基于状态估计反馈的策略自适应差分进化算法

王柳静 [1] 张贵军 [1] 周晓根 [1]

摘 要 借鉴闭环控制思想, 提出基于状态估计反馈的策略自适应差分进化 (Differential evolution, DE) 算法, 通过设计状态评价因子自适应判定种群个体所处于的阶段, 实现变异策略的反馈调节, 达到平衡算法全局探测和局部搜索的目的. 首先, 基于抽象凸理论对种群个体建立进化状态估计模型, 提取下界估计信息并结合进化知识设计状态评价因子, 以判定当前种群的进化状态; 其次, 利用状态评价因子的反馈信息, 实现不同进化状态下策略的自适应调整以指导种群进化, 达到提高算法搜索效率的目的. 另外, 20 个典型测试函数与 CEC2013 测试集的实验结果表明, 所提算法在计算代价、收敛速度和解的质量方面优于主流改进差分进化算法和非差分进化算法.

关键词 差分进化, 状态估计, 反馈, 全局优化, 抽象凸

引用格式 王柳静, 张贵军, 周晓根. 基于状态估计反馈的策略自适应差分进化算法. 自动化学报, 2020, **46**(4): 752−766

DOI 10.16383/j.aas.2018.c170338

Strategy Self-adaptive Differential Evolution Algorithm Based on State Estimation Feedback

WANG Liu-Jing[1] ZHANG Gui-Jun[1] ZHOU Xiao-Gen[1]

Abstract Inspired by the idea of closed-loop control, a strategy self-adaptive differential evolution (DE) algorithm based on state estimation feedback is proposed, the stage of individual can be self-adaptively determined by designing the state judgment factor, and achieve the feedback adjustment of mutation strategies. Consequently, the algorithm can get a trade-off between the exploration and exploitation. Firstly, the estimation model of evolution state is established based on abstract convex theory, from which the underestimation information is extracted combining with the evolutionary information to design the state judgment factor, so that the evolution state of the current population is estimated. Secondly, according to the feedback information of the state judgment factor, the strategy in different evolution state is adaptively selected to guide the evolution of the population. Therefore, the searching efficiency of the algorithm can be improved. Additionally, experimental results of 20 benchmark functions and CEC2013 test set show that the proposed algorithm is superior to the main-stream differential evolution variants and non-differential evolution algorithms mentioned in this paper in terms of computational cost, convergence speed, and solution quality.

Key words Differential evolution (DE), state estimation, feedback, global optimization, abstract convex

Citation Wang Liu-Jing, Zhang Gui-Jun, Zhou Xiao-Gen. Strategy self-adaptive differential evolution based on state estimation feedback. *Acta Automatica Sinica*, 2020, **46**(4): 752−766

差分进化 (Differential evolution, DE) 算法在优化求解过程中采用个体之间的竞争与合作机制指导种群搜索全局最优解, 其特有的记忆功能使其可以根据当前的搜索情况, 动态调整搜索方向[1]. DE 算法具有原理简单、控制参数少、易于实现的优点, 在处理具有非线性、强约束、多目标、多模态等特性的复杂优化问题时具有良好的优化效果[2]. 目前, DE 已被广泛应用于工业工程[3]、生产管理[4]、电力系统[5]、生物信息[6]、化工[7]、传感器网络[8] 等各种实际优化问题的求解. 然而, DE 算法的搜索能力很大程度上依赖差分向量对当前候选个体的扰动[9], 因此, 算法初期全局探测能力较好, 然而随着个体之间的差异逐渐减小, 导致局部增强能力较差[10]. 显然, 传统 DE 算法中单一的策略和参数不能适应搜索过程中不同阶段的变化. 如何判定个体所处的阶段, 进而在此基础上设计阶段特定的策略和参数, 是提高算法搜索性能的关键.

针对上述问题, 近年来国内外学者提出了一些改进 DE 算法, 将整个搜索过程划分为多个阶段, 并根据不同阶段的特点设置相应策略和参数. Cheng 等[11] 提出了一种多目标优化的改进算法 (Two-phase differential evolution, TDE), 基于给定迭代次数将搜索过程设置为三个阶段: 第一阶段, 变异策略中所有向量通过随机选取

收稿日期 2017-06-20 录用日期 2017-12-06
Manuscript received June 20, 2017; accepted December 6, 2017
国家自然科学基金 (61773346, 61573317), 浙江省自然科学基金重点项目 (LZ20F030002) 资助
Supported by National Natural Science Foundation of China (61773346, 61573317) and Key Program of Natural Science Foundation of Zhejiang Province of China (LZ20F030002)
本文责任编委 刘艳军
Recommended by Associate Editor LIU Yan-Jun
1. 浙江工业大学信息工程学院 杭州 310023
1. College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023

以保持种群多样性; 第二阶段, 变异策略中基向量选自精英个体库, 其他向量从种群中随机挑选, 从而平衡种群多样性和收敛速度; 第三阶段, 变异策略中所有向量从精英个体库随机选取, 从而加快收敛速度. Liu 等[12] 提出两阶段优化机制的差分进化算法 (TSDE), 基于给定迭代次数将整个搜索过程划分为相同的两个阶段: 第一阶段采用 DE/current-to-rand/1 策略, 以增强算法探测能力; 第二阶段采用 DE/current-to-best/1 策略, 提高算法的收敛性能. Tang 等[13] 提出一种具有个体依赖机制的差分进化算法 (IDE), 基于候选个体进入下一代的成功率将搜索过程区分为两个阶段, 以设计特定阶段的变异策略, 当 SRg 趋近于零时, 表明种群多样性减小, 算法搜索状态切换至下一阶段. Yu 等[14] 提出两层参数自适应差分进化算法 (ADE), 通过适应值和个体之间的欧氏距离衡量优化状态, 将搜索过程划分为两个阶段, 分别采用相应的参数调节机制和策略; 张贵军等[15] 提出动态小生境半径两阶段差分进化算法 (DNTDE), 根据退火曲线将算法划分为两个阶段: 第一阶段采用差分限制变异策略增强种群多样性; 第二阶段实现生境内部的局部增强.

DE 算法的搜索过程一般分为全局探测 (Exploration) 和局部增强 (Exploitation) 两个阶段[16], 实质上, 种群个体所处阶段是动态变化的. TDE[11] 和 TSDE[12] 是根据问题先验知识实现阶段划分的, IDE[13] 通过统计概率、ADE[14] 通过衡量优化状态、DNTDE[15] 通过指定退火曲线实现阶段划分. 本文提出一种基于状态估计反馈的策略自适应差分进化算法 (SEFDE), 基于抽象凸理论估计进化状态, 以动态划分当前种群个体所处于的不同阶段, 进而通过相应变异策略的反馈调节, 来平衡算法全局探测与局部增强能力.

1　SEFDE 算法

现有 DE 算法实质上是一个开环优化过程, 无法通过输出来动态调整进化算子的策略和参数. 本文所提 SEFDE 算法通过引入反馈环节将种群个体的搜索过程动态划分为全局探测阶段 (Exploration) 和局部增强阶段 (Exploitation), 根据不同阶段的特点自适应切换变异策略, 达到指导种群进化的目的. 借鉴闭环控制思想, 状态评价因子的作用类比于控制量, 以实现变异策略的切换, 方法如下:

1) 输入种群个体, 首先从两种变异策略中择一进行变异操作, 然后通过交叉、选择操作生成子代种群, 实现 SEFDE 算法的前向通路;

2) 对子代种群建立进化状态估计模型, 基于所求得的个体估计值与其适应值建立子代种群的状态评价因子, 进而根据状态评价因子的变化检测每一代种群的分布, 从而控制对个体所处阶段的判定, 自适应切换变异策略, 实现 SEFDE 算法的反馈通路.

如图 1 所示. 具体过程为: ①进化状态的估计: 确定采样个体, 对当前种群构建进化状态估计模型以获取个体下界估计信息, 结合进化知识设计状态评价因子, 该指标包含了当前种群个体的进化状态信息. ②阶段的动态划分: 结合进化状态信息, 根据阶段判别模型实现种群个体所处阶段的动态划分, 切换特定阶段的全局探测或局部增强变异策略, 提高算法搜索的收敛速度和可靠性.

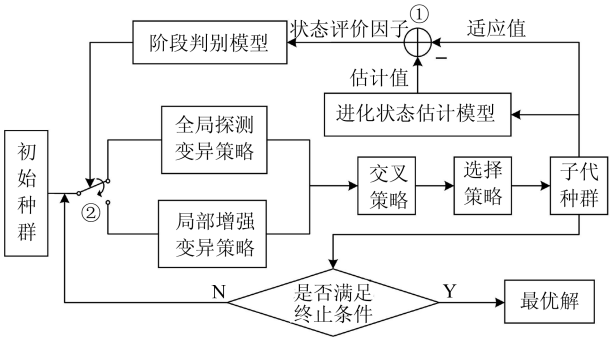


图 1　SEFDE 算法思想

Fig. 1　Main idea of SEFDE

1.1　进化状态估计模型

基于抽象凸理论[17−18] 建立进化状态估计模型, 随着该模型的逐渐收紧, 其与原目标函数之间的估计误差趋近于零. 进而, 基于上述思想能够通过估计误差值的变化对个体所处阶段进行判定.

定义 1. 对 D 内任意的 $\boldsymbol{x}_1, \boldsymbol{x}_2$, 若函数 $f: \boldsymbol{x} \to \mathbf{R}$ 均满足:

$$|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)| \leq M \left\|\boldsymbol{x}_1 - \boldsymbol{x}_2\right\| \tag{1}$$

则称函数 f 在可行域 D 中是 Lipschitz 连续的, M 被称作 Lipschitz 常数[19].

定义 2. 设 H 为可行域 D 内的函数族, 若函数 f 满足:

$$f(\boldsymbol{x}) = \sup\{h(\boldsymbol{x}) : h \in H\}, \forall \boldsymbol{x} \in D \tag{2}$$

则称函数 f 为关于函数族 H 的抽象凸函数.

设如下满足定义 1 的全局优化问题

$$\min\ f(\boldsymbol{x}), \boldsymbol{x} \in D \subset \mathbf{R}^N \tag{3}$$

其中, $\boldsymbol{x} = [x_1, x_2, \cdots, x_N]^{\mathrm{T}}$ 为 N 维连续优化变量, $f(\cdot)$ 为定义在可行域 D 上的 N 维目标函数.

假设种群规模为 NP, 根据当前第 g 代种群 $\{\boldsymbol{x}_i^g\}, i \in NP$ 的适应值升序排列筛选出前 K 个个体, $K < NP$, 记为采样个体 $\boldsymbol{x}'^g_k, k \in K$, 即采样个体 $\{\boldsymbol{x}'^g_k\}$ 为种群 $\{\boldsymbol{x}_i^g\}$ 的子集. 进而, 基于抽象凸理论, 根据采样点 $(\boldsymbol{x}'^g_k, f(\boldsymbol{x}'^g_k))$, 构建满足定义 2 的下

界估计模型 $h_k(\boldsymbol{x})$[20]:

$$h_k(\boldsymbol{x}_i^g) = \min_{j\in I}(f(\boldsymbol{x}'^g_k) - M(x'^g_{k,j} - x^g_{i,j})) = \min_{j\in I} M(l^g_{k,j} + x^g_{i,j}) \tag{4}$$

其中, $j \in I = \{1,\cdots,N+1\}$, $x^g_{i,j}$ 和 $x'^g_{k,j}$ 分别为 $\boldsymbol{x}_i^g$ 和 $\boldsymbol{x}'^g_k$ 的第 j 维向量, $k = 1,\cdots,K$, $M > 0$ 为 Lipschitz 常数, $l^g_{k,j}$ 为支撑向量 $\boldsymbol{l}^g_k$ 的第 j 维向量, 支撑向量 $\boldsymbol{l}^g_k$ 如式 (5) 所示:

$$\boldsymbol{l}^g_k = (l^g_{k,1},\cdots,l^g_{k,N+1}) = \left(\frac{f(\boldsymbol{x}'^g_k)}{M} - x'^g_{k,1},\cdots,\frac{f(\boldsymbol{x}'^g_k)}{M} - x'^g_{k,N+1}\right) \tag{5}$$

其中, $\boldsymbol{x}'^g_{k,N+1} = 1 - \sum_{j=1}^{N} \boldsymbol{x}'^g_{k,j}$ 为松弛变量[21].

以一维 Levy and Montalvo 2 (Lm2) 问题为例进行说明. 图 2 和图 3 中实线为 Lm2 的部分函数曲线, 虚线为下界估计模型, 折线为进化状态估计模型. 取 $\boldsymbol{x}'^g_1$, $\boldsymbol{x}'^g_2$ 和 $\boldsymbol{x}'^g_k$ 为采样个体, 图 2 为基于采样个体建立的下界估计模型, 对于种群个体 $\boldsymbol{x}^g_i$ 而言, 取其最接近目标函数 $f(\boldsymbol{x})$ 的下界估计值, 即 $h_1(\boldsymbol{x}^g_i)$, $h_2(\boldsymbol{x}^g_i)$ 和 $h_k(\boldsymbol{x}^g_i)$ 中的最大值为该个体的估计值 u^g_i. 如图 3 所示, 进一步, 建立目标函数 $f(\boldsymbol{x})$ 的进化状态估计模型 $H(\boldsymbol{x})$:

$$H(\boldsymbol{x}^g_i) = \max_{k\le K} h_k(\boldsymbol{x}^g_i) = \max_{k\le K}\min_{j\in I} M(l^g_{k,j} + x^g_{i,j}) \tag{6}$$

其中, $j \in I = \{1,\cdots,N+1\}, k = 1,\cdots,K, K < NP$.

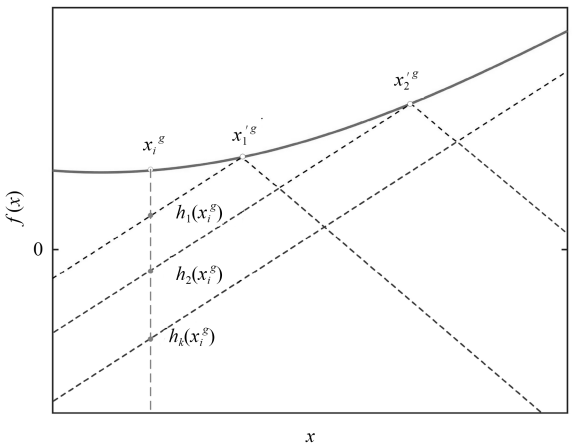


图 2 下界估计模型 $h_k(\boldsymbol{x})$

Fig. 2 Underestimation model $h_k(\boldsymbol{x})$

引理 1. 设 $\boldsymbol{x}'^g_k, k = 1,\cdots,K, K<NP$ 为当前种群 $\{\boldsymbol{x}^g_1,\cdots,\boldsymbol{x}^g_i,\cdots,\boldsymbol{x}^g_{NP}\}$ 中的采样个体, 则函数 $f(\boldsymbol{x})$ 的进化状态估计模型 $H(\boldsymbol{x})$ 满足下述性质:

$$\begin{aligned} &H(\boldsymbol{x}^g_i) \le f(\boldsymbol{x}^g_i), \forall \boldsymbol{x}^g_i \in D \\ &H(\boldsymbol{x}'^g_k) = f(\boldsymbol{x}'^g_k), \forall \boldsymbol{x}'^g_k \in D \end{aligned} \tag{7}$$

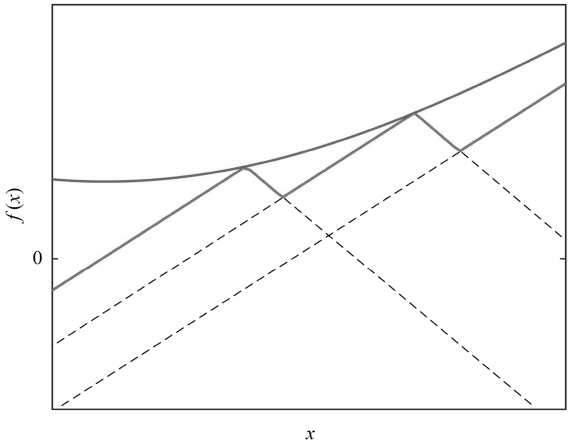


图 3 进化状态估计模型 $H(\boldsymbol{x})$

Fig. 3 Estimation model of evolution state $H(\boldsymbol{x})$

详细证明见作者之前的研究工作[21]. 引理 1 表明, 进化状态估计模型总是位于目标函数曲面下方, 二者存在误差, 且采样个体处的估计值等于其目标函数值. 由该性质可知, $H(\boldsymbol{x}^g_i) = \max_{k\le K} h_k(\boldsymbol{x}^g_i) \le f(\boldsymbol{x}^g_i)$, 从理论上来讲, 若采样个体数 $K \to \infty$, 则可以得到目标函收紧的下界. 以一维 Rastrigin (Rg) 问题作为实例进行说明, 图 4 表示无偏采样 $K = 50, 100, 200, 400$ 时, 进化状态估计模型的逼近结果. 从图 4(a) ~ (d) 所示结果来看, 采样个体数越多, 进化状态估计模型的精度越高, 至 $K = 400$, 已基本逼近原目标.

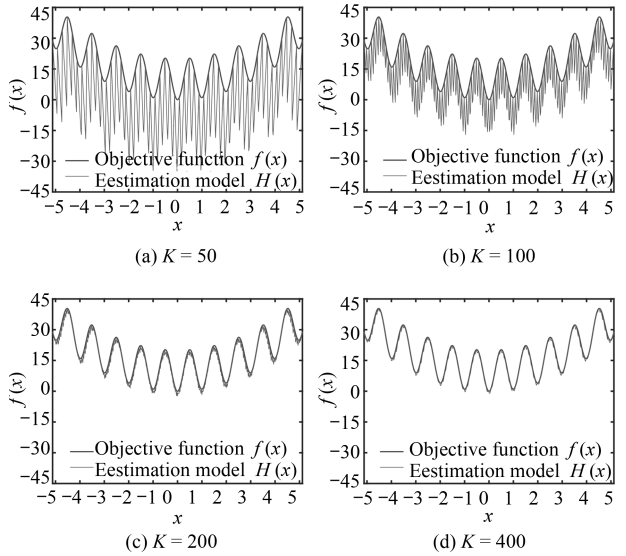


图 4 不同 K 值下的进化状态估计模型

Fig. 4 Model of evolution state under different K

本文所提算法是基于 DE 算法框架的, 而 DE 算法属于元启发式算法, 是一个有偏随机采样过程, 以种群个体作为采样个体建立进化状态估计模型时, 同样以一维 Rastrigin (Rg) 问题作为实例进行说明, 如图 5 所示, 某区域生成的个体越多, 则该区域中进化状态估计模型的精度越高, 估计误差越小.

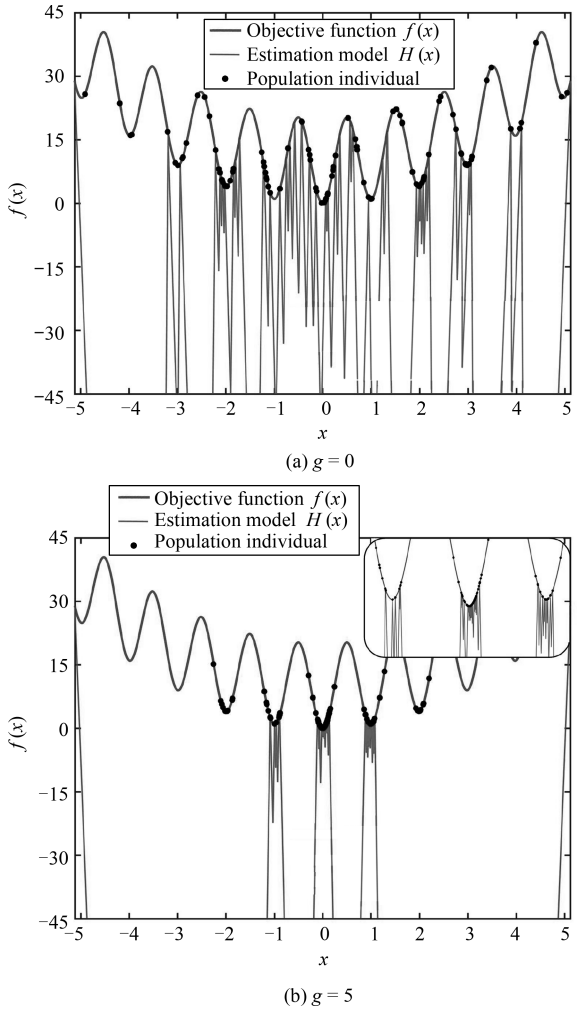


图 5　不同 g 值下 Rg 的进化状态估计模型

Fig. 5　Rg model of evolution state under different g

图 5 中, 种群个体为 100, 采样个体取 50, 黑色曲线表示目标函数 Rastrigin, 灰色线表示进化状态估计模型, 黑色圆点表示种群个体, 图 5 (a)、(b) 分别为算法迭代至 $g = 0$、5 时, 种群的分布情况与进化状态估计模型的变化. 由于采样个体是取种群中适应值最好的前 K 个个体, 图 5 (a) 所示, $g = 0$ 时种群较为分散, 进化状态估计模型在采样个体密集的区域精确度高, 其他区域的逼近程度不够, 此时第 g 代的平均估计误差较大; $g = 5$ 时, 在图 5 (b) 所示情况下, 采样个体聚集于某些局部陷阱区域, 则进化状态估计模型在该区域基本逼近原目标, 若此时得到的平均估计误差值较小, 表明种群已聚集于当前所能探测到的最好解附近. 因此, 估计误差与采样个体间的密度有关, 并且基于该性质能够通过估计误差值的变化对个体所处阶段进行判定.

1.2　状态评价因子

进化状态估计模型的估计误差能够表征当前种群个体的进化状态, 即估计误差较大, 表明当前种群较为分散, 则种群个体可能处于全局探测阶段, 以进行大范围的寻优; 而估计误差较小, 表明当前种群聚集于局部陷阱区域, 进而可以推断出大部分个体已处于局部增强阶段, 从而可以对此种群密集区进行最优解搜索. 因此, 基于估计误差设计状态评价因子, 通过该量化指标衡量进化状态, 有助于实现种群个体所处阶段的动态判定.

首先, 对第 g 代种群计算进化状态估计模型得到估计值向量 $\boldsymbol{u}_g = [u_1^g, u_2^g, \cdots, u_{NP}^g]^{\mathrm{T}}$, 其中 $u_i^g = H(\boldsymbol{x}_i^g)$, 并计算适应值向量 $\boldsymbol{f}_g = [f_1^g, f_2^g, \cdots, f_{NP}^g]^{\mathrm{T}}$;

然后, 基于两者 1-范数距离计算当前种群的平均估计误差 $\bar{E}^g$:

$$\bar{E}^g = \frac{1}{NP}\sum_{i=1}^{NP-K} |f_i^g - u_i^g|, \quad i \neq k \tag{8}$$

其中, $i = 1, \cdots, NP, k = 1, \cdots, K, K < NP$, f_i^g 和 u_i^g 分别为 $\boldsymbol{x}_i^g$ 的适应值和估计值.

最后, 根据平均估计误差的最大值 $\bar{E}_{\max}$ 和最小值 $\bar{E}_{\min}$ 对当前的平均估计误差 $\bar{E}^g$ 进行归一化处理, 则得到状态评价因子 J:

$$J = \frac{\bar{E}^g - \bar{E}_{\min}}{\bar{E}_{\max} - \bar{E}_{\min}} \tag{9}$$

其中, $\bar{E}_{\max}$ 为最大平均估计误差, 种群初始化时分布于整个搜索空间, 将其平均估计误差记为平均估计误差的最大值, 并且随着算法的迭代而不断更新. 由于种群最终收敛于全局或局部最优解, 此时, 平均估计误差会趋近于 0, 即 $\bar{E}_{\min} \approx 0$.

如图 6 所示, 以 Lm2 测试函数为例进行说明. 对平均估计误差 $\bar{E}^g$ 进行归一化处理得到的状态评价因子 J, 其值随着搜索过程中不同阶段的切换而发生变化. 以值 1 为起点, 图 6 中 J 值曲线于算法初期逐渐下降, 表明大部分种群个体经全局寻优定位至某一区域进行局部搜索, 在函数评价次数 (FEs) 为 2 000 左右时, 个体探测到更好的可行区域, J 值曲线呈上升趋势, 即表明跳出了之前的局部极值区域, 并进入局部增强阶段, 图 6 中表现为 J 值持续减小直至不再变化, 表明种群个体收敛至全局最优解附近.

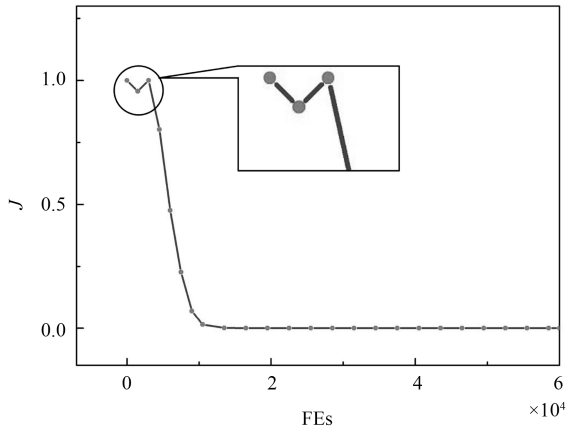


图 6 Lm2 状态评价因子变化曲线

Fig. 6 The curve graph of Lm2 state judgment factor

1.3 阶段判定

状态评价因子 J 反映了搜索过程中种群个体进化状态的变化情况, 基于状态评价因子 J 设计阶段判别模型 Ψ:

$$\Psi=\begin{cases}S_1, & rand(0,1)<J\\ S_2, & \text{其他}\end{cases} \tag{10}$$

S_1 表示全局探测阶段, S_2 表示局部增强阶段. $rand(0,1)$ 表示 0 和 1 之间的随机数.

采用式 (10) 确定当前种群个体所处于的阶段, 主要有以下几个方面的原因:

1) 状态评价因子 J 包含了当前种群进化状态的信息, 如第 1.2 节所述, 随着不同阶段的切换, 种群个体的进化状态会发生相应的变化, 即处于全局探测阶段时, 种群个体分布于整个搜索空间进行寻优, 当部分个体聚集以定位某一区域搜索最优解时, 另一部分个体会继续探测可行区域, 因此不同阶段在同一代种群中可能是共存的.

2) 当前种群个体若处于全局探测阶段, 则种群较为分散, 对应的状态评价因子 J 的值越大, 以 $rand(0,1)$ 获取的随机数有更大的概率小于 J, 因此, 采用全局探测能力较强的变异策略 (11) 的个体越多; 而确定种群个体处于局部增强阶段时, 种群聚集导致状态评价因子 J 的值逐渐变小, 以 $rand(0,1)$ 获取的随机数有更大的概率大于 J, 从而采用所设计的策略 (12) 进行局部增强的个体越多.

3) 实际上, 当种群聚集时也可能是大部分个体定位到了某局优解所在区域, 在这种情况下状态评价因子 J 的值同样会减小, 采用随机数判定方式确定个体选用的策略, 能够缓解因判定错误而导致算法早熟的情况.

针对每个种群个体, 当 $rand(0,1)<J$ 时, 即表明当前个体处于全局探测阶段 S_1, 此阶段应保持种群的多样性, 从而保证算法对整个搜索域进行充分探测. 因此, 采用策略 DE/rand/1 进行变异操作:

$$\boldsymbol{v}_i^g=\boldsymbol{x}_{r_1}^g+F\cdot(\boldsymbol{x}_{r_2}^g-\boldsymbol{x}_{r_3}^g) \tag{11}$$

其中, $r_1\neq r_2\neq r_3\neq i$, $F\in[0,1]$ 为步长因子, $\boldsymbol{v}_i^g$ 为当前种群个体经变异操作生成的变异个体.

确定当前个体处于局部增强阶段 S_2 时, 为保证算法对已探测到的区域进行局部增强操作, 加快收敛速度, 同时又要防止算法早熟收敛而陷入局部最优, 因此设计 DE/Krand/1 策略进行变异操作:

$$\boldsymbol{v}_i^g=\boldsymbol{x}_i^g+F\cdot(\boldsymbol{x}_{r_1}^g-\boldsymbol{x}_{r_2}^g)+F\cdot(\boldsymbol{x}_{Krand}^g-\boldsymbol{x}_i^g) \tag{12}$$

其中, $\boldsymbol{x}_i^g$ 为当前目标个体, $\boldsymbol{x}_{Krand}^g$ 为 K 个采样个体中随机选择的个体.

变异操作之后, 进行如式 (13) 所示的交叉操作生成试验个体 $\boldsymbol{t}^g$:

$$\boldsymbol{t}_{i,j}^g=\begin{cases}\boldsymbol{v}_{i,j}^g, & rand(j)\leq CR \text{ 或 } j=jrand\\ \boldsymbol{x}_{i,j}^g, & \text{其他}\end{cases} \tag{13}$$

其中, $j=1,\cdots,N$, $rand(j)$ 表示 j 服从 [0, 1] 的均匀分布, $jrand$ 为 1 到 N 之间随机选取的整数, $CR\in(0,1)$ 为交叉概率.

最后, 进行如式 (14) 所示的选择操作生成新一代种群:

$$\boldsymbol{x}_i^{g+1}=\begin{cases}\boldsymbol{t}_i^g, & f(\boldsymbol{t}_i^g)\leq f(\boldsymbol{x}_i^g)\\ \boldsymbol{x}_i^g, & \text{其他}\end{cases} \tag{14}$$

其中, $f(\boldsymbol{t}_i^g)$ 和 $f(\boldsymbol{x}_i^g)$ 分别为的 $\boldsymbol{t}_i^g$ 和 $\boldsymbol{x}_i^g$ 的适应值.

1.4 算法描述

整体算法流程描述如下:

步骤 1. 初始化: 对控制参数初始化, 即设置种群规模 NP, 步长因子 F, 交叉率 CR 和常数 M, 随机生成初始种群.

步骤 2. 获取种群个体的估计值.

步骤 2.1. 筛选个体: 依据个体适应值对当前种群所有个体进行升序排列, 选取前 K 个个体作为采样个体.

步骤 2.2. 构建进化状态估计模型: 基于步骤 2.1 选取的采样个体, 根据式 (5) 建立支撑向量, 进而建立进化状态估计模型.

步骤 2.3. 获取估计值: 计算进化状态估计模型, 获取当前种群所有个体的估计值.

步骤 3. 建立状态评价因子.

步骤 3.1. 获取种群个体的估计值与适应值, 根据式 (8) 计算当前种群的平均估计误差 $\bar{E}^g$.

步骤 3.2. 根据式 (9) 计算当前代的状态评价因子 J.

步骤 4. 判断个体所处阶段: 根据式 (10) 确定当前种群中个体所处的阶段, 进而采用相适应的变异策略.

步骤 4.1. 若处于全局探测阶段, 则采用式 (11) 变异策略进行变异操作.

步骤 4.2. 若处于局部增强阶段, 则采用式 (12) 变异策略进行变异操作.

步骤 5. 种群更新: 对步骤 4 中生成的变异个体进行交叉和选择操作, 生成新一代种群个体.

步骤 5.1. 采用式 (13) 进行交叉操作;

步骤 5.2. 采用式 (14) 进行选择操作;

步骤 6. 判断是否满足终止条件, 若是, 则保存最优个体并退出算法, 否则, 转至步骤 2.

1.5 复杂度分析

步骤 3 中, 首先步骤 3.1 对种群个体进行排名的时间复杂度为 $\mathrm{O}(NP\cdot\log_2 NP)$, 其次步骤 3.2 中建立进化状态估计模型的时间复杂度为$\mathrm{O}(N+1)$, 最后步骤 3.3 中计算种群所有个体的下界估计信息时, 采用 n 叉树数据结构来保存估计值, 并且在更新操作完成以后删除树, 从 n 叉树中查询相应叶子节点的时间复杂度为 $\mathrm{O}(NP\cdot\log_{N+1}T)$, 其中, T ($T\le NP^2$) 为 n 叉树的节点数, 所以, 步骤 3 获取下界估计值的时间复杂度为 $\mathrm{O}(NP\cdot\log_2 NP)$; 另外, 步骤 4 中计算平均估计误差 $\bar{E}_g$ 的时间复杂度为 $\mathrm{O}(NP)$, 计算状态评价因子 J 的时间复杂度为 $\mathrm{O}(NP)$; 步骤 5 中判定个体所处阶段后采用阶段特定的变异策略更新种群, 其时间复杂度为 $\mathrm{O}(NP)$. 因此, SEFDE 算法相对于基本 DE 算法所增加的时间复杂度为 $\mathrm{O}(NP\cdot\log_2 NP)$ 或 $\mathrm{O}(N+1)$, 并没有明显增加 DE 算法的时间复杂度.

2 数值实验

2.1 参数选择分析

为了合理评估本文所提算法 SEFDE 的优化性能, 选取了 20 个典型的标准测试问题[22] 以保证实验结果的客观性, 其中包含 8 个高维单模态问题 (f_1-f_8) 和 12 个高维多模态问题 (f_9-f_{20}), 表 1 列出了各测试函数的数学表达式及对应的参数. 根据文献 [23−24] 中对参数设置的建议, 在 SEFDE 算法中, 设置种群规模 $NP=50$, 步长因子 $F=0.5$, 交叉概率 $CR=0.5$, 另外, SEFDE 算法需要对当前种群筛选 K 个采样个体以建立进化状态估计模型, 才能建立状态评价因子对搜索过程中的不同阶段进行判定. 因此, 在进行对比实验之前, 需要确定引入的新参数, 即参数 K 和 M 的具体设置.

对于采样个体数 K, 取 $K=2,3,4,5,6$ 进行参数分析实验. 在具体实验中, 记录各函数的平均函数评价次数 (FEs) 和成功率 (SR), 其中, 规定所求得的最优值 $f_{\min}(\boldsymbol{x})$ 达到一定精度 (即 $|f_{\min}(\boldsymbol{x})-optimum|\le 10^{-5}$, $optimum$ 为对应函数的全局最优值) 时为成功. 表 2 给出了 30 维测试函数, 30 次独立运行的平均函数评价次数和成功率. 在参数 K 的不同设置下, SEFDE 算法对各测试函数的求解结果均在同一数量级, 并且 SEFDE 算法对所有函数都能成功求解, 所需函数评价次数较为相近. 但考虑到: 1) 采样个体过少, 不具有代表性, 无法描述种群的收敛情况; 2) 采样个体过多, 需要构建的支撑向量越多, 则算法的空间复杂度也会随之变大. 因此选择表 2 中数据结果较好的参数设置, 即 $K=5$.

对于问题特定的常数 M, 在种群较为分散的算法初期, M 值的选择对进化状态估计模型的建立有一定影响, M 值过小, 则出现高估现象, M 值过大, 则导致算法收敛缓慢; 算法后期, 由于进化状态估计模型已基本逼近原目标, 求得的下界估计值与实际值之间相差很小, 此时 M 值对算法性能的影响较小. 本文采用预处理方式进行启发式设置, 将 M 的初始值设置为足够大的正数, 然后逐渐减小其值, 逐次计算当前 M 值下得到的种群个体估计值, 依据平均估计误差进行当前 M 值的合理性判断, 具体设置可参见作者之前的工作[25].

2.2 与 State-of-the-art 算法比较

实验选用 DELU[26]、SHADE[27]、EPSDE[28]、CoDE[29] 这 4 种 State-of-the-art 算法, 分别从计算代价和可靠性、求解精度、收敛性能等几个方面与 SEFDE 进行比较. 对比算法均采用各自原文献中的参数设置, 各算法对各测试函数均独立运行 30 次.

Zhou 等[26] 提出基于抽象凸估计策略的差分进化算法 (DELU), 通过对试验个体的邻近个体构建支撑超平面以获取试验个体的下界估计值, 进而设计抽象凸估计策略减小计算代价; Tanabe 等[27] 提出的自适应差分进化算法 (SHADE), 利用进化过程知识实现参数的自适应调整; Mallipeddi 等[28] 提出具系综变异策略及参数的差分进化算法 (EPSDE), 保留产生优质试验个体的变异策略和参数, 对产生较差试验个体的策略和参数以相同的概率保留或丢弃; Wang 等[29] 提出了变异策略和参数混合的差分进化算法 (CoDE), 通过竞争方式完成对不同策略与参数随机组合的选择, 以确定每一代种群采用的变异策略.

表 1 标准测试函数的参数

Table 1 The parameters of benchmark functions

函数名	表达式	维数 (N)	取值范围	全局最小值
Sphere	$f_1(\boldsymbol{x}) = \sum_{i=1}^{N} x_i^2$	30, 50	$(-100, 100)^N$	0
SumSquares	$f_2(\boldsymbol{x}) = \sum_{i=1}^{N} i x_i^2$	30, 50	$(-10, 10)^N$	0
Schwefel 2.22	$f_3(\boldsymbol{x}) = \sum_{i=1}^{N} \lvert x_i \rvert + \prod_{i=1}^{N} \lvert x_i \rvert$	30, 50	$(-10, 10)^N$	0
Exponential	$f_4(\boldsymbol{x}) = -\exp(-0.5 \sum_{i=1}^{N} x_i^2)$	30, 50	$(-1, 1)^N$	-1
Tablet	$f_5(\boldsymbol{x}) = 10^6 x_1^2 + \sum_{i=2}^{N} x_i^2$	30, 50	$(-100, 100)^N$	0
Step	$f_6(\boldsymbol{x}) = \sum_{i=1}^{N} \lfloor x_i + 0.5 \rfloor^2$	30, 50	$(-100, 100)^N$	0
Zakharov	$f_7(\boldsymbol{x}) = \sum_{i=1}^{N} x_i^2 + \sum_{i=1}^{N} (0.5 i x_i)^2 + \sum_{i=1}^{N} (0.5 i x_i)^4$	30, 50	$(-5, 10)^N$	0
Rosenbrock	$f_8(\boldsymbol{x}) = \sum_{i=1}^{N-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30, 50	$(-30, 30)^N$	0
Griewank	$f_9(\boldsymbol{x}) = 1 + \frac{1}{4000} \sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N} \cos(\frac{x_i}{\sqrt{i}})$	30, 50	$(-600, 600)^N$	0
Schaffer 2	$f_{10}(\boldsymbol{x}) = \sum_{i=1}^{N-1} (x_i^2 + x_{i+1}^2)^{0.25} (\sin^2(50(x_i^2 + x_{i+1}^2)^{0.1}) + 1)$	30, 50	$(-100, 100)^N$	0
Schwefel 2.26	$f_{11}(\boldsymbol{x}) = -\sum_{i=1}^{N} x_i \sin(\sqrt{\lVert x_i \rVert})$	30, 50	$(-500, 500)^N$	$-12\,569.18$
Himmelblau	$f_{12}(\boldsymbol{x}) = N^{-1} \sum_{i=1}^{N} (x_i^4 - 16 x_i^2 + 5 x_i)$	30, 50	$(-5, 5)^N$	-78.3323
Levy and Montalvo 1	$f_{13}(\boldsymbol{x}) = \frac{\pi}{N} (10 \sin^2(\pi y_1) + \sum_{i=1}^{N-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_N - 1)^2), \quad y_i = 1 + \frac{1}{4}(x_i + 1)$	30, 50	$(-10, 10)^N$	0
Levy and Montalvo 2	$f_{14}(\boldsymbol{x}) = 0.1(\sin^2(3\pi x_i) + \sum_{i=1}^{N-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_N - 1)^2 [1 + \sin^2(2\pi x_N)])$	30, 50	$(-5, 5)^N$	0
Ackley	$f_{15}(\boldsymbol{x}) = -20 \exp(-0.02 \sqrt{N^{-1} \sum_{i=1}^{N} x_i^2}) - \exp(N^{-1} \sum_{i=1}^{N} \cos(2\pi x_i)) + 20 + e$	30, 50	$(-30, 30)^N$	0
Rastrigin	$f_{16}(\boldsymbol{x}) = 10N + \sum_{i=1}^{N} (x_i^2 - 10 \cos(2\pi x_i))$	30, 50	$(-5, 5)^N$	0
Penalized 1	$f_{17}(x) = \frac{\pi}{N} \{\sum_{i=1}^{N-1} (y_i - 1)^2 [1 + \sin(\pi y_{i+1})] + (y_N - 1)^2 + (10 \sin^2(\pi y_1))\} + \sum_{i=1}^{N} u(x_i, 10, 100, 4), \quad y_i = 1 + \frac{x_i + 1}{4}$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	30, 50	$(-50, 50)^N$	0
Penalized 2	$f_{18}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{N-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_N - 1)^2 [1 + \sin^2(2\pi x_N)]\} + \sum_{i=1}^{N} u(x_i, 5, 100, 4)$	30, 50	$-(50, 50)^N$	0
Neumaier	$f_{19}(\boldsymbol{x}) = \sum_{i=1}^{N} (x_i - 1)^2 - \sum_{i=2}^{N} x_i x_{i-1} + \frac{N(N+4)(N-1)}{6}$	30, 50	$(-900, 900)^N$	0
Alpine	$f_{20}(\boldsymbol{x}) = \sum_{i=1}^{N} \lvert x_i \sin x_i + 0.1 x_i \rvert$	30, 50	$(-10, 10)^N$	0

表 2 SEFDE 中参数 K 设置下的平均函数评价次数和成功率

Table 2 Average numbers of function evaluations and success rates of parameter settings K in SEFDE

Fun	N	$K=2$		$K=3$		$K=4$		$K=5$		$K=6$	
		FEs	SR	FEs	SR	FEs	SR	FEs	SR	FEs	SR
f_1	30	1.51E+04	1.00	1.19E+04	1.00	1.26E+04	1.00	1.21E+04	1.00	1.18E+04	1.00
f_2	30	1.17E+04	1.00	1.14E+04	1.00	1.19E+04	1.00	1.06E+04	1.00	1.14E+04	1.00
f_3	30	1.85E+04	1.00	1.76E+04	1.00	1.73E+04	1.00	1.75E+04	1.00	1.83E+04	1.00
f_4	30	7.88E+03	1.00	7.89E+03	1.00	7.85E+03	1.00	7.83E+03	1.00	7.87E+03	1.00
f_5	30	2.94E+04	1.00	2.97E+04	1.00	2.93E+04	1.00	2.91E+04	1.00	2.90E+04	1.00
f_6	30	8.94E+03	1.00	8.61E+03	1.00	8.56E+03	1.00	8.28E+03	1.00	8.45E+03	1.00
f_7	30	1.46E+05	1.00	1.45E+05	1.00	1.46E+05	1.00	1.46E+05	1.00	1.44E+05	1.00
f_8	30	1.24E+05	1.00	1.20E+05	1.00	1.21E+05	1.00	1.22E+05	1.00	1.21E+05	1.00
f_9	30	1.28E+04	1.00	1.26E+04	1.00	1.25E+04	1.00	1.27E+04	1.00	1.24E+04	1.00
f_{10}	30	1.00E+05	1.00	1.09E+05	1.00	1.10E+05	1.00	1.07E+05	1.00	1.09E+05	1.00
f_{11}	30	4.43E+04	1.00	4.34E+04	1.00	4.57E+04	1.00	4.40E+04	1.00	4.48E+04	1.00
f_{12}	30	1.24E+04	1.00	1.71E+04	1.00	1.74E+04	1.00	1.64E+04	1.00	1.70E+04	1.00
f_{13}	30	1.26E+04	1.00	1.24E+04	1.00	1.26E+04	1.00	1.27E+04	1.00	1.26E+04	1.00
f_{14}	30	1.18E+04	1.00	1.19E+04	1.00	1.20E+04	1.00	1.22E+04	1.00	1.19E+04	1.00
f_{15}	30	2.02E+04	1.00	1.95E+04	1.00	2.01E+04	1.00	2.02E+04	1.00	1.98E+04	1.00
f_{16}	30	5.60E+04	1.00	5.44E+04	1.00	5.62E+04	1.00	5.34E+04	1.00	5.66E+04	1.00
f_{17}	30	1.90E+04	1.00	1.96E+04	1.00	1.91E+04	1.00	1.92E+04	1.00	1.98E+04	1.00
f_{18}	30	2.32E+04	1.00	2.29E+04	1.00	2.32E+04	1.00	2.30E+04	1.00	2.29E+04	1.00
f_{19}	30	1.72E+05	1.00	1.70E+05	1.00	1.67E+05	1.00	1.70E+05	1.00	1.68E+05	1.00
f_{20}	30	3.62E+04	1.00	3.75E+04	1.00	3.13E+04	1.00	3.48E+04	1.00	3.47E+04	1.00
AVE		4.41E+04	1.000	4.42E+04	1.000	4.41E+04	1.000	4.40E+04	1.000	4.41E+04	1.000

表 3 函数评价次数和成功率对比数据

Table 3 Compared data on function evaluations and success rates

Fun	N	DELU		SHADE		EPSDE		CoDE		SEFDE	
		FEs	SR	FEs	SR	FEs	SR	FEs	SR	FEs	SR
f_1	30	1.22E+04	1.00	2.35E+04	1.00	1.67E+04	1.00	4.02E+04	1.00	**1.21E+04**	1.00
f_2	30	1.07E+04	1.00	2.16E+04	1.00	1.51E+04	1.00	3.63E+04	1.00	**1.06E+04**	1.00
f_3	30	3.15E+04	1.00	3.42E+04	1.00	2.32E+04	1.00	5.64E+04	1.00	**1.75E+04**	1.00
f_4	30	7.98E+03	1.00	1.67E+04	1.00	9.36E+03	1.00	2.24E+04	1.00	**7.83E+03**	1.00
f_5	30	2.55E+04	1.00	2.67E+04	1.00	**1.83E+04**	1.00	4.13E+04	1.00	2.91E+04	1.00
f_6	30	1.16E+04	1.00	1.19E+04	1.00	**8.33E+03**	1.00	2.01E+04	1.00	**8.28E+03**	1.00
f_7	30	7.71E+04	1.00	**5.98E+04**	1.00	1.33E+05	1.00	7.80E+04	1.00	1.46E+05	1.00
f_8	30	**7.09E+04**	1.00	1.09E+05	1.00	1.16E+05	0.87	2.37E+05	1.00	1.22E+05	1.00
f_9	30	1.54E+04	1.00	2.56E+04	1.00	1.76E+04	0.83	4.39E+04	1.00	**1.27E+04**	1.00
f_{10}	30	**9.78E+04**	1.00	1.08E+05	0.97	1.13E+05	1.00	1.73E+05	1.00	1.07E+05	1.00
f_{11}	30	**1.34E+04**	1.00	9.67E+04	1.00	3.71E+04	1.00	6.00E+04	1.00	4.40E+04	1.00
f_{12}	30	1.98E+04	1.00	3.35E+04	0.97	2.33E+04	1.00	3.56E+04	1.00	**1.64E+04**	1.00
f_{13}	30	**1.05E+04**	1.00	1.67E+04	1.00	1.30E+04	1.00	2.63E+04	1.00	1.27E+04	1.00
f_{14}	30	**8.71E+03**	1.00	1.71E+04	1.00	1.15E+04	1.00	2.70E+04	1.00	1.22E+04	1.00
f_{15}	30	2.65E+04	1.00	3.21E+04	1.00	2.32E+04	1.00	5.68E+04	1.00	**2.02E+04**	1.00
f_{16}	30	5.94E+04	1.00	1.24E+05	1.00	6.40E+04	1.00	1.30E+05	0.97	**5.34E+04**	1.00
f_{17}	30	**9.17E+03**	1.00	1.92E+04	1.00	1.29E+04	1.00	3.03E+04	1.00	1.92E+04	1.00
f_{18}	30	1.52E+04	1.00	**1.50E+04**	1.00	1.59E+04	0.97	3.54E+04	1.00	2.30E+04	1.00
f_{19}	30	1.68E+05	0.97	**1.09E+05**	1.00	NA	0.00	2.69E+05	1.00	1.70E+05	1.00
f_{20}	30	NA	0.00	NA	0.00	1.00E+05	1.00	NA	0.00	**3.48E+04**	1.00
AVE		4.96E+04	0.95	6.00E+04	0.947	5.36E+04	0.934	8.59E+04	0.949	**4.40E+04**	**1.000**

2.2.1 计算代价和可靠性

首先, 采用性能指标函数评价次数 (FEs) 和成功率 (SR) 来评价各算法的计算代价和可靠性. 即在给定最大函数评价次数 (MaxFEs) 内, 各算法达到给定函数误差 ($|f_{\min}(\boldsymbol{x}) - optimum|$) 的精度 θ 时, 所需 FEs 以及达到的 SR. 具体实验过程中, 设置 $\theta = 10^{-5}$, MaxFEs = 300 000, 该实验中取各测试函数的维数 $N = 30$, 30 次独立运行的 FEs 和 SR 结果见表 3, 最优结果加粗表示, 其中在 MaxFEs 内仍未能成功求解的记为 "NA".

对比表 3 所列数据, 所提 SEFDE 建立了状态评价因子, 从而能够根据当前种群个体所处阶段合理分配变异策略, 加快算法搜索进程, 因此 20 个函数问题中其函数评价次数取得了 10 个最优, 并且对所有函数均能成功求解; 由于将下界估计值引入选择环节, DELU 有效减小了计算代价, 故对 6 个测试函数 f_8, f_{10}, f_{11}, f_{13}, f_{14}, f_{17} 的优化相对于其他算法取得了最优; SHADE 取得了 3 个最优; EPSDE 有 2 个最优; CoDE 均不及其他算法. 但在成功率方面, 除了 SEFDE 算法能够对所有函数成功求解外, 其他算法均存在一些无法以 100 % 的成功率求解测试函数, 甚至出现陷入局部最优的情况. 综合 20 个函数问题的平均结果, 所提 SEFDE 算法在计算代价及可靠性方面的性能最好, 表明通过判定个体所处阶段并切换变异策略, 能够有效提高算法搜索效率并实现了计算代价的减小.

2.2.2 求解精度

其次, 采用性能指标函数误差的平均值 (Mean) 和标准差 (Std) 来评价各算法的求解精度. 即在给定 $2\,000N$ 的函数评价次数内, 5 种算法对 20 个 30 维的典型标准测试函数各自独立优化 30 次后, 计算 Mean 和 Std 来量化优化结果的质量. 为了清楚地表现 SEFDE 的优劣, 采用 Wilcoxon Signed Rank Test[30] 对优化结果进行非参数假设检验, 显著性水平为 0.05, 结果见表 4, 最优结果加粗表示, 各结果的上标 "+"、"−" 和 "≈" 分别表示本文算法显著优于所比算法、显著差于所比算法、与所比算法无明显差异.

对比表 4 所列数据, 总体而言, 所提 SEFDE 算法在全部 20 个测试函数有 14 个函数的结果优于其他 4 种算法, 表明 SEFDE 算法根据个体所处于的不同阶段分配特定阶段变异策略, 有助于算法性能

表 4 DELU、SHADE、EPSDE、CoDE、SEFDE 对 30 维测试函数的优化结果, 平均值 (标准差)

Table 4 Compared data of 30 D optimization results on DELU、SHADE、EPSDE、CoDE、SEFDE, Mean (Std)

Fun	N	DELU	SHADE	EPSDE	CoDE	SEFDE
		Mean error (Std Dev)	Mean error (Std Dev)	Mean error (Std Dev)	Mean error (Std Dev)	Mean error (Std Dev)
f_1	30	9.25E−19(2.99E−36)$^+$	2.87E−19(2.31E−19)$^+$	3.87E−31(9.46E−31)$^+$	2.16E−10(1.73E−10)$^+$	**4.01E−38(5.05E−38)**
f_2	30	2.56E−32(4.00E−32)$^+$	4.40E−20(3.25E−20)$^+$	5.52E−31(1.94E−30)$^+$	2.83E−11(2.61E−11)$^+$	**5.83E−42(9.48E−42)**
f_3	30	1.46E−11(1.23E−11)$^+$	4.98E−10(2.39E−10)$^+$	1.29E−16(2.15E−16)$^+$	3.86E−06(1.32E−06)$^+$	**2.03E−23(1.62E−23)**
f_4	30	**0.00E+00(0.00E+00)**$^\approx$	4.07E−17(5.44E−17)$^+$	**0.00E+00(0.00E+00)**$^\approx$	1.30E−14(1.13E−14)$^+$	**0.00E+00(0.00E+00)**
f_5	30	1.65E−20(4.70E−20)$^-$	1.97E−18(2.27E−18)$^-$	**8.85E−30(1.75E−29)**$^-$	4.04E−10(3.04E−10)$^+$	1.06E−15(3.87E−16)
f_6	30	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**
f_7	30	**1.82E−04(3.39E−04)**$^-$	3.57E−04(1.00E−03)$^-$	1.42E+01(2.25E+01)$^-$	5.69E−04(7.53E−04)$^-$	3.59E+01(1.12E+01)
f_8	30	**1.28E−04(1.36E−04)**$^-$	1.81E+01(1.11E+00)$^-$	8.61E+00(2.09E+00)$^-$	1.97E+01(5.80E−01)$^-$	2.54E+01(8.26E−01)
f_9	30	**0.00E+00(0.00E+00)**$^\approx$	2.47E−04(1.35E−03)$^+$	6.57E−04(2.50E−03)$^+$	2.47E−07(7.34E−07)$^+$	**0.00E+00(0.00E+00)**
f_{10}	30	**2.03E−02(6.76E−03)**$^-$	3.56E−01(5.96E−02)$^+$	2.35E−01(1.31E−01)$^+$	1.97E+00(4.24E−01)$^+$	1.84E−01(7.45E−02)
f_{11}	30	**0.00E+00(0.00E+00)**$^\approx$	1.75E+02(5.85E+01)$^+$	**0.00E+00(0.00E+00)**$^\approx$	1.66E−02(3.13E−02)$^+$	**0.00E+00(0.00E+00)**
f_{12}	30	**0.00E+00(0.00E+00)**$^\approx$	3.14E−05(1.02E−07)$^+$	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**
f_{13}	30	2.19E−30(2.11E−30)$^+$	5.53E−21(5.00E−21)$^+$	2.91E−29(1.14E−28)$^+$	1.80E−13(3.36E−13)$^+$	**1.57E−32(0.00E+00)**
f_{14}	30	1.36E−32(3.13E−34)$^+$	4.07E−21(6.03E−21)$^+$	1.75E−32(9.04E−33)$^+$	1.31E−13(1.51E−13)$^+$	**1.35E−32(0.00E+00)**
f_{15}	30	2.42E−10(1.35E−10)$^+$	1.19E−10(6.88E−11)$^+$	6.04E−15(1.66E−15)$^+$	3.75E−06(1.88E−06)$^+$	**4.00E−15(1.02E−15)**
f_{16}	30	1.99E−06(2.50E−06)$^+$	2.30E+01(2.38E+00)$^+$	5.48E−02(1.39E−01)$^+$	3.30E+01(5.81E+00)$^+$	**5.18E−07(1.12E−06)**
f_{17}	30	4.57E−26(7.30E−26)$^+$	2.85E−20(4.26E−20)$^+$	**6.02E−32(1.48E−31)**$^-$	1.44E−12(1.26E−12)$^+$	2.09E−26(1.95E−26)
f_{18}	30	3.80E−21(4.03E−21)$^+$	3.51E−19(3.01E−19)$^+$	3.66E−04(2.01E−03)$^+$	2.45E−11(2.36E−11)$^+$	**2.48E−21(4.41E−21)**
f_{19}	30	**3.67E+02(4.24E+02)**$^-$	6.53E+02(6.50E+02)$^+$	9.64E+02(4.81E+02)$^+$	6.92E+02(8.00E+02)$^+$	6.23E+02(5.05E+02)
f_{20}	30	2.20E−01(1.67E−05)$^+$	1.60E−02(1.92E−03)$^+$	1.38E−03(1.06E−03)$^+$	1.75E+00(1.47E+00)$^+$	**2.07E−13(2.86E−13)**
$+/\approx/-$		10/5/3	16/1/3	12/4/4	16/2/2	−/−/−

表 5　DELU、SHADE、EPSDE、CoDE、SEFDE 对 50 维测试函数的优化结果, 平均值 (标准差)

Table 5　Compared data of 50 D optimization results on DELU、SHADE、EPSDE、CoDE、SEFDE, Mean (Std)

Fun	N	DELU	SHADE	EPSDE	CoDE	SEFDE
		Mean error (Std Dev)	Mean error (Std Dev)	Mean error (Std Dev)	Mean error (Std Dev)	Mean error (Std Dev)
f_1	50	4.24E−42(4.13E−42)$^+$	2.05E−53(3.34E−53)$^+$	2.52E−50(7.95E−50)$^+$	4.94E−21(5.74E−21)$^+$	**2.30E−57(2.91E−57)**
f_2	50	1.92E−32(4.01E−32)$^+$	**1.35E−53(3.43E−53)**$^-$	1.80E−51(3.75E−51)$^-$	4.46E−22(6.24E−22)$^+$	**1.59E−42(2.13E−42)**
f_3	50	7.87E−29(9.66E−29)$^+$	2.22E−27(1.67E−27)$^+$	5.63E−29(1.68E−28)$^+$	6.05E−12(2.77E−12)$^+$	**5.52E−29(4.63E−29)**
f_4	50	**0.00E+00(0.00E+00)**$^\approx$	9.99E−17(3.51E−17)$^+$	1.33E−16(4.68E−17)$^+$	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**
f_5	50	7.64E−21(8.52E−21)$^+$	**1.03E−52(1.07E−52)**$^-$	3.04E−50(7.79E−50)$^-$	1.01E−20(1.21E−20)$^+$	1.50E−27(1.66E−27)
f_6	50	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**$^\approx$	5.00E−01(7.07E−01)$^+$	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**
f_7	50	1.77E−04(9.11E−05)$^-$	**8.22E−07(1.80E−06)**$^-$	2.22E+02(6.11E+01)$^+$	2.80E−04(3.97E−04)$^-$	9.43E+01(9.34E+00)
f_8	50	**2.59E−06(1.29E−06)**$^-$	1.34E+01(1.83E+00)$^-$	2.72E+01(1.81E+01)$^-$	5.16E+01(2.64E+01)$^+$	3.73E+01(3.74E+00)
f_9	50	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**$^\approx$	2.46E−03(5.69E−03)$^+$	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**
f_{10}	50	2.28E−02(1.14E−02)$^-$	5.45E−02(3.68E−02)$^-$	**3.12E−03(4.16E−03)**$^-$	1.11E−01(7.80E−02)$^-$	2.19E−01(2.67E−01)
f_{11}	50	**0.00E+00(0.00E+00)**$^\approx$	1.19E+01(3.79E+01)$^+$	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**
f_{12}	50	**0.00E+00(0.00E+00)**$^\approx$	3.39E−01(3.95E−01)$^+$	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**
f_{13}	50	9.42E−33(1.44E−48)$^+$	9.42E−33(1.44E−48)$^+$	9.46E−33(6.47E−35)$^+$	4.76E−25(4.46E−25)$^+$	**1.44E−33(1.11E−33)**
f_{14}	50	1.35E−32(2.88E−48)$^+$	1.35E−32(2.88E−48)$^+$	1.40E−32(8.62E−34)$^+$	1.06E−24(8.10E−25)$^+$	**1.33E−32(0.00E+00)**
f_{15}	50	6.13E−15(1.95E−15)$^+$	7.11E−15(0.00E+00)$^+$	1.14E−14(4.97E−15)$^+$	8.59E−12(4.34E−12)$^+$	**1.04E−15(3.89E−15)**
f_{16}	50	2.45E−07(1.58E−07)$^+$	1.84E−01(5.31E−02)$^+$	4.42E+00(1.09E+01)$^+$	4.27E+01(7.39E+00)$^+$	**1.96E−07(1.10E−07)**
f_{17}	50	3.25E−30(1.48E−30)$^-$	**9.42E−33(1.44E−48)**$^-$	1.00E−32(1.96E−33)$^-$	9.53E−24(4.52E−25)$^+$	1.68E−26(3.74E−26)
f_{18}	50	3.08E−32(5.23E−33)$^-$	**1.36E−32(3.90E−34)**$^-$	1.10E−03(3.47E−03)$^+$	1.29E−22(1.22E−22)$^+$	1.71E−24(3.73E−24)
f_{19}	50	8.90E+03(6.05E+03)$^-$	**1.04E+03(6.24E+02)**$^-$	1.26E+04(1.65E+03)$^-$	1.09E+04(1.44E+03)$^-$	1.66E+04(2.24E+03)
f_{20}	50	2.70E−01(4.67E−02)$^+$	2.60E−01(5.16E−02)$^+$	2.80E−01(4.22E−02)$^+$	3.20E−01(4.22E−02)$^+$	**0.00E+00(0.00E+00)**
$+/\approx/-$		9/5/6	10/2/7	12/2/6	12/5/3	−/−/−

的改善, 解的质量较好. 对测试函数 f_4, f_6, f_{11}, f_{12} 的优化性能与其他算法相近, 是因为对比算法 DELU、SHADE、EPSDE、CoDE 与 SEFDE 算法均能优化得到全局最优解. 其中, 对测试函数 f_{10}, f_{17}, f_{19} 的优化结果不及 DELU, 但 SEFDE 也取得了次优解. 另外, SEFDE 对函数 f_5, f_7, f_8 优化显著较差, 主要是由于函数曲面以及算法特点所造成的, 比如, 由于 EPSDE 保存较优个体的变异策略和参数, 对于 f_3 (Schwefel2.22 问题) 这类单模函数的优化效果更加优秀; 但 EPSDE 对 f_7 (Zakharov 问题) 的优化与 SEFDE 的结果在同一数量级, 不如 DELU、SHADE、CoDE; f_8 (Rosenbrock 问题)[31], 当其维数大于 3 时则变为多模函数, 虽然 SEFDE 能够很快找到其局部最优解, 但是由于其函数曲面极其复杂并且具有欺骗性, 最终极易陷入局部最优, 从表 4 中可以看出, 除了 DELU 算法因在选择操作引入估计值使得结果相对较好外, 其他算法在实验中均无法找到满意解, 并且所有算法对 f_7, f_8 的优化均未达到 10^{-5} 的精度.

为了验证所提 SEFDE 算法对高维函数的优化性能, 分别对这 5 种算法进行 50 维典型标准测试函数的优化实验, 其他条件保持不变. 如表 5 所示, 在全部 20 个测试函数中有 12 个函数的结果优于其他 4 种算法, 表明 SEFDE 对 50 维函数的优化效果同样优越且稳定. DELU 对函数 f_7、f_8、f_{10}、f_{17}、f_{18}、f_{19} 的优化结果好于 SEFDE, 但 SEFDE 有 9 个函数的优化结果好于 DELU, 二者对其余函数均能得到全局最优解; SHADE 有 7 个函数的优化结果好于 SEFDE, 数据结果表明其对高维函数的优化性能较好, 但有 10 个函数的优化结果逊于 SEFDE; SEFDE 对函数 f_2、f_5、f_8、f_{10}、f_{17}、f_{19} 的优化结果逊于 EPSDE, 定位至最优解附近但有 12 个函数的优化结果好于 EPSDE, 对函数 f_{11}、f_{12} 的优化效果相近; SEFDE 与 CoDE 相比, 二者有 5 个函数优化效果相近, 对 12 个函数的优化好于 CoDE, 对函数 f_7、f_{10}、f_{19} 的优化较差.

2.2.3　收敛性能

最后, 采用平均收敛速度来刻画 SEFDE 算法在平均收敛速度方面的优势, 选取 4 个具有代表性的测试函数 f_1, f_9, f_{15}, f_{16} 绘制平均收敛曲线图进行分析说明. 图 8 给出了这 4 个测试函数的收敛曲线图, 图中数据是基于算法 30 次独立优化的平均结果, 横坐标均取函数评价次数, 纵坐标均取平均函数误差的对数.

结合图 7 状态评价因子变化曲线对图 8 收敛曲线进行分析. 由于 SEFDE 算法通过判断当前种群个体所处阶段来切换相适应的策略, 因此能够有效提高算法效率, 显著加快了收敛速度. 具体而言, 如图 8 (a) 所示, SEFDE 算法用以寻优的种群在前期

较为分散, 处于全局探测阶段, 故建立的进化状态估计模型误差较大, 图 7 中即表现为测试函数 f_1 的状态评价因子曲线首先呈上升趋势, 在图 8 (a) 中相应地表现为 SEFDE 对 f_1 的收敛速度在前期逊于 DELU 和 EPSDE, 而随着模型的逐渐精确, 图 7 中 f_1 的状态评价因子曲线呈下降趋势, 表明进入局部增强阶段, 此时图 8 (a) 中 SEFDE 快速收敛到全局最优解附近, 收敛曲线位于其他算法的下方, 优化效果最好; 图 8 (b) 中, SEFDE 的收敛曲线在前期除 DELU 外位于其他算法下方, 与图 7 中 f_9 状态评价因子的变化趋势一致, SEFDE 算法快速进行局部搜索, 相比于其他所有算法, 由于 SEFDE 算法采用了特定阶段的策略而迅速收敛, DELU 在选择环节引入估计值减小了计算代价, 故前期收敛较快, 但 SEFDE 算法收敛更快, 超过 DELU 快速收敛至全局最优解附近, 而 EPSDE 和 CoDE 由于 f_9 的多模性质陷入局部最优, 未能找到最优解; 图 8 (c) 的收敛情况同图 8 (b), SEFDE 的收敛速度在前期弱于 DELU, 然后快速收敛超过其他算法; 图 8 (d) 中 f_{16} 为曲面复杂的多模函数, 除 SEFDE 外的其他算法对 f_{16} 优化的速度较慢且解的质量不高, SHADE 和 CoDE 甚至陷入局部最优, 图 7 中 f_{16} 的状态评价因子曲线呈跨度较长的下降趋势, 表明 SEFDE 首先进行全局探测并维持这一过程用以全局寻优, 采

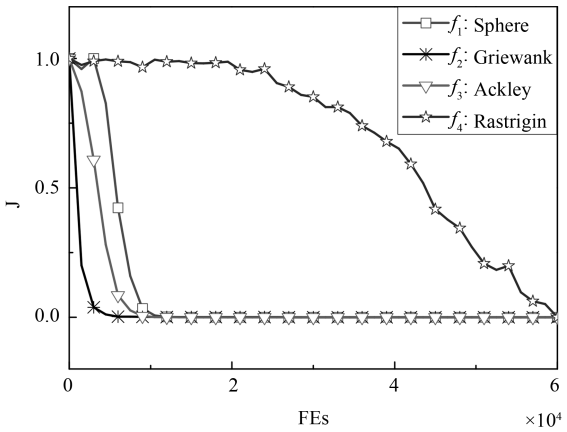
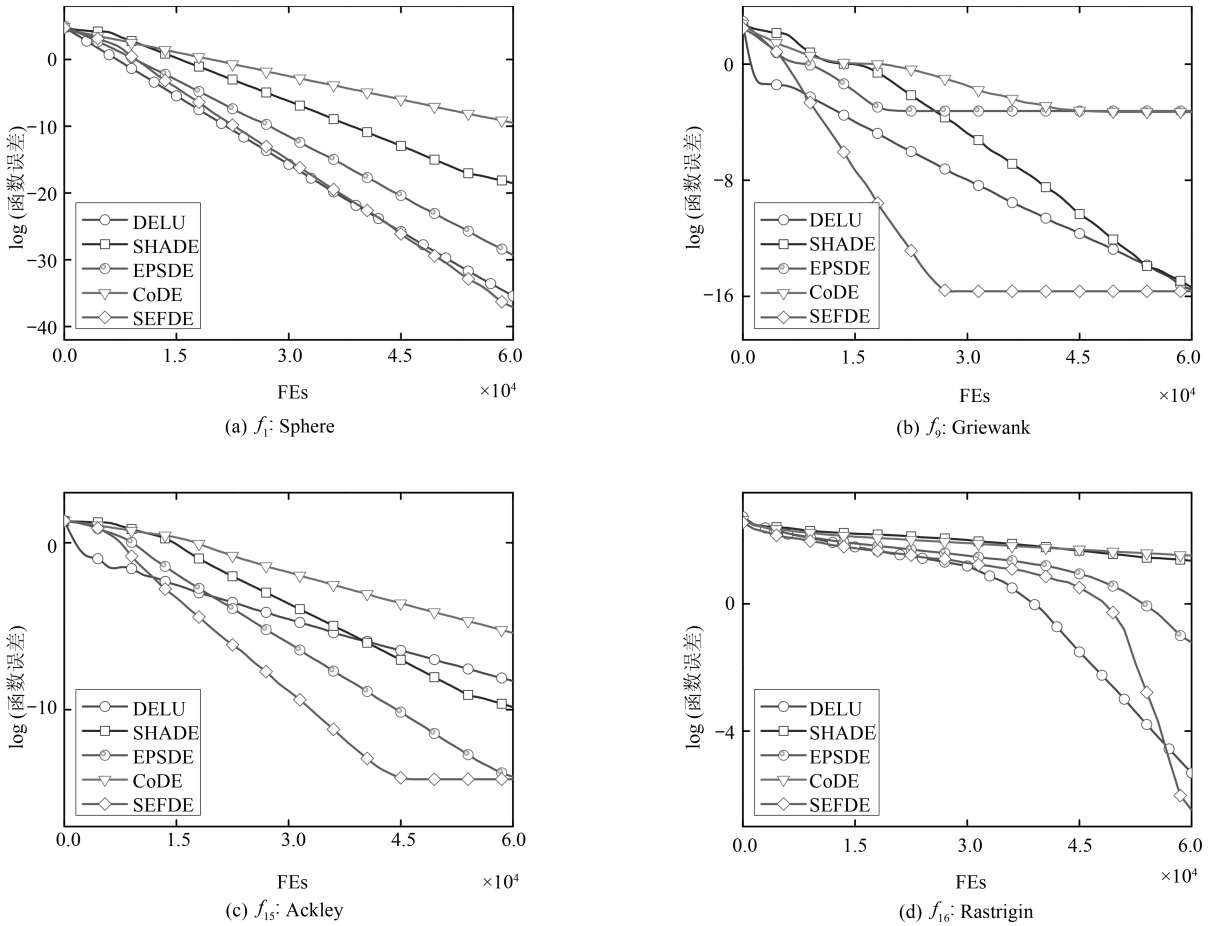


图 7 状态评价因子变化曲线

Fig. 7 The curve graph of state judgment factor



(a) f_1: Sphere (b) f_9: Griewank

(c) f_{15}: Ackley (d) f_{16}: Rastrigin

图 8 DELU、SHADE、EPSDE、CoDE、SEFDE 的平均收敛速度曲线图

Fig. 8 The curve graph of mean convergence speed on DELU、SHADE、EPSDE、CoDE、SEFDE

用合理的策略致使状态评价因子曲线下降, SEFDE 进入局部增强阶段, 最终快于其他算法收敛至全局最优解.

上述实验结果表明, SEFDE 相对于 DELU、SHADE、EPSDE 和 CoDE 这些 State-of-the-art 算法, 在计算代价和可靠性、求解精度、收敛速度方面都具有显著优势.

2.3 与非 DE 算法比较

选用 CLPSO[32]、CMA-ES[33]、GL-25[34] 这 3 种经典的非 DE 算法, 采用性能指标函数误差的平均值 (Mean) 和标准差 (Std), 与 SEFDE 算法进行求解精度的比较. 测试函数选择表 1 所示的 20 个问题, 而终止条件的设置则出于以下考虑: 第 2.2.2 节比较求解精度的实验中, State-of-the-art 算法收敛性能好, 在 FEs = $2\,000N$ 的终止条件下即能得到多数函数的满意解, 但本节中选用的经典非 DE 算法一般需要更多次的函数评价次数才能收敛. 因此参考各算法的原文献, 本实验中将 CLPSO、CMA-ES、GL-25、SEFDE 这 4 种算法的终止条件均设置为函数评价次数 FEs = $5\,000N$, 其中 N 为测试函数各自的维数.

Liang 等[32] 提出的综合学习粒子群算法 (CLPSO), 该算法采用新型学习机制利用其他粒子的历史最优信息更新每个粒子的速度, 保留粒子群的多样性以避免早熟收敛; Hansen 等[33] 提出的基于协方差矩阵的自适应进化策略算法 (CMA-ES), 该算法提出变参数控制策略和两级去随机原理, 并利用协方差矩阵更新正态分布, 有效利用演化路径而不是单一的搜索步骤进行全局优化; Garcia-Martinez 等[34] 提出基于父代中心交叉操作的遗传算法 (GL-25), 利用父代中心实参交叉操作提高算法的全局和局部搜索能力. 表 6 给出了 CLPSO、CMA-ES、GL-25 和本文所提 SEFDE 算法在 30 次独立运行所获得的函数误差的平均值和标准差. 对比表 6 所列数据, 在给定 $5\,000N$ 的函数评价次数内, SEFDE 有 16 个测试函数的优化结果优于 CLPSO、CMA-ES 和 GL-25 这 3 种非 DE 算法. 相比于 CLPSO, SEFDE 对测试函数 f_8 和 f_{10} 的优化结果较差, 二者对函数 f_6、f_{11}、f_{12} 均能找到其全局最优解, 但 SEFDE 有 15 个函数的优化结果优于 CLPSO; SEFDE 对测试函数 f_7、f_8 和 f_{19} 的优化结果逊于 CMA-ES, 但对 16 个测试函数的优化优于 CMA-ES, 对函数 f_6 有相近的优化效果; SEFDE 对 16 个测试函数的优化结果优于 GL-25, 对函数 f_4、f_6 的优化与 GL-25 的优化效果相近, 只对函数 f_5 优化逊于 GL-25. 与第 2.2 节中 State-of-the-art 算法比较的结论类似, 无论数值还是显著性, SEFDE 算法在解的质量和稳定性方面比 CLPSO、CMA-ES、GL-25 更具优势.

表 6　CLPSO、CMA-ES、GL-25 和 SEFDE 的优化结果性能对比数据, 平均值 (标准差)

Table 6　Compared data of optimization results on CLPSO、CMA-ES、GL-25 and SEFDE, Mean (Std)

Fun	N	CLPSO	CMA-ES	GL-25	SEFDE
		Mean error (Std Dev)	Mean error (Std Dev)	Mean error (Std Dev)	Mean error (Std Dev)
f_1	30	9.13E−14(3.33E−14)$^+$	1.97E−29(2.07E−30)$^+$	4.78E−87(1.69E−86)$^+$	**1.26E−103(2.39E−103)**
f_2	30	9.79E−15(5.44E−15)$^+$	3.75E−28(4.61E−29)$^+$	2.66E−78(1.25E−77)$^+$	**2.40E−112(3.36E−112)**
f_3	30	4.48E−09(1.48E−09)$^+$	2.03E−14(9.76E−16)$^+$	2.98E−28(1.12E−27)$^+$	**3.68E−60(3.36E−112)**
f_4	30	3.37E−16(7.98E−17)$^+$	4.81E−17(5.60E−17)$^+$	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**
f_5	30	9.33E−14(6.40E−14)$^+$	1.50E−24(2.15E−25)$^+$	**4.38E−85(2.40E−84)**$^-$	2.42E−45(2.29E−45)
f_6	30	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**$^\approx$	**0.00E+00(0.00E+00)**
f_7	30	4.73E+00(1.20E+00)$^+$	**2.95E−27(4.82E−28)**$^-$	3.14E−02(8.77E−02)$^-$	2.66E−01(2.28E−01)
f_8	30	1.95E+01(2.62E+00)$^-$	**2.66E−01(1.01E+00)**$^-$	2.21E+01(6.19E−01)$^+$	2.19E+01(5.02E−01)
f_9	30	2.40E−09(3.67E−09)$^+$	1.40E−03(3.70E−03)$^+$	1.61E−15(4.55E−15)$^+$	**0.00E+00(0.00E+00)**
f_{10}	30	1.54E−01(2.09E−02)$^-$	2.50E+02(1.23E+01)$^+$	2.95E+00(1.00E+00)$^+$	**1.30E−01(6.42E−02)**
f_{11}	30	**0.00E+00(0.00E+00)**$^\approx$	5.19E+03(6.07E+02)$^+$	4.46E+03(1.36E+03)$^+$	**0.00E+00(0.00E+00)**
f_{12}	30	**0.00E+00(0.00E+00)**$^\approx$	1.30E+01(2.50E+00)$^+$	3.14E−05(3.97E−14)$^+$	**0.00E+00(0.00E+00)**
f_{13}	30	1.13E−17(6.31E−18)$^+$	9.13E−01(1.06E+00)$^+$	8.82E−31(4.10E−30)$^+$	**1.57E−32(0.00E+00)**
f_{14}	30	5.94E−17(3.31E−17)$^+$	1.10E−03(3.35E−03)$^+$	1.28E−30(5.05E−30)$^+$	**1.35E−32(0.00E+00)**
f_{15}	30	9.76E−08(2.38E−08)$^+$	1.93E+01(1.97E−01)$^+$	1.09E−13(1.91E−13)$^+$	**7.55E−15(0.00E+00)**
f_{16}	30	9.41E−07(6.48E−07)$^+$	2.20E+02(5.64E+01)$^+$	3.07E+01(2.71E+01)$^+$	**0.00E+00(0.00E+00)**
f_{17}	30	3.49E−16(1.59E−16)$^+$	1.73E−02(3.93E−02)$^+$	1.26E+02(1.09E+01)$^+$	**1.57E−32(0.00E+00)**
f_{18}	30	2.52E−14(1.22E−14)$^+$	2.20E−03(4.47E−03)$^+$	1.97E+03(1.93E+02)$^+$	**1.35E−32(0.00E+00)**
f_{19}	30	6.26E+03(9.93E+02)$^+$	**5.59E−10(7.36E−11)**$^-$	2.49E+03(4.07E+02)$^+$	5.60E+02(1.36E+02)
f_{20}	30	2.33E−04(9.49E−05)$^+$	8.96E−02(1.46E−01)$^+$	3.55E−04(9.33E−04)$^+$	**1.23E−10(2.47E−10)**
+/ ≈ /−		15/3/2	16/1/3	16/2/2	−/−/−

表 7 SHADE、IDE、ZPEDE、SinDE、SEFDE 的优化结果性能对比数据, 平均值 (标准差)

Table.7 Compared data of optimization results on SHADE、IDE、ZEPDE、SinDE、SEFDE, Mean (Std)

Fun	N	SHADE	IDE	ZEPDE	SinDE	SEFDE
		Mean error (Std Dev)	Mean error (Std Dev)	Mean error (Std Dev)	Mean error (Std Dev)	Mean error (Std Dev)
F_1	30	**0.00E+00(0.00E+00)**$^{\approx}$	**0.00E+00(0.00E+00)**$^{\approx}$	**0.00E+00(0.00E+00)**$^{\approx}$	2.27E−13(1.53E−28)$^{+}$	**0.00E+00(0.00E+00)**
F_2	30	**2.66E+04(1.13E+04)**$^{-}$	1.68E+06(4.23E+05)$^{-}$	1.97E+05(7.53E+04)$^{-}$	2.66E+06(8.33E+05)$^{-}$	2.85E+07(4.18E+06)
F_3	30	8.80E+05(1.96E+06)$^{+}$	1.38E+05(1.85E+05)$^{+}$	1.50E+06(2.07E+06)$^{+}$	1.01E+05(3.77E+05)$^{+}$	**5.80E+04(3.74E+04)**
F_4	30	**1.61E−03**(1.41E−03)$^{-}$	6.85E+03(1.10E+03)$^{+}$	7.66E−01(4.78E−01)$^{-}$	8.28E+03(1.54E+03)$^{+}$	3.13E+03(1.12E+03)
F_5	30	**0.00E+00**(0.00E+00)$^{\approx}$	**0.00E+00(0.00E+00)**$^{\approx}$	**0.00E+00(0.00E+00)**$^{\approx}$	1.14E−13(7.65E−29)$^{+}$	**0.00E+00(0.00E+00)**
F_6	30	4.28E+01(5.52E+00)$^{+}$	4.34E+01(2.62E−04)$^{+}$	4.34E+01(3.18E−13)$^{+}$	4.34E+01(1.44E−14)$^{+}$	**1.52E+01(1.95E−01)**
F_7	30	2.33E+01(9.32E+00)$^{-}$	3.18E+00(1.55E+00)$^{-}$	1.37E+01(4.88E+00)$^{-}$	**6.10E−01**(5.97E−01)$^{-}$	2.80E+01(7.96E+00)
F_8	30	2.09E+01(1.68E−01)$^{+}$	2.11E+01(2.44E−02)$^{+}$	2.11E+01(1.17E−01)$^{+}$	2.11E+01(3.59E−02)$^{+}$	**2.09E+01(2.78E−02)**
F_9	30	5.54E+01(1.98E+00)$^{+}$	3.56E+01(5.54E+00)$^{+}$	3.74E+01(5.85E+00)$^{+}$	3.48E+01(4.34E+00)$^{+}$	**3.04E+01(6.01E−01)**
F_{10}	30	7.37E−02(3.67E−02)$^{+}$	4.38E−02(2.17E−02)$^{+}$	1.37E−01(6.96E−02)$^{+}$	7.93E−02(3.57E−02)$^{+}$	**2.71E−02(1.69E−03)**
F_{11}	30	**0.00E+00(0.00E+00)**$^{\approx}$	**0.00E+00(0.00E+00)**$^{\approx}$	3.65E−01(6.12E−01)$^{+}$	5.92E+00(2.86E+00)$^{+}$	**0.00E+00(0.00E+00)**
F_{12}	30	5.86E+01(1.11E+01)$^{-}$	6.89E+01(8.82E+00)$^{-}$	6.04E+01(1.76E+01)$^{-}$	**5.61E+01**(1.41E+01)$^{-}$	1.21E+02(9.62E+00)
F_{13}	30	1.45E+02(1.95E+01)$^{+}$	1.34E+02(2.28E+01)$^{+}$	1.32E+02(3.62E+01)$^{+}$	1.39E+02(3.41E+01)$^{+}$	**1.24E+02(1.29E+00)**
F_{14}	30	**3.45E−02(1.93E−02)**$^{-}$	1.17E+02(8.38E+01)$^{+}$	4.83E+00(2.70E+00)$^{-}$	2.34E+02(9.23E+01)$^{-}$	5.85E+00(2.05E+00)
F_{15}	30	6.82E+03(4.41E+02)$^{+}$	6.54E+03(5.91E+02)$^{+}$	6.59E+03(9.36E+03)$^{+}$	6.80E+03(1.00E+03)$^{+}$	**5.95E+03(2.37E+02)**
F_{16}	30	1.28E+00(2.07E−01)$^{-}$	1.59E+00(2.36E−01)$^{-}$	**7.82E−01(6.74E−01)**$^{-}$	2.08E+00(3.66E−01)$^{-}$	1.02E+02(1.66E−01)
F_{17}	30	5.08E+01(4.27E−14)$^{+}$	5.92E+01(1.41E+00)$^{+}$	5.11E+01(1.60E−01)$^{+}$	6.52E+01(3.47E+00)$^{+}$	**4.21E+01(6.06E+01)**
F_{18}	30	1.37E+02(1.29E+01)$^{-}$	1.68E+02(1.27E+01)$^{-}$	**1.03E+02(1.19E+01)**$^{-}$	1.41E+02(2.27E+01)$^{-}$	3.01E+02(9.05E+00)
F_{19}	30	2.64E+00(2.83E−01)$^{-}$	**2.24E+00(3.66E−01)**$^{-}$	3.71E+00(7.55E−01)$^{-}$	4.85E+00(8.82E−01)$^{-}$	1.03E+02(1.58E−01)
F_{20}	30	1.93E+01(7.70E−01)$^{-}$	1.93E+01(4.47E−01)$^{-}$	1.97E+01(7.88E−01)$^{-}$	**1.92E+01**(7.52E−01)$^{-}$	1.12E+02(9.46E−02)
F_{21}	30	8.45E+02(3.63E+02)$^{+}$	7.32E+02(3.82E+02)$^{+}$	6.33E+02(4.48E+02)$^{+}$	5.84E+02(4.22E+02)$^{+}$	**5.03E+02(4.03E+01)**
F_{22}	30	**1.33E+01(7.12E+00)**$^{-}$	6.88E+01(2.03E+01)$^{-}$	4.23E+02(5.75E+02)$^{+}$	3.51E+02(2.72E+02)$^{+}$	2.41E+02(1.24E+01)
F_{23}	30	7.63E+03(6.58E+02)$^{+}$	7.32E+03(6.92E+02)$^{+}$	7.02E+03(8.73E+02)$^{+}$	6.59E+03(8.47E+02)$^{+}$	**6.38E+03(2.25E+02)**
F_{24}	30	2.34E+02(1.01E+01)$^{+}$	2.02E+02(1.14E+00)$^{-}$	2.35E+02(1.09E+01)$^{+}$	**2.00E+02**(1.34E−01)$^{-}$	2.32E+02(1.18E+00)
F_{25}	30	3.40E+02(3.09E+01)$^{+}$	3.03E+02(1.09E+01)$^{+}$	3.23E+02(1.31E+01)$^{+}$	2.97E+02(1.33E+01)$^{+}$	**2.58E+02(1.05E+01)**
F_{26}	30	2.58E+02(8.08E+01)$^{-}$	**2.23E+02(4.46E+01)**$^{-}$	2.27E+02(6.20E+01)$^{-}$	2.76E+02(5.96E+01)$^{-}$	3.02E+02(1.90E−01)
F_{27}	30	9.36E+02(3.07E+02)$^{+}$	**3.58E+02(3.30E+01)**$^{-}$	9.38E+02(1.40E+02)$^{+}$	4.75E+02(1.55E+02)$^{-}$	6.11E+02(1.62E+02)
F_{28}	30	4.58E+02(4.13E+02)$^{+}$	**4.00E+02(0.00E+00)**$^{\approx}$	**4.00E+02(0.00E+00)**$^{\approx}$	**4.00E+02**(0.00E+00)$^{\approx}$	**4.00E+02(0.00E+00)**
$+/\approx/-$		14/3/11	13/4/11	15/3/10	16/1/11	−/−/−

2.4 基于 CEC2013 测试集的比较

为了进一步验证所提算法 SEFDE 的优化性能, 采用 SHADE[27]、IDE[13]、ZEPDE[35]、SinDE[36] 这 4 种近年 CEC 竞赛中的算法作为对比算法, 同 SEFDE, 对 CEC2013 测试集进行实验. CEC2013 测试集包括 28 个函数, 其中有 5 个单模函数 F_1-F_5, 15 个基本多模函数 F_6-F_{20} 和 8 个复合函数 $F_{21}-F_{28}$, 具体函数描述详见文献 [37], 并且按文献 [37] 所述, 各算法终止条件均设为 $10\,000N$, N 设置为 30 维, 以各算法独立运行 30 次所得函数误差的平均值和标准差来衡量求解精度.

Tanabe 等[27] 提出了 CEC2013 中性能最好的 DE 改进算法 SHADE, 该算法利用成功参数设置的历史经验指导子代控制参数的选择, 实现控制参数的自适应调整; Tang 等[13] 提出了基于个体依赖机制的差分进化算法 (IDE), 通过个体适应值的差异调整控制参数, 并且根据进化过程中的不同阶段为个体分配相应的策略; Fan 等[35] 提出了参数组合区域进化和适应性变异策略的自适应差分进化算法 (ZEPDE), 该算法中变异策略随种群进化适应性调整, 参数组合在其各自的区域内自适应进化, 而且能够自主寻找附近的最优值; Draa 等[36] 提出了一种正弦差分进化算法 (SinDE), 该算法设计了新的正弦方程调整步长因子和交叉概率, 以平衡全局探测能力和局部搜索能力.

对比表 7 数据, 鉴于 CEC2013 测试集函数的复杂性和对比算法的优越性, SEFDE 算法对 CEC2013 全部 28 个测试函数有 15 个函数的优化结果优于对比算法, 可表明 SEFDE 在解的质量和稳定性方面具有优势. 具体而言, 所提算法 SEFDE 与 SHADE 相比, 对 14 个函数的优化效果较好, 对 11 个函数的优化效果较差, 以及对 3 个函数 F_1、F_5、F_{11} 有相近的优化效果; SEFDE 算法有 11 个函数的优化结果逊于 IDE, 但对 13 个函数的优化优于 IDE, 另外二者均能得到函数 F_1、F_5、F_{11}、F_{28}

的全局最优解; SEFDE 对 15 个函数的优化结果优于 ZEPDE, 有 10 个函数的优化结果逊于 ZEPDE, 对函数 F_1、F_5、F_{28} 优化结果相近; 与 SinDE 对比, SEFDE 有 16 个函数的优化结果较好, 对其他 11 个函数的优化结果较差, 对 F_{28} 的优化与 SEFDE 结果相同.

3 结论

为了平衡算法的全局探测和局部增强, 本文提出基于状态估计反馈的策略自适应差分进化算法 SEFDE, 基于抽象凸理论建立进化状态估计模型以获取估计信息, 并结合进化知识建立状态评价因子, 进而基于状态评价因子的反馈信息自适应判定当前种群个体所处于的阶段, 切换变异策略以指导种群进化. 通过与主流改进 DE 算法与非 DE 算法进行对比实验, 验证了所提 SEFDE 算法在计算代价、可靠性、求解精度及收敛速度方面的优越性能, 实验结果表明 SEFDE 是一种有效的全局优化算法. 下一步的主要工作将针对复杂实际问题进行应用研究.

References

1 Storn R, Price K. Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 1997, **11**(4): 341−359

2 Das S, Suganthan P N. Differential evolution: a survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 2011, **15**(1): 4−31

3 Raghunathan T, Ghose D. Differential evolution based 3-D guidance law for a realistic interceptor model. *Applied Soft Computing*, 2014, **16**: 20−33

4 Hu Rong, Qian Bin. A hybrid differential evolution algorithm for stochastic flow shop scheduling with limited buffers. *Acta Automatica Sinica*, 2009, **35**(12): 1580−1586
(胡蓉, 钱斌. 一种求解随机有限缓冲区流水线调度的混合差分进化算法. 自动化学报, 2009, **35**(12): 1580−1586)

5 Pandit M, Srivastava L, Sharma M. Environmental economic dispatch in multi-area power system employing improved differential evolution with fuzzy selection. *Applied Soft Computing*, 2015, **28**: 498−510

6 Zhang G J, Zhou X G, Yu X F, Hao X H, Yu L. Enhancing protein conformational space sampling using distance profile-guided differential evolution. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2017, **14**(6): 1288−1301

7 Zhu Teng, Wang Jing-Chun, Xiong Zhi-Hua. DE-based nonlinear model predictive control of a pH neutralization process. *Acta Automatica Sinica*, 2010, **36**(1): 159−163
(朱腾, 王京春, 熊智华. 基于改进 DE-NMPC 的酸碱中和反应 pH 值控制. 自动化学报, 2010, **36**(1): 159−163)

8 Qiao D P, Pang G K H. A modified differential evolution with heuristic algorithm for nonconvex optimization on sensor network localization. *IEEE Transactions on Vehicular Technology*, 2016, **65**(3): 1676−1689

9 Das S, Mullick S S, Suganthan P N. Recent advances in differential evolution — an updated survey. *Swarm and Evolutionary Computation*, 2016, **27**: 1−30

10 Dragoi E N, Dafinescu V. Parameter control and hybridization techniques in differential evolution: a survey. *Artificial Intelligence Review*, 2016, **45**(4): 447−470

11 Cheng M Y, Tran D H. Two-phase differential evolution for the multiobjective optimization of time — cost tradeoffs in resource-constrained construction projects. *IEEE Transactions on Engineering Management*, 2014, **61**(3): 450−461

12 Liu Z Z, Wang Y, Yang S X, Cai Z X. Differential evolution with a two-stage optimization mechanism for numerical optimization. In: Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC). Vancouver, BC, Canada: IEEE, 2016. 3170−3177

13 Tang L X, Dong Y, Liu J Y. Differential evolution with an individual-dependent mechanism. *IEEE Transactions on Evolutionary Computation*, 2015, **19**(4): 560−574

14 Yu W J, Shen M E, Chen W N, Zhan Z H, Gong Y J, Lin Y, Liu Q, Zhang J. Differential evolution with two-level parameter adaptation. *IEEE Transactions on Cybernetics*, 2014, **44**(7): 1080−1099

15 Zhang Gui-Jun, Chen Ming, Zhou Xiao-Gen. Two-stage differential evolution algorithm using dynamic niche radius for multimodal optimization. *Control and Decision*, 2016, **31**(7): 1185−1191
(张贵军, 陈铭, 周晓根. 动态小生境半径两阶段多模态差分进化算法. 控制与决策, 2016, **31**(7): 1185−1191)

16 Price K, Storn R M, Lampinen J A. *Differential Evolution — A Practical Approach to Global Optimization*. Berlin Heidelberg: Springer, 2005: 1−24

17 Zhang Gui-Jun, He Yang-Jun, Guo Hai-Feng, Feng Yuan-Jing, Xu Jian-Ming. Differential evolution algorithm for multimodal optimization based on abstract convex underestimation. *Journal of Software*, 2013, **24**(6): 1177−1195
(张贵军, 何洋军, 郭海锋, 冯远静, 徐建明. 基于广义凸下界估计的多模态差分进化算法. 软件学报, 2013, **24**(6): 1177−1195)

18 Zhou Xiao-Gen, Zhang Gui-Jun, Hao Xiao-Hu. Differential evolution algorithm with local abstract convex region partition. *Acta Automatica Sinica*, 2015, **41**(7): 1315−1327
(周晓根, 张贵军, 郝小虎. 局部抽象凸区域剖分差分进化算法. 自动化学报, 2015, **41**(7): 1315−1327)

19 Beliakov G. Extended cutting angle method of global optimization. *Pacific Journal of Optimization*, 2008, **4**(1): 153−176

20 Beliakov G, Lim K F. Challenges of continuous global optimization in molecular structure prediction. *European Journal of Operational Research*, 2007, **181**(3): 1198−1213

21 Zhou Xiao-Gen, Zhang Gui-Jun, Hao Xiao-Hu, Yu Li. Differential evolution algorithm based on local lipschitz underestimate supporting hyperplanes. *Chinese Journal of Computers*, 2016, **39**(12): 2631−2651
(周晓根, 张贵军, 郝小虎, 俞立. 一种基于局部 Lipschitz 下界估计支撑面的差分进化算法. 计算机学报, 2016, **39**(12): 2631−2651)

22 Ali M M, Khompatraporn C, Zabinsky Z B. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of Global Optimization*, 2005, **31**(4): 635−672

23 Sarker R A, Elsayed S M, Ray T. Differential evolution with dynamic parameters selection for optimization problems. *IEEE Transactions on Evolutionary Computation*, 2014, **18**(5): 689−707

24 Dragoi E N, Dafinescu V. Parameter control and hybridization techniques in differential evolution: a survey. *Artificial Intelligence Review*, 2015, **45**(4): 447−470

25 Zhou X G, Zhang G J. Abstract convex underestimation assisted multistage differential evolution. *IEEE Transactions on Cybernetics*, 2017, **47**(9): 2730−2741

26 Zhou X G, Zhang G J, Hao X H, Yu L. A novel differential evolution algorithm using local abstract convex underestimate strategy for global optimization. *Computers & Operations Research*, 2016, **75**: 132−149

27 Tanabe R, Fukunaga A. Success-history based parameter adaptation for differential evolution. In: Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC). Cancun, Mexico: IEEE, 2013. 71−78

28 Mallipeddi R, Suganthan P N, Pan Q K, Tasgetiren M F. Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, 2011, **11**(2): 1679−1696

29 Wang Y, Cai Z X, Zhang Q F. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, 2011, **15**(1): 55−66

30 Corder G W, Foreman D I. *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*. Hoboken, NJ: Wiley 2009. 38−55

31 Shang Y W, Qiu Y H. A note on the extended Rosenbrock function. *Evolutionary Computation*, 2006, **14**(1): 119−126

32 Liang J J, Qin A K, Suganthan P N, Baskar S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, 2006, **10**(3): 281−295

33 Hansen N, Ostermeier A. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 2001, **9**(2): 159−195

34 García-Martínez C, Lozano M, Herrera F, Molina D, Sánchez A M. Global and local real-coded genetic algorithms based on parent-centric crossover operators. *European Journal of Operational Research*, 2008, **185**(3): 1088−1113

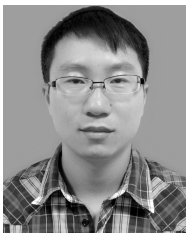
35 Fan Q Q, Yan X F. Self-adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies. *IEEE Transactions on Cybernetics*, 2016, **46**(1): 219−232

36 Draa A, Bouzoubia S, Boukhalfa I. A sinusoidal differential evolution algorithm for numerical optimisation. *Applied Soft Computing*, 2015, **27**: 99−126

37 Liang J J, Qu B Y, Suganthan P N, Hernández-Díaz A G. Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization, Technical Report 201212, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, 2013.

王柳静 浙江工业大学信息工程学院博士研究生. 主要研究方向为智能信息处理, 优化理论及算法设计.
E-mail: wlj@zjut.edu.cn
(**WANG Liu-jing** Ph. D. candidate at the College of Information Engineering, Zhejiang University of Technology. Her research interest covers intelligent information processing, optimization theory, and algorithm design.)

张贵军 浙江工业大学信息工程学院教授. 主要研究方向为智能信息处理, 优化理论及算法设计, 生物信息学. 本文通信作者. E-mail: zgj@zjut.edu.cn
(**ZHANG Gui-Jun** Professor at the College of Information Engineering, Zhejiang University of Technology. His research interest covers intelligent information processing, optimization theory and algorithm design, and bioinformatics. Corresponding author of this paper.)



周晓根 浙江工业大学信息工程学院博士研究生. 主要研究方向为智能信息处理, 优化理论及算法设计.
E-mail: zhouxiaogen53@126.com
(**ZHOU Xiao-Gen** Ph. D. candidate at the College of Information Engineering, Zhejiang University of Technology. His research interest covers intelligent information processing, optimization theory, and algorithm design.)