。因此，选择合适的邻域和设计有效的局部最优解逃离策略对算法性能至关重要[10]。

基于种群的优化算法又可以分为进化算法和群智能优化算法。在遗传、选择和变异等一系列作用下，自然界中的生物适者生存，从低级到高级不断地进化和发展。人们根据“适者生存”这一进化规律的本质建模，从而形成一种优化算法，即进化优化算法。常见的进化类算法有遗传算法（Genetic Algorithm，GA）[11]、差分进化算法（Differential Evolution，DE）[12]、免疫算法（Immune Algorithm，IA）[13]等。基于种群的元启发式方法在搜索过程中利用了多个候选解向量，多解向量的元启发模式保持了种群的多样性，有很大可能避免解向量陷入局部最优。群智能是指“非智能单体通过合作表现出智能行为的特征”，是一种基于生物群体行为规则的计算技术。它的灵感来自群居昆虫(如蚂蚁和蜜蜂)和群居脊椎动物(如鸟群、鱼群和兽群)，用来解决分布式问题。最著名的群智能优化算法是蚁群优化算法(Ant Colony Optimization，ACO)[14]和粒子群优化算法(Particle Swarm optimization，PSO)[15]。而后其他学者相继提出果蝇优化算法（Fruit fly Optimization algorithm，FOA）[16]、蝙蝠算法（Bat Algorithm，BA）[17]、鲸鱼优化算法（Whale Optimization Algorithm，WOA）[18]、樽海鞘群体算法(Salp Swarm Algorithm，SSA)[18]、哈里斯鹰优化算法(Harris Hawk Optimization，HHO)[19]等。群智能算法无需集中控制和全局模型，为解决复杂的分布式问题提供了一种新的方法[20]。

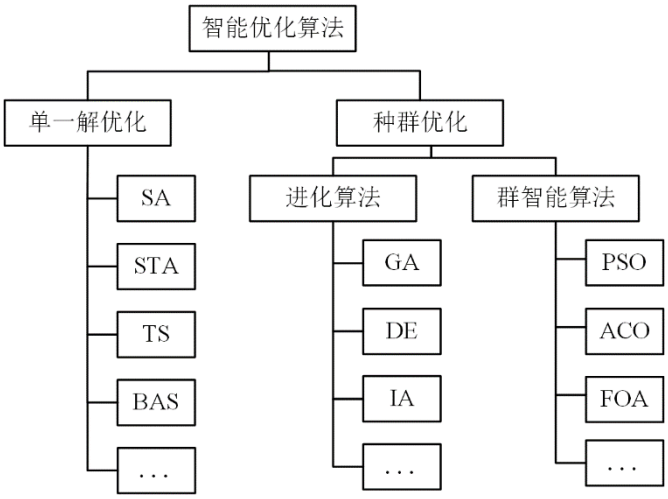


图 1　智能优化算法的分类

本章首先介绍几种典型的智能优化算法，包括模拟退火（SA）算法、状态转移（STA）算法、实数编码遗传（RCGA）算法、差分进化（DE）算法、免疫（IA）算法、粒子群（PSO）算法、蚁群（ACO）算法和多种自适应策略粒子群（MAPSO）算法。然后通过与误差函类型无关的 CEC2017 数学测试集测试以上智能优化算法，给出一套评分体系评价以上各智能优化算法的优化能力。之后介绍利用优化算法通过曲线拟合的方法反演土的本构模型参数的过程，并分析不同误差函数地选择对本构模型参数的影响。结合 CEC2017 测试结果和土的本构模型参数优化确定结果，选出优化能力较强的优化算法。

4.1 典型智能优化算法简介

4.1.1 模拟退火算法（SA）算法

1953 年，Metropolis 等最早提出模拟退火（Simulated Anneaing，SA）算法的思想。1983 年，Kirkpatrick 等首次使用模拟退火算法求解组合优化问题[4]。模拟退火算法是一种基于蒙特卡罗（Monte Carlo）迭代求解策略的随机优化算法。高温下金属内能最高，熔化的金属内部原子可以自由移动，随着温度逐渐降低金属会逐渐冷却为固体，其内的原子也会丧失流动性，但在丧失流动性前会进行有序排列，使金属达到最低能态。表 1 展示了金属物理退火与模拟退火算法中一些对应的概念。

表 1　SA 与金属退火过程的相似关系

金属物理退火	模拟退火
粒子状态	解 **x**

能量最低状态	最优解 $\mathbf{x}_{best}$
熔解过程	设定初温
等温过程	Metropolis 采样过程
冷却	控制参数的下降
能量 $E(\mathbf{x})$	目标函数 $f(\mathbf{x})$

模拟退火算法是一种单一解随机搜索算法，即全局搜索和局部搜索都围绕一个解展开，算法的主要内容是产生新解和判断是否接收新解：

（1）根据当前解产生一个新解

通常选用根据当前解进行简单变换就能产生新解的方法，比如直接在当前解附近产生新解：

$$x_{g+1,d} = x_{g,d} + step \cdot rand \cdot \left(R_{u,d} - R_{l,d}\right) + R_{l,d}, d = 1,2,...,D \qquad (1.2)$$

式中，$x_{g,d}$ 为第 g 代当前解向量 $\mathbf{x}_g$ 的第 d 个元素；*step* 为步长大小，可以取 0.01；*rand* 函数可以取 0 到 1 之间的均匀随机数；$R_{u,d}$ 和 $R_{l,d}$ 分别为 $x_{g,d}$ 的上限和下限；D 表示解向量 $\mathbf{x}_g$ 的维度，也就是待优化参数的个数。新解产生后可以分别计算当前解 $\mathbf{x}_g$ 和新解 $\mathbf{x}_{g+1}$ 的目标函数，分别记为 $f\left(\mathbf{x}_g\right)$ 和 $f\left(\mathbf{x}_{g+1}\right)$，当然，也可以记为 $E\left(\mathbf{x}_g\right)$ 和 $E\left(\mathbf{x}_{g+1}\right)$。

（2）是否接收新解的判别准则

上述产生的新解 $\mathbf{x}_{g+1}$ 可能比当前解 $\mathbf{x}_g$ 好，即 $f\left(\mathbf{x}_{g+1}\right) < f\left(\mathbf{x}_g\right)$，也可能比当前解差，即 $f\left(\mathbf{x}_{g+1}\right) > f\left(\mathbf{x}_g\right)$，这就需要一个判别准则来判断是否接收新产生的解。最常用的判别准则是 Metropolis 准则：若 $f\left(\mathbf{x}_{g+1}\right) < f\left(\mathbf{x}_g\right)$ 则接受新解 $\mathbf{x}_{g+1}$ 作为新的当前解，此处实现了局部搜索；否则，以概率 p 接受 $\mathbf{x}_{g+1}$ 作为新的当前解。接受新解 $\mathbf{x}_{g+1}$ 作为新的当前解 $\mathbf{x}_g$ 的概率 p：

$$p = \begin{cases} 1 & f\left(\mathbf{x}_{g+1}\right) < f\left(\mathbf{x}_g\right) \\ \exp\left(-\dfrac{f\left(\mathbf{x}_{g+1}\right) - f\left(\mathbf{x}_g\right)}{T}\right) & f\left(\mathbf{x}_{g+1}\right) \geq f\left(\mathbf{x}_g\right) \end{cases} \qquad (1.3)$$

式(1.3)中 T 表示当前温度，概率 p 随着温度 T 的减小而减小，如果概率 p 大于某

个 0 到 1 之间的随机数 *rand* 则接受新解 $\mathbf{x}_{g+1}$ 作作为新的当前解 $\mathbf{x}_g$ 。

$f\left(\mathbf{x}_{g+1}\right) \geq f\left(\mathbf{x}_g\right)$ 时 $p>0$，即新解 $\mathbf{x}_{g+1}$ 比当前解 $\mathbf{x}_g$ 差时仍有一定的概率接受较差的解 $\mathbf{x}_{g+1}$ 作为当前解，这就是模拟退火算法可以进行全局搜索的原因。

第 g 代的温度 $T(g)$一般随着代数 g 的增大而缩小，缩小速率 K 一般给一个接近但小于 1 的数，比如令缩小速率 K=0.998。温度 T 随着代数的增加而降低的过程就是模拟的金属的退火过程。开始优化时温度 T 较高，根据式(1.3)可知可以接受较差的新解作为当前解。随着温度 T 的降低，只能接受稍差的新解作为当前解。当温度 T 降低为 0 时就不再接受任何较差的新解作为当前解了。

由于模拟退火算法具有全局优化能力，所以从理论上说求得的全局最优解结果与初始解无关。模拟退火算法的流程如图 2 所示，现叙述如下：

（1）初始化：设置较大的初始温度 $T(g)$、终止时最低温度 T_{end} 和两代间容许误差 f_{tol}，随机给定初始解向量 $\mathbf{x}_g$ ，每个温度下迭代的次数 L，此时 g=0；

（2）利用式(1.2)根据当前解 $\mathbf{x}_g$ 产生新解 $\mathbf{x}_{g+1}$；

（3）计算当前解和新解的目标函数，根据式(1.3)判断是否接收新解作为新的当前解。

（4）在此温度下是否达到 L 次搜索，如果未达到转第（2）步。

（5）当前温度 $T(g)$是否低于设定的终止温度 T_{end}，前后两代最优解的目标函数下降差值 $\Delta f = f\left(\mathbf{x}_{g+1}\right) - f\left(\mathbf{x}_g\right)$ 是否小于 f_{tol}，如果都满足则输出当前解作为全局最优解，结束程序。否则，转第（2）步继续计算。

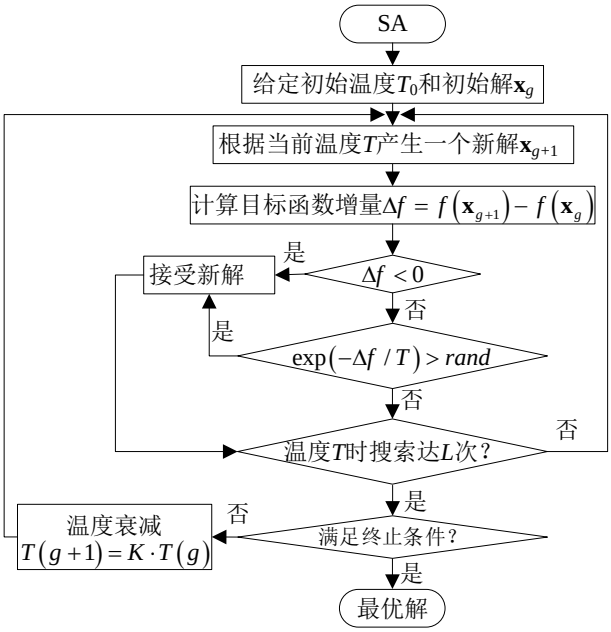


图 2　模拟退火（SA）算法的流程

4.1.2 状态转移算法（STA）

状态转移算法（STA）[5]也是单一解优化算法，即全局搜索和局部搜索都围绕一个解展开。状态转移算法将问题的解看作是一个状态，那么问题的解更新过程就是状态的转移过程。状态转移算法的优越性在于其设计的具有一定几何意义的搜索算子：旋转变换、平移变换、伸缩变换、坐标变换。其中旋转变换算子和平移变换算子旨在提高算法的局部精细化搜索能力，而伸缩变换算子和坐标变换算子旨在保证算法的全局搜索能力。下面介绍状态转移算法的四个算子：

（1）旋转变换（Rotation transformation，RT）

$$\mathbf{x}_{g+1}=\mathbf{x}_g+\alpha\frac{1}{D\left\|\mathbf{x}_g\right\|_2}\mathbf{R}_r\mathbf{x}_g \tag{1.4}$$

式中，$\mathbf{x}_g\in\mathbb{R}^D$为输入的当前解向量，其中 D 表示 $\mathbf{x}_g$ 的维度，也表示待优化参数的个数；$\mathbf{x}_{g+1}\in\mathbb{R}^D$，为根据当前解 $\mathbf{x}_g$ 旋转后的输出解向量；旋转因子 α 决定搜索的最大范围，一般取一个不大于 1 的正值常数，α 越小搜索越精细；$\mathbf{R}_r\in\mathbb{R}^D$是各元素均匀分布在$(-1,1)$之间的随机行向量；$\|\bullet\|_2$表示向量二范数。

当 D=2、α=1、$\mathbf{x}_g=(1.2,0.25)$时，根据式(1.4)搜索的 3000 个$\mathbf{x}_{g+1}$落成了如图 3 所示的区域，可见式(1.4)所示的 RT 算子在矩形区域内进行均匀搜索。

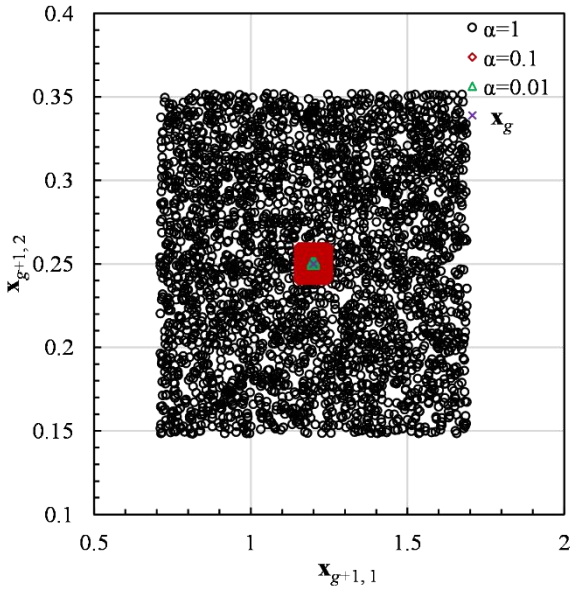


图 3　旋转因子α取不同值时旋转变换（RT）算子在二维空间中搜索效果[21]

（2）平移变换（Translation transformation，TT）

$$\mathbf{x}_{g+1}=\mathbf{x}_g+\beta R_t\frac{\left(\mathbf{x}_g-\mathbf{x}_{g-1}\right)}{\left\|\mathbf{x}_g-\mathbf{x}_{g-1}\right\|_2} \tag{1.5}$$

式中，$\mathbf{x}_{g-1}$是输入的解向量；$\mathbf{x}_g$ 是输入的比$\mathbf{x}_{g-1}$更好的解向量；$\mathbf{x}_{g+1}$是根据$\mathbf{x}_{g-1}$和$\mathbf{x}_g$ 平移的解，为输出解向量；β为转移因子，是一个正数，一般取 1；R_t 是一个均匀分布在$(0,1)$之间的随机数。式(1.5)中$\frac{\left(\mathbf{x}_g-\mathbf{x}_{g-1}\right)}{\left\|\mathbf{x}_g-\mathbf{x}_{g-1}\right\|_2}$为平移的方向；$\beta$为最大平移距离；$R_t$ 是平移距离的缩放系数。

平移算子具有沿着过点$\mathbf{x}_{g-1}$和点$\mathbf{x}_g$ 的直线方向进行搜索的功能，搜索起点为点$\mathbf{x}_g$，搜索范围最大为β。图 4 所示为二维空间中通过式(1.5)的 TT 算子根据点$\mathbf{x}_{g-1}$和点$\mathbf{x}_g$ 搜索 30 次得到的 30 个$\mathbf{x}_{g+1}$分布，其中，平移搜索的方向直线通过点$\mathbf{x}_{g-1}=(1.2,0.25)$和$\mathbf{x}_g=(1.5,0.3)$，搜索大小$\beta$=1。

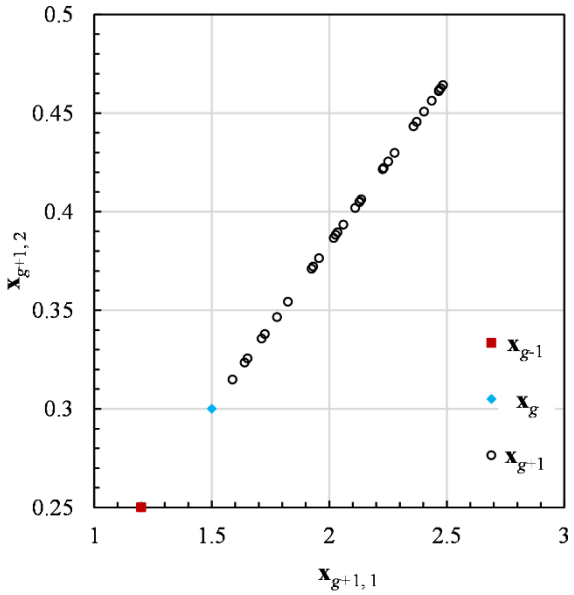


图 4　最大平移距离β=1 时平移变换（TT）算子在二维空间中的搜索效果

（3）伸缩变换（Expansion transformation，ET）

$$\mathbf{x}_{g+1}=\mathbf{x}_{g}+\lambda\mathbf{R}_{\mathrm{e}}\mathbf{x}_{g} \tag{1.6}$$

式中，λ 为伸缩因子，是一个正的常数；$\mathbf{R}_{\mathrm{e}}\in\mathbb{R}^{D\times D}$ 是一个随机对角矩阵，它里面每一个非零元素服从均值为 0、方差为 1 的高斯分布。

当 n=2、$\mathbf{x}_{g}=(1.2,0.25)$时，根据式(1.6)搜索的 3000 个 $\mathbf{x}_{g+1}$ 落成了如图 5 所示的区域，可见式(1.6)所示的 ET 算子可以产生一个以当前最好解 $\mathbf{x}_{g}$ 为中心的类似椭球形状邻域，新候选解 $\mathbf{x}_{g+1}$ 在每一个维度上中间值多，边缘值少，复合高斯分布。此变换在概率意义上具有使得 $\mathbf{x}_{g}$ 中每个元素伸缩变换到正负无穷区间的全局搜索功能。

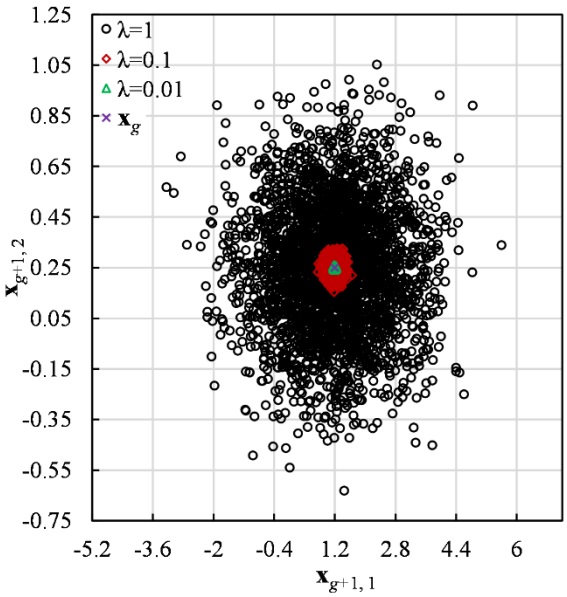


图 5　伸缩因子 λ 取不同值时伸缩变换（ET）算子在二维空间中的搜索效果

（4）坐标变换（Axesion transformation，AT）

$$\mathbf{x}_{g+1}=\mathbf{x}_g+\delta\mathbf{R}_a\mathbf{x}_g \tag{1.7}$$

式中，δ 是坐标变换因子，是一个正的常数；$\mathbf{R}_a\in\mathbb{R}^{D\times D}$ 是一个随机对角稀疏矩阵，它只在某个随机位置有非零元素，其非零元素服从高斯分布。

当 n=2、$\mathbf{x}_g=(1.2,0.25)$时，根据式(1.7)搜索的 3000 个 $\mathbf{x}_{g+1}$ 落成了如图 6 所示的区域，可见式(1.7)所示的 ET 算子可保持其余维度待优化参数不变，仅在某一维度上进行搜索。因此坐标变换算子可以产生一个以当前最好解 $\mathbf{x}_g$ 为中心的坐标轴方向邻域，新候选解每次只在一个维度上有改变，理论上能随正态概率分布于正负无穷之间，因此拥有坐标轴上的单维全局搜索能力。

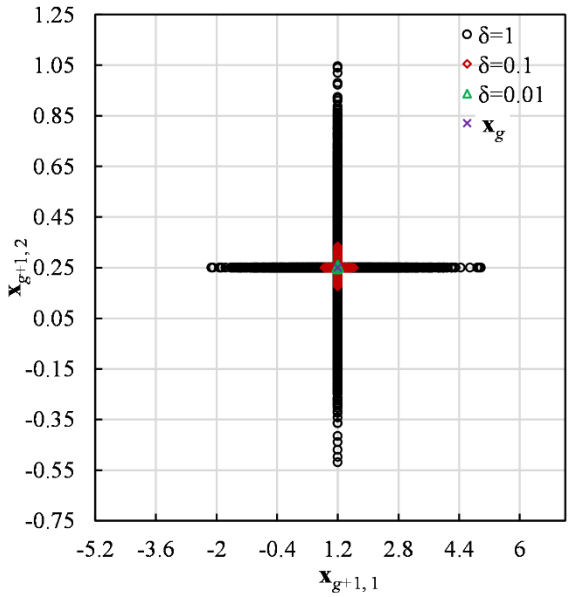


图 6　坐标变换因子 δ 取不同值时坐标变换（AT）算子在二维空间中的搜索效果

至此，四个搜索算子介绍完毕。状态转移算法从当前最好的解出发，利用四个算子产生邻域并从邻域挑选更好的解作为当前解，四个算子配合情况如图 7 所示。状态转移算法的具体计算步骤如下：

（1）设置算法参数，比如 $\Omega=\left(1,0.1,...,1\times10^{-8}\right)^T$，采样个数 SE=30，令 g=0，并给定初始解向量 $\mathbf{x}_g$。

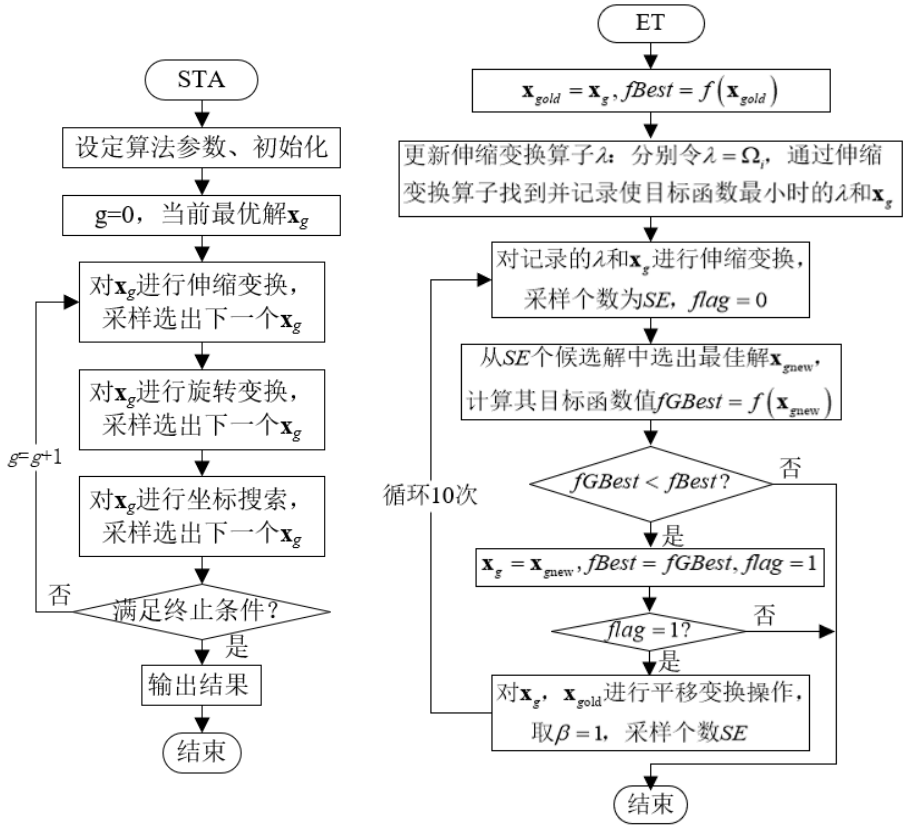
（2）伸缩变换操作。包括找到最合适的伸缩变换因子 λ 和配合平移操作的伸缩变换两部分。令 λ 遍历 Ω 的元素各进行一次伸缩变换操作，找到并记录能使目标函数最小的那次的 λ 和解向量 $\mathbf{x}_g$。然后利用这个最合适的伸缩变换因子 λ 并以解向量 $\mathbf{x}_g$ 为初始解进行伸缩变换，如果能找到更好的解，记为 $\mathbf{x}_{g+1}$，并沿 $\mathbf{x}_g$ 和 $\mathbf{x}_{g+1}$ 方向进行平移变换操作以精细化搜索。伸缩变换和平移变换配合过程操作 10 次，如图 7（b）所示，以期使找到的最合适的伸缩变换因子 λ 得到充分使用。

（3）旋转变换操作。包括找到最合适的旋转因子 α 和配合平移操作的旋转变换两部分。令 α 遍历 Ω 的元素各进行一次旋转变换操作，找到并记录能使目标函数最小的那次的 α 和解向量 $\mathbf{x}_g$。然后利用这个最合适的 α 并以解向量 $\mathbf{x}_g$ 为初始解进行旋转变换，如果能找到更好的解，记为 $\mathbf{x}_{g+1}$，并沿 $\mathbf{x}_g$ 和 $\mathbf{x}_{g+1}$ 方向进行平移变换操作以精细化搜索。旋转变换和平移变换配合过程操作 10 次。整个过程类似

于伸缩变换操作，以期使找到的最合适的旋转因子α得到充分使用。

（4）坐标变换操作。包括找到最合适的坐标变换δ和配合平移操作的坐标变换两部分。令δ遍历Ω的元素各进行一次旋转变换操作，找到并记录能使目标函数最小的那次的δ和解向量$\mathbf{x}_g$。然后利用这个最合适的δ并以解向量$\mathbf{x}_g$为初始解进行坐标变换，如果能找到更好的解，记为$\mathbf{x}_{g+1}$，并沿$\mathbf{x}_g$和$\mathbf{x}_{g+1}$方向进行平移变换操作以精细化搜索。上述坐标变换和平移变换配合过程操作 10 次。整个过程类似于伸缩变换操作，以期使找到的最合适的坐标变换因子δ得到充分使用。

（5）伸缩、旋转、坐标变换操作完成后进行下一轮操作。



（a）STA 整体流程图　　（b）伸缩变换算子

图 7　状态转移（STA）算法流程图

平移变换算子嵌入了伸缩、旋转、坐标变换中，实现进一步的细化搜索。伸缩变换算子与平移变换算子的配合过程如如图 7（b）所示，旋转和坐标变换算子与平移变换算子的配合与伸缩变换算子与平移变换算子的配合相似。STA 算法的一大特色就在于伸缩、旋转、坐标变换操作中尝试不同的搜索半径，找到合适的

搜索半径后就通过平移变换操作精细化搜索。

4.1.3 实数编码的遗传算法（RCGA）

越优秀的个体越适应环境，也越容易繁衍后代。Holland 教授等基于达尔文的这种自然选择理论于上世纪六十年代提出遗传算法，1975 年提出标准遗传算法[11]，它的优化变量由二进制编码来描述，多个优化变量的二进制编码串接在一起组成染色体，这种编码既适用于变异操作，又适用于交叉操作。二进制编码适用于低精度中小型规模问题，十进制编码适用于较高精度和较大型规模问题[22]。在此介绍一种十进制实数编码遗传算法（RCGA），在该算法中待优化参数向量可以认为是个体，多个个体组成群体，群体内个体间通过选择算子、交叉算子和变异算子不断演化，最后生存下来的都是适应度高的。对于本构模型参数优化反演问题，适应度高在此只目标函数值小。下面分别介绍遗传算法的三大算子：

（1）选择算子。根据个体的适应度，按照一定的规则从第 g 代群体 P(g)中选出适应度高的个体组成新的群体以进行后续的交叉操作。常见的选择方法有二元锦标赛选择算子和轮盘赌选择算子。

二元锦标赛选择算子具体操作过程为从群体 P(g)中随机抽取两个个体，比较两个个体的目标函数值，将目标函数值小的复制到群体 P(g+1)中，直到 P(g+1)内个体数与 P(g)中个体数相等。从上述过程可以发现，某些目标函数值较小的个体可能不止一次的被选出放入群体 P(g+1)内，所以选择算子虽然不能找到目标函数更小的解，但能提高群体 P(g+1)内解的平均水平质量。

轮盘赌选择算子是最早提出的选择方法，由 Holland 提出的一种基于比例的选择方法，类似于往轮盘上甩飞镖，轮盘上角度越大的扇形区域得到飞镖的概率越大。由于轮盘赌选择算子简单实用，因而被广泛实用，在此也使用轮盘赌选择算子。计算每个个体 $\mathbf{x}_i$ 的目标函数值 $f\left(\mathbf{x}_i\right)$，并计算总和归一化的个体选中概率 p_i：

$$p_i = \exp\left(-\frac{\beta \cdot f\left(\mathbf{x}_i\right)}{f\left(\mathbf{x}\right)_{\max}}\right) \Bigg/ \left(\sum_{i=1}^{N_\mathrm{p}} f\left(\mathbf{x}_i\right)\right) \tag{1.8}$$

式中，β 为选择压力，可取 8；$f\left(\mathbf{x}\right)_{\max}$ 表示本代群体中所有个体最大的目标函数值；N_p 表示群体中个体的个数。

概率 p_i 随着目标函数值 $f(\mathbf{x}_i)$ 的增大而减小，将每个个体对应的概率 p_i 相加，在选择过程中随机给出 0 到 1 之间的随机数 *rand*，*rand* 落在哪个个体的概率区间则选中哪个个体。轮盘赌选择算子的效果也是目标函数越大的越不容易被选中。

（2）交叉算子是将群体 P(g)内以概率 p_{m} 随机抽取的个体通过一定的规则交换或改变个体中参数得到新个体的算子。在此使用以下形式的算术交叉算子[23]：

$$\left.\begin{aligned}\mathbf{x}_{i,g+1}&=\boldsymbol{\alpha}\cdot\mathbf{x}_{i,g}+(1-\boldsymbol{\alpha})\cdot\mathbf{x}_{i+1,g}\\ \mathbf{x}_{i+1,g+1}&=\boldsymbol{\alpha}\cdot\mathbf{x}_{i+1,g}+(1-\boldsymbol{\alpha})\cdot\mathbf{x}_{i,g}\end{aligned}\right\} \tag{1.9}$$

式中 $\boldsymbol{\alpha}\in\mathbb{R}^D$ 表示[-b,1+b]之间的随机数，b 在此取 0.4。此种形式的交叉被 Muhlenbein 和 Schlierkamp-Voosen 称为扩展中间交叉。交叉算子能产生新解，但产生的新解要检查边界，确保解向量所有元素都在参数上下限范围内。

（3）变异算子。对群体中的每个个体以概率 p_{m} 将某一个或某些参数改变为上下限范围内其值。

在此令 p_{m}=0.3，即参与变异的个体有 $N_{\mathrm{p}}\times p_{\mathrm{m}}$ 个。每个个体中又有 D 个参数，参与变异的参数为 $ceil(0.1D)$ 个，其中 *ceil* 函数表示向上取整。变异的位置随机给定，变异公式如下：

$$\mathbf{x}_{i,g+1}=\mathbf{x}_{i,g}+0.1\times rand\cdot\left(\mathbf{R}_{\mathrm{u},ceil(0.1D)}-\mathbf{R}_{\mathrm{l},ceil(0.1D)}\right) \tag{1.10}$$

式中 $\mathbf{x}_{i,g}$ 为待变异的解向量；rand 为 0 到 1 之间的均匀随机数；$\mathbf{R}_{u,d}$ 和 $\mathbf{R}_{\mathrm{l},d}$ 分别表示第 $ceil(0.1D)$ 维参数的上、下限。

遗传算法的主要步骤为：

（1）设定遗传算法参数，比如交叉概率 p_{c}、变异概率 p_{m}、群体中个体数目 N_{p}，最大迭代代数 G 等。

（2）初始化个体并计算所有个体的目标函数值。

（3）根据交叉概率和群体中总个体数计算出交叉个体数，通过轮盘赌选择算子选择出交叉计算需要的个体。

（4）对选择算子选择出的个体进行交叉运算，之后计算由交叉算子新产生的个体的目标函数值。

（5）根据变异概率和群体中总个体数计算出变异个体数，从群体中随机选取

出需要变异的个体。

（6）对第（5）步选出的个体进行变异操作，得到新的个体，并计算新个体的目标函数值。

（7）将交叉和变异产生的所有新个体以及交叉和变异的源群体混合成一个群体并按目标函数值升序排列，只保留前 N_p 个个体。

（8）判断是否满足终止条件，如果不满足，则转到第（3）步，如果满足则输出最优解并终止程序。

4.1.4 差分进化算法（DE）

差分进化（Differential Evolution，DE）算法是由 Storn 等于 1995 年提出的，其最初的设想是用于解决切比雪夫多项式问题，后来发现它也是解决复杂优化问题的有效技术。同其 IA 算法、遗传算方等他进化算法一样，差分进化算法也是对候选解的种群进行操作，但其种群繁殖方案与其他进化算法不同：它通过把种群中两个成员之间的加权差向量加到第三个成员上来产生新的参数向量，该操作称为“变异”；然后将变异向量的参数与另外预先确定的目标向量参数按一定规则混合来产生试验向量，该操作称为“交叉”；最后，若试验向量的代价函数比目标向量的代价函数低，试验向量就在下一代中代替目标向量，该操作称为“选择”。种群中所有成员必须当作目标向量进行一次这样的操作，以便在下一代中出现相同个数竞争者。在进化过程中每一代的最佳参数向量都进行评价，以记录最小化过程。这样利用随机偏差扰动产生新个体的方式，可以获得一个收敛性非常好的结果，引导搜索过程向全局最优解逼近。

差分进化算法是一种随机的启发式搜索算法，简单易用，有较强的鲁棒性和全局寻优能力。它从数学角度看是一种随机搜索算法，从工程角度看是一种自适应的迭代寻优过程。除了具有较好的收敛性外，差分进化算法非常易于理解与执行，只包含初始化、变异、交叉、选择和边界条件处理 5 个算子。

（1）初始化

初始化是指初始种群的初始化，一般选用均匀随机方法给定初始种群，例如种群中某个个体（解向量）：

$$x_{i,d} = rand \cdot \left(R_{u,d} - R_{l,d} \right) + R_{l,d}, i = 1,2,...,N_p, d = 1,2,...,D \qquad (1.11)$$

式中，$x_{i,d}$ 是解向量 $\mathbf{x}_i$ 的第 d 个元素；$rand$ 函数可以取 0 到 1 之间的均匀随机数；

$R_{u,d}$ 和 $R_{l,d}$ 分别为 $x_{i,d}$ 的上限和下限；N_{p} 表示种群规模，也就是种群中解向量的个数；D 表示解向量 $\mathbf{x}_i$ 的维度，也就是待优化参数的个数。

（2）变异

变异与其他随机扰动变异算法不同，DE 中是通过种群中两个成员之间的加权差向量加到第三个成员上来产生新的参数向量，这样做要不随机给定扰动更能找到更优解。对每个解向量都要执行变异操作：

$$\mathbf{v}_{i,g+1}=\mathbf{x}_{\mathrm{r1},g}+F\cdot\left(\mathbf{x}_{\mathrm{r2},g}-\mathbf{x}_{\mathrm{r3,g}}\right),i=1,2,...,N_{\mathrm{p}} \tag{1.12}$$

式中，变量下标 g 表示这一代；g+1 表示下一代；r1，r2 和 r3 为互不相等的三个随机整数且与 k 也不相等，r1，r2 和 r3 都在 1 到 N_{p} 之间，即从种群所有解向量中随机抽取三个解向量 $\mathbf{x}_{\mathrm{r1},g}$、$\mathbf{x}_{\mathrm{r2},g}$、$\mathbf{x}_{\mathrm{r3},g}$ 进行变异，生成一个新的解向量 $\mathbf{v}_{i,g+1}$，新的解向量 $\mathbf{v}_{i,g+1}$ 构成一个新的种群。由于变异操作的存在，种群规模 N_{p} 要大于 3。变异算子 $F\in[0,2]$ 是一个实常数因数，它控制偏差变量的缩放。变异算子中 F 过小，则可能造成算法“早熟”。随着 F 值的增大，防止算法陷入局部最优的能力增强，但当 F＞1 时，想要算法快速收敛到最优值会变得十分不易；这是由于当差分向量的扰动大于两个个体之间的距离时，种群的收敛性会变得很差。目前的研究表明，F 小于 0.4 和大于 1 的值仅偶尔有效，F=0.5 通常是一个较好的初始选择，在此采用 F=0.5。若种群过早收敛，那么 F 应该增大。

（3）交叉

为了增加干扰参数向量的多样性，引入交叉操作：

$$u_{i,d,g+1}=\begin{cases}v_{i,d,g+1} & rand<CR\text{或}i=randi\\ x_{i,d,g+1} & \text{其他}\end{cases} \tag{1.13}$$

式中，$v_{i,d,g+1}$ 为式(1.12)中 $\mathbf{v}_{i,g+1}$ 的第 d 个元素；$x_{i,d,g+1}$ 为第 g+1 代 $\mathbf{x}_i$ 的第 d 个元素；*rand* 函数产生一个 0 到 1 之间的均匀随机数；*randi* 函数产生一个 1 到 D 之间的均匀随机整数；$u_{i,d,g+1}$ 为交叉后的解向量 $\mathbf{u}_{i,g+1}$ 的元素。交叉算子 CR 是大于 0 小于 1 的实数，它控制着一个试验向量参数来自随机选择的变异向量而不是原来向量的概率。交叉算子中 CR 越大，发生交叉的可能性就越大。但较大的 CR 通常会加速收敛，CR 的一个较好的选择是 0.1，在此采用 CR=0.1。

（4）选择

为决定试验向量$\mathbf{u}_{i,g+1}$是否会成为下一代中的成员，差分进化算法按照贪婪准则将试验向量与当前种群中的目标向量$\mathbf{x}_{i,g}$进行比较。如果目标函数要被最小化，那么具有较小目标函数值的向量将在下一代种群中出现。下一代中的所有个体都比当前种群的对应个体更佳或者至少一样好。注意：在差分进化算法选择程序中，试验向量只与一个个体相比较，而不是与现有种群中的所有个体相比较。

（5）边界条件处理

在有边界约束的问题中，必须保证产生新个体的参数值位于问题的可行域中，一个简单方法是将不符合边界约束的新个体用在可行域中随机产生的参数向量代替，产生方法如式(1.11)。另外一个方法是进行边界吸收处理，即将超过边界约束的个体值设置为临近的边界值。

差分进化算法采用实数编码、基于差分的简单变异操作和“一对一”的竞争生存策略，其具体步骤如下：

（1）确定差分进化算法的控制参数和所要采用的具体策略。差分进化算法的控制参数包括：种群数量、变异算子、交叉算子、最大进化代数、终止条件等。

（2）随机产生初始种群，进化代数 k=l。

（3）对初始种群进行评价，即计算初始种群中每个个体的目标函数值。

（4）判断是否达到终止条件或达到最大进化代数：若是，则进化终止，将此时的最佳个体作为解输出；否则，继续下一步操作。优化过程中如果经过 0.02×G 代迭代目标函数值仍无下降，则将种群中一半的个体重新均匀随机初始化以增加多样性，避免早熟。

（5）进行变异操作和交叉操作，对边界条件进行处理，得到临时种群。

（6）对临时种群进行评价，计算临时种群中每个个体的目标函数值。

（7）对临时种群中的个体和原种群中对应的个体，进行“一对一”的选择操作，得到新种群。

（8）进化代数 k=k+1，转步骤（4）。

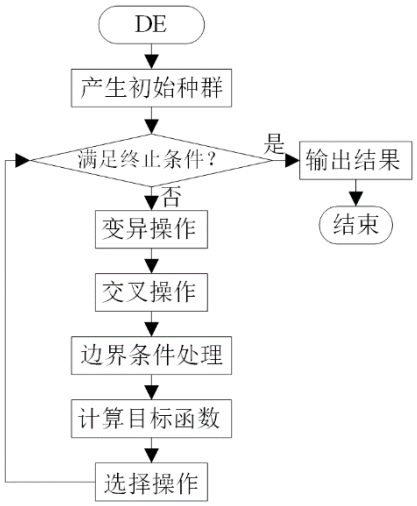


图 8　差分进化（DE）算法计算流程

DE 算法的优点是可以全局寻优，但 DE 算法也属于基于种群进化算法，不可避免的有进化算法存在的早熟收敛（过早的陷入局部最优解）和搜索停滞等缺陷方面问题[24]。针对以上缺点，可以通过改进初始种群的均匀性和添加局部搜索算子进行改进。

4.1.5 免疫算法（IA）

免疫算法（IA）受生物免疫系统的启发而推出的一种新型的智能搜索算法。它是一种其确定性和随机性选择相结合的、具有“勘探”与“开采”能力的启发式随机搜索算法。免疫算法将优化问题中待优化的问题对应免疫应答中的抗原，可行解对应抗体（B 细胞），可行解质量对应免疫细胞与抗原的亲和度。如此则可以将优化问题的寻优过程与生物免疫系统识别抗原并实现抗体进化的过程对应起来，将生物免疫应答中的进化过程抽象为数学上的进化寻优过程，形成一种智能优化算法。免疫算法是对生物免疫系统机理抽象而得的，算法中的许多概念和算子与免疫系统中的概念和免疫机理存在着对应关系。免疫算法与生物免疫系统概念的对应关系如表 4.1 所示。由于抗体是由 B 细胞产生的，在免疫算法中对抗体和 B 细胞不进行区分，都对应为优化问题的可行解。

表 2　免疫算法与生物免疫系统概念的对应关系

生物免疫系统	免疫算法
抗原	优化问题
抗体（B 细胞）	优化问题的可行解
亲和度	可行解的质量

细胞活化	免疫选择
细胞分化	个体克隆
亲和度成熟	变异
克隆抑制	克隆抑制
动态维持平衡	种群刷新

根据上述的对应关系，模拟生物免疫应答的过程形成了用于优化计算的免疫算法。算法主要包含以下几大模块：（1）抗原识别与初始抗体产生。根据待优化问题的特点设计合适的抗体编码规则，并在此编码规则下利用问题的先验知识产生初始抗体种群。（2）抗体评价。对抗体的质量进行评价，评价准则主要为抗体亲和度和个体浓度，评价得出的优质抗体将进行进化免疫操作，劣质抗体将会被更新。（3）免疫操作。利用免疫选择、克隆、变异、克隆抑制、种群刷新等算子模拟生物免疫应答中的各种免疫操作，形成基于生物免疫系统克隆选择原理的进化规则和方法，实现对各种最优化问题的寻优搜索。

与遗传算法等其他智能优化算法类似，免疫算法的进化寻优过程也是通过算子来实现的。免疫算法的算子包括：亲和度评价算子、抗体浓度评价算子、激励度计算算子、免疫选择算子、克隆算子、变异算子、克隆抑制算子和种群刷新算子等。由于算法的编码方式可能为实数编码、离散编码等，不同编码方式下的算法算子也会有所不同。

（1）亲和度评价算子

亲和度表征免疫细胞与抗原的结合强度，与遗传算法中的适应度类似。亲和度评价算子通常是一个函数 $f\left(\mathbf{x}\right),\mathbf{x}\in\mathbb{R}^{D}$，其中 $\mathbf{x}$ 为问题的可行解，$\mathbb{R}$ 为实数域，D 表示可行解维度。函数的输入为一个抗体个体（可行解），输出即为亲和度评价结果。曲线拟合设计的误差函数一般越小越好，在此，可将误差函数取倒数或相反数作为亲和度，亲和度越大代表误差函数越小，抗体质量越好。

（2）抗体浓度评价算子

抗体浓度表征抗体种群的多样性好坏。抗体浓度过高意味着种群中非常类似的个体大量存在，则寻优搜索会集中于可行解区间的一个区域，不利于全局优化。因此优化算法中应对浓度过高的个体进行抑制，保证个体的多样性。抗体浓度 $den\left(\mathbf{x}_{k}\right)$ 通常定义为：

$$den(\mathbf{x}_k)=\frac{1}{N_{\mathrm{p}}}\sum_{l=1}^{N_{\mathrm{p}}}S(\mathbf{x}_k,\mathbf{x}_l) \tag{1.14}$$

式中，N_{p} 表示种群规模，即种群内抗体个数；$S(\mathbf{x}_k,\mathbf{x}_l)$ 表示抗体间的相似度：

$$S(\mathbf{x}_k,\mathbf{x}_l)=\begin{cases}1 & d(\mathbf{x}_k,\mathbf{x}_l)<\delta_S\\ 0 & d(\mathbf{x}_k,\mathbf{x}_l)\geq\delta_S\end{cases} \tag{1.15}$$

式中，δ_S 表示相似度阈值，一般取 0.2。$d(\mathbf{x}_k,\mathbf{x}_l)$ 表示抗体 $\mathbf{x}_k$ 与抗体 $\mathbf{x}_l$ 之间的欧氏距离：

$$d(\mathbf{x}_k,\mathbf{x}_l)=\sqrt{\sum_{d=1}^{D}\left(\mathbf{x}_{k,d}-\mathbf{x}_{l,d}\right)^2} \tag{1.16}$$

所以抗体浓度 $den(\mathbf{x}_k)$ 是指抗体 $\mathbf{x}_k$ 与其他所有抗体距离远近的一个衡量。抗体浓度越大表示抗体 $\mathbf{x}_k$ 与其他抗体越近，这样的抗体越多表示种群多样性越差。

（3）激励度计算算子

其他优化算法大多采用 $f(\mathbf{x})$ 作为评价函数（Evaluation function, E*f*），IA 算法为了增加种群的多样性，将抗体浓度也考虑进去共同作为评价函数，也就是激励度。亲和度大、浓度低的抗体会得到较大的激励度。抗体激励度：

$$Ef(\mathbf{x}_k)=a\cdot f(\mathbf{x}_k)-b\cdot den(\mathbf{x}_k) \tag{1.17}$$

式中，a 和 b 一般取常数 1，表示亲和度和抗体浓度的权重。

（4）免疫选择算子

免疫选择算子根据抗体的激励度确定选择哪些抗体进入克隆选择操作。认为抗体激励度越大的质量越好。

（5）克隆算子

克隆算子将免疫选择算子选中的抗体个体进行复制，可以记为克隆 m 个，m>1。

（6）变异算子

变异算子对克隆算子得到的抗体克隆结果进行变异操作，以产生亲和度突变，实现局部搜索。变异算子是免疫算法中产生有潜力的新抗体、实现区域搜索的重要算子，它对算法的性能有很大影响。实数编码算法的变异策略是在变异源个体

中加入一个小的随机扰动，使其稍偏离原来的位置，落入源个体邻域中的另一个位置，实现变异源邻域的搜索。实数编码算法变异算子可以描述为：

$$P_m\left(\mathbf{x}_{k,d,m}\right)=\begin{cases}\mathbf{x}_{k,d,m}+\left(rand-0.5\right)\cdot\delta & rand<p_{\mathrm{m}}\\ \mathbf{x}_{k,d,m} & \text{其他}\end{cases} \tag{1.18}$$

式中，$\mathbf{x}_{k,d,m}$是抗体$\mathbf{x}_k$的第 m 个克隆体的第 d 维；δ 为扰动系数，可以设定为常数也可以根据进化调整；*rand* 函数可以产生 0 到 1 之间的均匀随机数；p_{m} 为抗体内元素的变异概率，可以设为 0.7。

（7）克隆抑制算子

克隆抑制算子用于对经过变异后的克隆体进行再选择，抑制亲和度低（误差函数高）的抗体，保留亲和度高的抗体进入新的抗体种群。

（8）种群刷新算子

种群刷新算子删除种群中激励度较低的抗体，进而用随机产生的抗体代替。

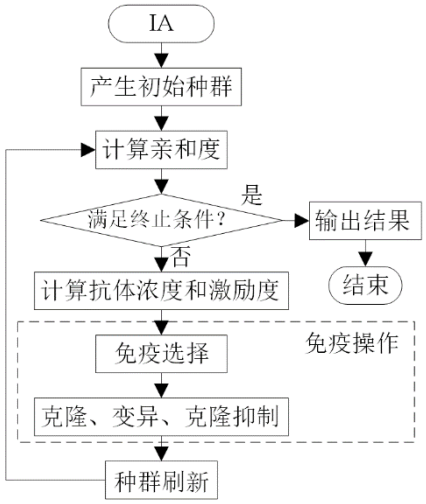


图 9　免疫算法（IA）流程

免疫算法的流程如图 9 所示，分为以下几个步骤：

（1）根据要优化的问题及可行解范围生成初始抗体种群。

（2）对种群中的每一个可行解进行亲和度评价。

（3）判断是否满足算法终止条件：如果满足条件，则终止免疫算法，输出计算结果；否则，继续寻优运算。

（4）根据式(1.14)和(1.17)计算抗体浓度和激励度。

（5）进行免疫操作，包括免疫选择、克隆、变异和克隆抑制。

（6）种群刷新，以随机生成的新抗体替代种群中激励度较低的抗体，形成新

一代抗体，转步骤（2）。

免疫算法是模仿生物免疫机制，结合基因的进化机理，人工构造出的一种新型智能优化算法，因而具有一般免疫系统的特征。它采用群体搜索策略，通过迭代计算，最终以较大的概率得到问题的最优解。免疫算法具有自适应性、随机性、并行性、全局收敛性、种群多样性等优点。相比于其他算法，免疫算法利用自身产生多样性和维持机制的特点，保证了种群的多样性，克服了一般寻优过程（特别是多峰值的寻优过程）中不可避免的“早熟”问题，可以求得全局最优解。但由于保持了解之间的间距，因此局部搜索不够精细。

4.1.6 粒子群算法（PSO）

经典粒子群优化(PSO)算法最初由 Kennedy 和 Eberhart 于 1995 年开发，用于优化非线性函数和训练神经网络[15,25]。鸟类在捕食过程中，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。粒子群算法受鸟类捕食行为的启发并对这种行为进行模仿，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粒子，粒子无质量、无体积，用以表征问题的一个可行解，优化问题所要搜索到的最优解则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简单行为规则，使整个粒子群的运动表现出与鸟类捕食相似的特性，从而可以求解复杂的优化问题。

1998 年，Shi Yuhui 等[26]提出了带有惯性权重的改进粒子群算法，由于该算法能够保证较好的收敛效果，所以被默认为标准粒子群算法。其明显的特征是含有惯性权重的粒子的速度：

$$\mathbf{v}_{g+1} = w \cdot \mathbf{v}_g + c_1 \cdot r_1 \cdot \left(\mathbf{xp}_g - \mathbf{x}_g\right) + c_2 \cdot r_2 \cdot \left(\mathbf{xg}_g - \mathbf{x}_g\right) \tag{1.19}$$

式中，$\mathbf{x}_g$ 表示第 g 代某粒子或个体的位置，也代表了待求问题的解向量；$\mathbf{xp}_g$ 表示某粒子截止到第 g 代出现过最好的位置；$\mathbf{xg}_g$ 代表所有粒子中截止到第 g 代出现过的最好位置，也表示了目前所知的全局最优解向量；$\left(\mathbf{xp}_g - \mathbf{x}_g\right)$表示某粒子的速度受此粒子历史上最好的位置的影响；$\left(\mathbf{xg}_g - \mathbf{x}_g\right)$表示某粒子的速度也受所有粒子历史上出过的最好的位置的影响；c_1和c_2分别是学习因子，可都取 1.5；r_1和r_2为 0 到 1 之间的均匀随机数；$\mathbf{v}_g$ 表示第 g 代某粒子的速度，代表了解向量

的变化速率；惯性权重 w 用来缩放第 g 代某粒子的速度 $\mathbf{v}_g$ 对第 g+1 代此粒子的速度 $\mathbf{v}_{g+1}$ 影响的大小，随代数 g 的变化而变化：

$$w = w_{\max} - \frac{(w_{\max} - w_{\min}) \cdot g}{G} \tag{1.20}$$

式中，$w_{\max}$ 为最大惯性权重，可取 0.9；$w_{\min}$ 为最小惯性权重，可取 0.4；G 代表代数 g 可取的最大值。惯性权重 w 越大，表示某粒子第 g+1 代的速度受第 g 代的速度影响更大，全局搜索能力越强，局部搜索能力越弱。相反，如果惯性权重 w 越小，则全局搜索能力较弱，而局部搜索能力较强。

根据上述粒子更新后的速度 $\mathbf{v}_{g+1}$，就可以更新此粒子的位置：

$$\mathbf{x}_{g+1} = \mathbf{x}_g + \mathbf{v}_{g+1} \tag{1.21}$$

式中，$\mathbf{x}_g$ 为粒子更新前的位置，$\mathbf{x}_{g+1}$ 为粒子根据本身当前速度 $\mathbf{v}_{g+1}$ 更新后的粒子位置，粒子的更新过程如图 10 所示。

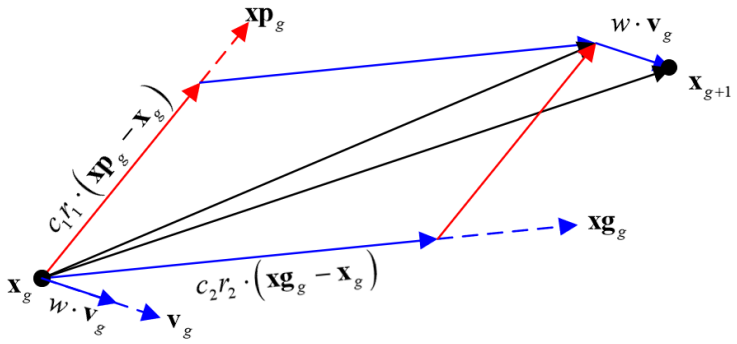


图 10　PSO 算法中粒子位置（解向量）更新过程

粒子群算法的大致步骤如下：

（1）设定粒子群算法参数，包括最大迭代代数 G，速度的惯性权重系数 w 的最大值 $w_{\max}$ 和最小值 $w_{\min}$，c_1 和 c_2 等。

（2）初始化粒子位置和速度。根据待优化问题中各待优化参数上下限初始化种群，使种群中每个粒子的位置都被赋值。待优化参数上下限的 0.2 倍作为粒子速度上下限，并根据粒子速度的上下限类似于粒子的位置进行粒子速度的初始化。

（3）计算每个粒子的目标函数值，记录每个粒子历史上最优值 $\mathbf{xp}_g$ 和群体中所有粒子历史上最优位置 $\mathbf{xg}_g$。

（4）对每个粒子的速度和位置进行更新，并对粒子的位置进行边界检查是指永远在各待优化参数的上下限范围内。

（5）判断是否满足边界条件，如果不满足则转到第（2）步，如果满足则输出 $\mathbf{xg}_g$ 并结束程序。

粒子群算法属于群体智能算法，通过 $\mathbf{xg}_g$ 进行粒子间信息交换，它没有交叉、变异等复杂操作，仅依靠“位移-速度”模型不断改变粒子位置在解空间进行搜索。粒子群算法计算速度快，能以较大概率收敛于全局最优解。实践证明，它适合在动态、多目标优化环境中寻优。

4.1.7 蚁群算法（ACO）

20 世纪 90 年代初期，M.Dorigo，V.Maniezzo 和 A.Colorni 等[14]通过模拟自然界中蚂蚁集体寻径行为提出了蚁群算法。蚂蚁寻找食物时会在走过的路径上残留信息素，信息素会随时间消散，其他蚂蚁在寻找食物时有更大概率走信息素浓度较大的路径，路径越短则蚂蚁在巢穴和食物间往返次数越多，越短的路径上残留的信息素也越浓，这条较短路径上就会吸引更多蚂蚁走过，又会留下更多的信息素，形成越短路径信息素越逐渐变浓的正向循环。蚂蚁走过的路径的结点可以看做是解向量元素，信息素浓度可以看做此解向量被选择的概率。蚁群算法有选择算子和生成新解向量两个算子：

（1）轮盘赌选择算子

在蚁群算法中，同样采用轮盘赌选择算子，某个体被选中的概率 p_i：

$$p_i = w_i \Big/ \left(\sum_{i=1}^{N_\mathrm{p}} w_i \right) \tag{1.22}$$

式中第 i 条路径或第 i 个解向量的权重 w_i：

$$w_i = \frac{\exp\left(-0.5 \left(\dfrac{i-1}{sp \cdot N_\mathrm{p}} \right)^2 \right)}{\sqrt{2\pi} \cdot sp \cdot N_\mathrm{p}} \tag{1.23}$$

式中，sp 为强化系数，表示选择压力大小，一般取 0.5；N_p 是种群规模，即种群中路径或解向量数目。此时当 i=1 是 p_i 最大，随着 i 的增大 p_i 逐渐减小。由此可

知需要各解向量的目标函数值按升序排序后使用 p_i 作为第 i 个解向量被选中的概率。升序后第 1 个解向量目标函数值最小，被选中的概率越大，p_i 能满足此需求。

（2）生成新的解向量算子

根据第 g 代解向量 $\mathbf{x}_g$ 和解向量元素间距离可以生成新的解向量 $\mathbf{x}_{g+1}$：

$$x_{i,d,g+1} = x_{i,d,g} + den\left(x_{i,d,g}\right) \cdot randn \tag{1.24}$$

式中，$x_{i,d,g}$ 为第 g 代种群中编号为 i 的解向量 $\mathbf{x}_i$ 的第 d 个元素；$randn$ 表示标准正太分布的随机数；$den\left(x_{i,d,g}\right)$ 表示第 g 代解向量 $\mathbf{x}_i$ 的第 d 个元素与第 g 代其他所有解向量的第 d 个元素的偏差距离：

$$den\left(x_{i,d,g}\right) = \frac{\zeta}{N_{\mathrm{p}} - 1} \sum_{l=1}^{N_{\mathrm{p}}} \left| x_{i,d,g} - x_{i,l,g} \right| \tag{1.25}$$

式中，ζ 是偏差距离比率，一般取 1；N_{p} 是种群规模。

蚁群算法的主要流程如下：

（1）设定蚁群算法参数，包括强化系数 sp、偏差距离比率 ζ、种群规模 N_{p}、最大允许迭代代数 G 等。

（2）在解向量上下限范围内均匀生成含 N_{p} 个解向量的初始种群，并计算每个解向量的目标函数，之后按目标函数值大小升序排列各解向量，记为 Pop1。根据式(1.22)计算解向量编号为 i 的选中概率 p_i。

（3）根据式(1.25)计算种群 Pop1 中每个解向量元素与其他解向量元素的偏差距离。

（4）根据轮盘赌算子和解向量选中的概率 p_i，选出式(1.24)所需的 $\mathbf{x}_i$，并根据式(1.24)生成一个新的解向量。上述选出 $\mathbf{x}_i$ 并生成新的解向量的过程执行 $4 \times N_{\mathrm{p}}$ 次，$4 \times N_{\mathrm{p}}$ 个新的解向量形参新的解向量种群 Pop2。

（5）将旧种群 Pop1 和新种群 Pop2 合并，合并后按升序排列。用排序后的所有个体的 N_{p} 个解向量覆盖种群 Pop1 中 N_{p} 个旧的解向量。

（6）检查是否满足停止条件，如果不满足转到第（3）步，如果满足则输出

最优解向量并终止程序。

蚁群算法一般需要较长的搜索时间和容易出现停滞现象等不足，但在求解旅行商（TSP）问题、分配问题、车间调度等问题上取得了较好的效果。

4.1.8 多种自适应策略的粒子群算法（MAPSO）

为了更好地平衡勘探（全局搜索）和开发（局部搜索）的能力，Wei 等[27]提出了多种自适应策略的粒子群优化算法（MAPSO），在复杂问题上比 PSO 具有很好的性能。自适应策略包括自动选择学习粒子（ALE）策略和自动调整种群大小（APS）策略。相比于 PSO 算法，MAPSO 算法最鲜明的特色是将一个种群分为了 N_s 个小的子群，每个子群按目标函数值 $f(\mathbf{x})$ 从小到大分为 3 种粒子或解向量，分别为精英粒子 E_i、中庸粒子 M_i 和劣等粒子 I_i，如图 11 所示。其中每个子群截止到目前出现过的最优粒子记为 B_i，不同子群中所有 B_i 组成集合 **B**，B_i 是所有个体的第一个学习个体。

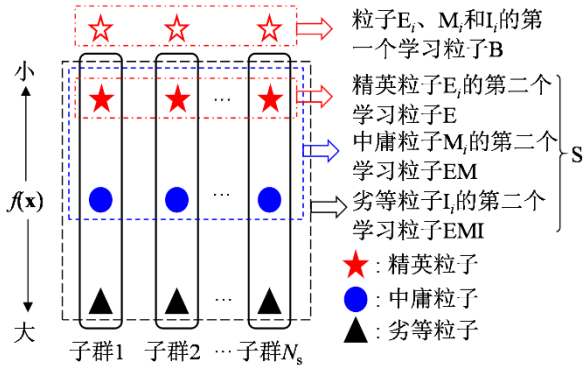


图 11　不同粒子的学习粒子

不同子群中所有精英粒子 E_i 组成集合 **E**，不同子群中所有精英粒子 E_i 和中庸粒子 M_i 组成集合 **EM**，不同子群中所有精英粒子 E_i、中庸粒子 M_i 和劣等粒子 I_i 组成集合 **EMI**。集合 **E**、**EM** 或 **EMI** 中粒子为某粒子的第 2 个学习粒子。具体选择哪个粒子作为第 2 个学习粒子，要看当前粒子属于哪类粒子。如果当前粒子属于劣等粒子 I_i，则从集合 **EMI** 中随机选取一个粒子作为 I_i 的第二个学习粒子。上述根据粒子自身种类自动选取相应集合内粒子作为学习粒子的过程就是自动选择学习粒子（ALE）策略。在此，记集合 **S** 为任意粒子的第二个学习粒子，则任意粒子的速度可按下式更新：

$$\mathbf{v}_{g+1} = w \cdot \mathbf{v}_g + r_1 \cdot \left(\mathbf{B}_g - \mathbf{x}_g\right) + r_2 \cdot \left(\mathbf{S}_g - \mathbf{x}_g\right) \tag{1.26}$$

式中，$\mathbf{v}_g$ 表示第 g 代当前粒子的速度；$\mathbf{B}_g$ 为第 g 代当前粒子的第一个学习粒子的速度，第一个学习粒子是从集合 **B** 中随机选取的粒子；$\mathbf{S}_g$ 为第 g 代当前粒子的第二个学习粒子的速度，第 2 个学习粒子是根据 ALE 策略选取的；r_1 和 r_2 为两个 0 到 1 之间的均匀分布随机数。w 为惯性权重系数为 0.9 - 0.8 × g/ G，其中 g 为当前代数，G 为最大迭代代数。

粒子的速度更新后，同 PSO 算法中粒子位置更新公式(1.21)一样进行粒子位置（解向量）更新，如图 12 所示。

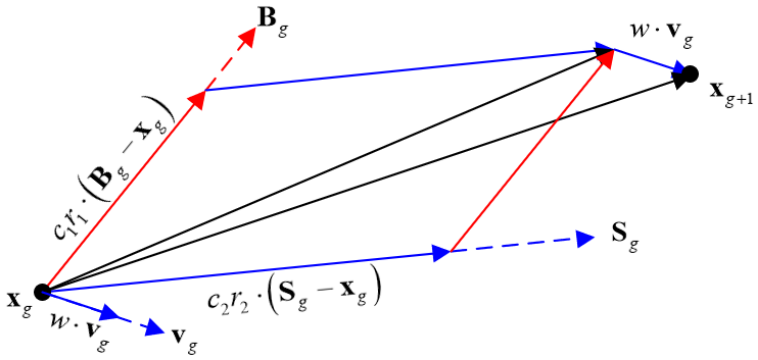


图 12　PSO 算法中粒子位置（解向量）更新过程

为了合理分配计算资源，MAPSO 优化算法还在进化过程中对种群大小进行了自适应调整(APS)，包括删除粒子和添加粒子两个过程。APS 的主要思想是当问题易于优化时从当前种群中删除一些不利的粒子。相反，如果当前种群不能找到更有希望的解，一些有潜力的粒子会通过差分进化算法生成并被添加到当前的群体中。由于 APS 策略在执行时不可能不限制种群大小，所以使用两个预先定义的整数 $N_{\max}$ 和 $N_{\min}$ 来在区间内限制种群大小。

MAPSO 优化算法的主要步骤如下：

（1）设定优化算法参数，包括子群数目 Ns、最大迭代代数 G、种群的最大最小种群大小 $N_{\max}$ 和 $N_{\min}$ 等。

（2）根据参数上下限随机给定粒子位置和粒子速度作为初始种群中粒子位置和速度，并求得每个粒子的目标函数值。

（3）根据目标函数值对所有粒子排序，并将所有粒子按图 11 分为 N_s 个子群，并给集合 **B**、**E**、**EM** 和 **EMI** 指定粒子。

（4）根据式(1.26)和式式(1.21)更新每个粒子的速度和位置。

（5）重复第（3）步得到新的 **B**、**E**、**EM** 和 **EMI**。

（6）根据 APS 测量增加或删除粒子。

（7）重复第（4）步直到迭代代数 g 达到最大迭代代数 G 或当前调用目标函数的次数达到目标函数允许的最大调用次数。

（8）从集合 **B** 中选择目标函数值最小的粒子输出粒子位置作为最优解向量。

4.1.9 几种典型优化算法优缺点及改进方向

前面简要介绍了几种典型智能优化算法，包括模拟退火（SA）算法、状态转移（STA）算法、实数编码遗传（RCGA）算法、差分进化（DE）算法、免疫（DE）算法、粒子群（DE）算法、蚁群（DE）算法、多种自适应策略粒子群（MAPSO）算法。下面将各种算法的优缺点及改进方向汇总于表 3 中。

表 3　几种典型优化算法的优缺点及改进方向

优化算法	优点	缺点	改进方向
SA	①算法简单便于实现 ②可靠性高	①实际运行效率不高 ②精度稍差	①增加记忆、重升温以及多次搜索对比功能 ②与其他算法结合
STA	①四个搜索算子都能够产生具有规则形状、可控大小的几何邻域，易理解 ②速度快	①流程稍复杂 ②在某些复杂高维函数寻优后期表现出收敛慢、精度低的问题	①引入其他机制的局部搜索算子
RCGA	①可以以决策变量的编码为计算对象，适用性广 ②隐含并行性 ③具有自组织、自适应、自学习特性	①早熟收敛 ②局部搜索能力差	①改进编码机制 ②改进遗传算子 ③改进控制参数 ④与其他算法结合
DE	①全局搜索能力强、收敛速度快 ②结构简单、性能优越	①早熟收敛 ②可能陷入局部最优解	①增加个体多样性 ②与其他算法结合
IA	①多样性保持机制利于全局搜索 ②并行分布式搜索机制，可以得到次优解	①计算量偏大、速度慢 ②概念和计算流程偏复杂	①使克隆、变异和克隆抑制等算子自适应变化 ②与遗传算法等结合
PSO	①概念简单、控制参数少 ②可以得到次优解 ③对种群大小不敏感 ④可根据粒子记忆功能调整搜索策略	①易陷入局部最优 ②早熟收敛	①改进算法控制参数 ②增加个体多样性 ③与其他算法结合
ACO	①概念简单易于实现 ②对计算机硬件要求低	①搜索时间过长 ②易陷入局部最优	①避免陷入局部最优 ②与其他优化算法相结合
MAPSO	①自适应选择学习样本，全局优化能力强	①算法复杂 ②在单峰函数上耗时多	①更恰当的时机改变学习样本

②自适应种群大小节省计算时间	②更有效的策略调整种群规模

参考文献:

[1] 高岳林, 杨钦文, 王晓峰, 李嘉航, et al. 新型群体智能优化算法综述 [J]. 郑州大学学报(工学版): 1-10.

[2] 周雪刚. 非凸优化问题的全局优化算法 [D]; 中南大学, 2010.

[3] Katoch, S, Chauhan, S S, Kumar, V. A review on genetic algorithm: past, present, and future [J]. Multimedia Tools Applications, 2021, 80(5): 8091-8126.

[4] Kirkpatrick, S, Gelatt Jr, C D, Vecchi, M P. Optimization by simulated annealing [J]. Science, 1983, 220(4598): 671-680.

[5] Zhou, X, Yang, C, Gui, W. Initial Version of State Transition Algorithm; proceedings of the 2011 Second International Conference on Digital Manufacturing & Automation, F 5-7 Aug. 2011, 2011 [C].

[6] Glover, F. Future paths for integer programming and links to artificial intelligence [J]. Computers operations research, 1986, 13(5): 533-549.

[7] Jiang, X, Li, S. BAS: Beetle Antennae Search Algorithm for Optimization Problems [Z]. 2017

[8] Creutz, M. Microcanonical monte carlo simulation [J]. Physical Review Letters, 1983, 50(19): 1411.

[9] Papadimitriou, C H, Steiglitz, K. Combinatorial optimization: algorithms and complexity [M]. Courier Corporation, 1998.

[10] 田冉, 孙林夫, 唐慧佳, 赵进超. 求解卸装一体化车辆路径问题的改进导向局部搜索算法 [J]. 科学技术与工程, 2015, 15(18): 66-70.

[11] Holland, J H. Adaptation in natural and artificial systems [M]. AnnArbor: University of Michigan Press, 1975.

[12] Storn, R, Price, K. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces [J]. Journal of Global Optimization, 1997, 11(4): 341-359.

[13] Jerne, N K. Towards a network theory of the immune system [J]. COLLECTANNINSTPASTEUR, 1974, 125 C(1-2): 373-389.

[14] Dorigo, M, Maniezzo, V, Colorni, A. Ant system: optimization by a colony of cooperating agents [J]. IEEE Transactions on Systems, Man, Cybernetics, Part B, 1996, 26(1): 29-41.

[15] Kennedy, J, Eberhart, R. Particle swarm optimization; proceedings of the Proceedings of ICNN'95-international conference on neural networks, F, 1995 [C]. IEEE.

[16] Pan, W T. A new fruit fly optimization algorithm: taking the financial distress model as an example [J]. Knowledge-Based Systems, 2012, 26: 69-74.

[17] Yang, X S. A new metaheuristic bat-inspired algorithm [M]. Nature inspired cooperative strategies for optimization (NICSO 2010). Springer. 2010: 65-74.

[18] Mirjalili, S, Lewis, A. The whale optimization algorithm [J]. Advances in engineering software, 2016, 95: 51-67.

[19] Heidari, A A, Mirjalili, S, Faris, H, Aljarah, I, et al. Harris hawks optimization: Algorithm and applications [J]. Future generation computer systems, 2019, 97: 849-872.

[20] 包子阳, 余继周, 杨杉. 智能优化算法及其 MATLAB 实例（第 3 版） [M]. 北京: 电子工业出版社, 2021.

[21] 黄淼. 基于状态转移算法的一类动态优化方法研究 [D]; 中南大学, 2019.

[22] Goldberg, D E. Real-coded genetic algorithms, virtual alphabets, and blocking [J]. Complex Systems, 1991, 5(2): 139--167.

[23] 玄光男. 遗传算法与工程优化 [M]. 清华大学出版社, 2004.

[24] Ali, M, Pant, M. Improving the performance of differential evolution algorithm using Cauchy mutation [J]. Soft Computing, 2011, 15(5): 991-1007.

[25] Eberhart, R, Kennedy, J. A new optimizer using particle swarm theory; proceedings of the MHS'95 Proceedings of the Sixth International Symposium on Micro Machine and Human Science, F, 1995 [C]. Ieee.

[26] Shi, Y, Eberhart, R. A modified particle swarm optimizer; proceedings of the 1998 IEEE international conference on evolutionary computation proceedings IEEE world congress on computational intelligence (Cat No 98TH8360), F, 1998 [C]. IEEE.

[27] Wei, B, Xia, X W, Yu, F, Zhang, Y L, et al. Multiple adaptive strategies based particle swarm optimization algorithm [J]. Swarm Evolutionary Computation, 2020, 57: 100731.