基于种群状态信息的自适应差分进化算法

麦伟杰[1]，刘伟莉[2]，钟竞辉[1]

（1. 华南理工大学计算机科学与工程学院，广东 广州 510006；2. 广东技术师范大学计算机科学学院，广东 广州 510665）

摘 要：种群的局部最优与停滞状态会严重影响差分进化（DE）算法的性能。为了消除这 2 种状态引起的不利因素，提出一种带有种群状态处理措施的改进 DE 算法。当种群处于局部最优状态时，运用限制记忆的拟牛顿（LBFGS）方法对种群中的个体进行随机学习提高解的全局质量，通过高斯变异生成新个体，促使种群跳出局部最优；当算法处于停滞状态时，运用种群的协方差矩阵，通过空间坐标旋转对目标个体进行重组，从而抑制种群停滞状态，加强算法全局搜索能力。此外，算法设计一种新型的选择策略，该选择策略设置一个存放经贪心选择后被遗弃个体的外部存档。当实验个体劣于目标个体时，算法则不再以贪心选择策略生成下一代，而是围绕外部存档进行合理的智能选择，使算法向全局最优收敛。实验表明，通过与先进的 8 个 DE 算法在 29 个标准的测试函数比较，所提算法在解的精确度和收敛速度均具有更好的性能。

关键词：差分进化；选择策略；种群状态信息；协方差矩阵；外部存档

中图分类号：TP18

文献标志码：A

DOI: 10.11959/j.issn.1000−436x.2023113

Self-adaptive differential evolution algorithm based on population state information

MAI Weijie[1], LIU Weili[2], ZHONG Jinghui[1]

1. School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China

2. College of Computer Science, Guangdong Polytechnic Normal University, Guangzhou 510665, China

Abstract: The local optimum and stagnation state information of the population seriously affects the performance of differential evolution (DE) algorithm. An advanced DE algorithm with population state processing measures was proposed to address the above two issues. When the population falled into the local optimum, the individuals in the population were learned randomly by LBFGS method to improve the global quality of the solution, and Gaussian mutation was employed to trigger new individuals to jump out of local optimum. As for the stagnation state, the covariance matrix of the population was applied to reorganize the target individuals based on the rotation of the spatial coordinates to suppress the stagnation state of the population and enhance the global search ability of the algorithm. In addition, a new selection strategy was designed, which built an external archive to store abandoned individuals after greedy selection. When the trial individual was inferior to the target individual, the algorithm no longer generated the next generation with greedy selection strategy, but made reasonable intelligent selection around the external archive to ensure that the algorithm converges to the global optimum. Compared with eight state-of-the-art DE algorithms on 29 benchmark functions, the experimental results show that the proposed algorithm has better performance in terms of the solution accuracy and convergence speed.

Keywords: differential evolution, selection strategy, population status information, covariance matrix, external archive

---

收稿日期：2023−02−20；**修回日期**：2023−05−16

基金项目： 国家自然科学基金资助项目（No.62076098）；广东省基础与应用基础研究基金联合基金资助项目（No.2021A1515110072）

Foundation Items: The National Natura1 Science Foundation of China (No.62076098), Guangdong Basic and Applied Basic Research Foundation (No.2021A1515110072)

0　引言

优化问题一直存在于科学与工程技术中，当今流行的人工智能技术也以优化问题为研究要点。差分进化（DE, differential evolution）算法是一种优秀的进化算法，由 Storn 和 Price[1]为了解决 Chebyshev 多项式拟合问题共同提出。相对于其他进化算法，DE 算法由于具有参数少、结构简单、优化率高效及鲁棒性等特点而被广泛应用到不同工程领域。然而，DE 也存在不足之处，如容易陷入局部最优、求解不同问题依赖不同参数，这些缺点在求解多峰、高维问题时尤其明显。

国内外众多学者针对 DE 的缺点进行了改进，提出了具有特定策略的变种算法。Brest 等[2]提出了自适应差分进化算法 JDE，该算法将缩放因子 F 和交叉因子 CR 参与个体编码，初始化时，个体 x_i 对应一组在给定范围内产生的 F_i 以及 CR_i；在文献[3]中，缩放参数 Fm、交叉参数自适应地由柯西分布 Cauchy(Fm,0.1)与高斯分布 Gaussian(Fm,0.1)产生，类似的还有文献[4]。为了解决不同发布时间和作业大小的单个批处理机（BPM）的调度问题，Zhou 等[5]提出一种自适应差分进化算法，该算法中 F_i 与 CR_i 在进化的前 50 代通过几个预先设置好的均匀分布集合随机触发，50 代后即根据实验向量成功触发的新个体加权概率得到 F_i 及 CR_i。变异策略为 DE 的另一个研究重点，文献[6]针对复杂的多目标优化问题提出一种基于动态种群的多策略差分进化算法，该算法将种群划分为 3 个子种群，每个子种群对应不同进化策略，在进化迭代中通过不同子种群之间的策略相互配合协同进化来提高算法的性能。针对约束优化问题，Wang 等[7]提出一种具有 3 种不同优势的实验向量生成策略的差分进化算法，为了在多样性和收敛性之间取得平衡，利用其中一种实验向量来增加多样性，另外 2 种来测试算法的收敛性。此外，根据种群自身特点改善 DE 也是研究对象，如文献[8]通过具有余弦相似性自组织映射逐步学习和提取个体之间的邻域关系，在变异过程中利用个体的邻域构建进化方向来指导搜索。

大多数文献均关注研究 DE 的参数控制及其变异策略的利用，但探讨 DE 的选择操作以及种群状态信息的研究却不多。为了加强这方面的研究，本文主要进行如下 2 个方面的研究。

1) 研究种群状态信息，并提出相应的处理方法。若种群陷入局部状态，运用限制记忆的拟牛顿（LBFGS）方法[9]的非凸优化能力对种群中的个体进行随机学习提高解的全局质量，并应用高斯变异触发新个体；针对停滞状态，运用种群的协方差矩阵，基于空间坐标旋转对目标个体进行重组[10]，打破种群停滞僵局状态，增强算法全局搜索能力。

2) 提出新的选择策略。当种群处于停滞状态时，即使种群继续迭代，原始 DE 的贪心选择策略也很难触发更好的下一代。为了摆脱这个状态，本文设置一个存放经贪心选择策略后被遗弃个体的外部存档。经过多次迭代后，当实验个体劣于目标个体时，暗示着种群可能陷入停滞，算法不再以贪心选择策略生成下一代，而是围绕外部存档进行探索，合理智能地选择其他个体作为下一代。

1　相关工作

1.1　DE 及其变种算法

科学研究和工程应用中经常会遇到寻找全局最优值的问题。这类问题通常被称为全局优化问题（GOP, global optimization problem）。一般情况下，GOP 的最小值优化在数学上定义为

$$\min f(\boldsymbol{X}),\ \boldsymbol{X}=(x_1,x_2,x_D)\in\boldsymbol{R} \tag{1}$$

其中，$\boldsymbol{R}\subseteq\prod_{i=1}^{D}[\boldsymbol{x}_{i,L},\boldsymbol{x}_{i,U}]$，$-\infty<\boldsymbol{x}_{i,L},\boldsymbol{x}_{i,U}<+\infty$，$i=1,2,\cdots,D$，$D$ 表示决策变量 $\boldsymbol{X}$ 的维度，$\boldsymbol{x}_{i,L}$ 和 $\boldsymbol{x}_{i,U}$ 分别表示决策变量 $\boldsymbol{X}$ 的下限与上限。式(1)的目的是在 D 维空间中找到一个解 $\boldsymbol{X}^*\in\boldsymbol{R}$，使其对于 $\forall\boldsymbol{X}\in\boldsymbol{R}$，均满足 $f(\boldsymbol{X}^*)\leqslant f(\boldsymbol{X})$。

全局优化问题引起了研究人员的极大兴趣，许多自然启发的智能算法被提出用于解决全局优化问题。其中，DE 算法已被成功地应用到解决该类优化问题中，它的流程包括种群初始化、参数设置、个体适应度的评估及变异、交叉、选择步骤和操作，算法中所涉及的参数有缩放因子 F、交叉因子 CR、种群大小 NP 等。在所有参数中，缩放因子 F、交叉因子 CR 对 DE 的性能及收敛敏感，因此其也成为重点研究内容。

一些著名的 DE 变种算法被相继提出。通过学习先前的经验，SADE（DE with strategy adaptation）[11]将实验向量 $\boldsymbol{u}_i$ 与控制参数 F_i、CR_i 关联，在该算法中，实验向量生成策略及其相关的控制参数通过逐渐自适应方式从它们以前的经验中学习得到，然

后找出有前途的解；文献[12]提出触发实验向量的变异策略与其控制参数相对应，其主要思想是将多个实验向量生成策略与每一代的多个控制参数设置随机组合，以创建新的实验向量，这样可以保证生成的实验向量具有良好的质量；Gong 等[13]提出一种基于等级选择的差分进化算法 RankDE，具体来说，在差分变异操作中，变异父代向量不是随机从种群中选择，而是根据一个基于等级的概率来选择，等级越高被选择产生变异向量的概率就越高，在进化变异中，根据计算所得概率选择基向量和差分向量的末端向量，而第三个向量则按照原来的 DE 算法进行随机选择，这样种群会在等级高的个体下引导搜索，加快算法的收敛速度；Draa 等[14]于 2015 年提出 SinDE，指出有关 DE 主要参数（F、CR）的动态改进一般是线性的，这种调整往往允许在一个方向上调整某个值（单调递增或者单调递减），并且要为其设置一个最大值或最小值，为了克服这种不足，作者提出了一种基于正弦公式的方式计算 F、CR，该计算方式充分运用正弦函数的特点，所以不仅可以从 2 个方向来动态调整 F、CR，还可以调整方向，这样可以让算法在调整参数上更加灵活，因此 SinDE 有较好的全局搜索能力和局部开拓能力。本文实验中将以上算法与本文所提算法进行比较。

此外，引入其他技术也成为提高 DE 性能的研究方向。文献[15]引入了 K-means 算法，设计并集成了一种自适应小生境方案，该方案可以动态调整种群中每个小生境的大小，以防止进化搜索过早收敛；Wei 等[16]构造了一种具有动态惩罚半径的惩罚策略来求解多目标优化问题（MMOP）；文献[17]提出了一种应用双层协同的分布技术提高 DE 处理多峰问题收敛性的 TLCDE（two-layer collaborative different evolution）算法；自适应骨架 DE 在文献[18]中也得到研究。近年来，DDE（distributed differential evolution）成为热门研究对象，文献[19]提出了一种自适应 DDE 来缓解参数的敏感性；为了解决多模态优化问题，基于自适应估计分布的无参数小生境 DDE 算法[20]被提出。

1.2 种群状态信息

种群状态信息与种群分布呈正相关，实际上也与其多样性紧密相关，学者们提出了与之相关的技术。Cheng 等[21]提出一种两阶段的差分进化算法，该算法在演化中分为 2 个阶段，第一阶段以自适应变异策略维持种群多样性，第二阶段在变异操作中将精英个体用于基向量，其余参与变异的个体从种群中随机选择，目的是使种群多样性和收敛性得到平衡；文献[20]通过使用探索种群、开发种群和平衡种群 3 个子种群的分布式框架实现种群同时共同进化，不同种群将基于进化状态自适应地选择其合适的变异策略，以充分利用来自个体与相应种群的反馈状态信息；王柳静等[22]通过设计状态评价因子自适应判定种群个体所处的阶段，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；Zhang 等[23]提出一种新的自适应 ε 控制方法，并将其纳入基本差分进化算法中，该算法通过自适应的启发式规则控制 ε 的值以保持种群多样性，防止算法陷入局部最优；文献[24]为与本文对比的算法之一，作者在个体维度层面上研究了与种群多样性相关的情况，其基本思想是基于个体维度的分布来确定种群收敛或停滞的时刻，提出了一种自动增强种群多样性的机制来提高种群的多样性。此外，通过自适应调整参数 F、CR 及种群大小 NP 也可以调整种群的分布，从而达到种群多样化的目标，如 SADE[11]、JDE[2]。

1.3 选择策略

DE 在选择操作中常采用“贪心选择”策略选出下一代新个体，这在一般情况下能起到很好的效果。然而，当种群陷入局部最优或停滞状态时，根据“贪心选择”的原则，好的个体总是被选中，这样会导致算法进入无限“恶化”状态，即使最终达到停止条件也无法收敛[24]。文献[25]指出“贪心选择”策略的劣势，提出了当算法陷入局部最优时采用有偏式的选择帮助个体跳出局部最优状态，同时证明了当种群处于正常状态时，“贪心选择”能够很好地帮助算法达到收敛；文献[26]指出当个体停止更新的步数达到预定值时，即认为种群陷入停滞状态，当种群处于停滞状态时，考虑使用一些候选向量来替换父向量，并在此基础上继续进化，然而，候选向量必须具有良好的适应度值，否则很难触发一个好解；文献[27]基于目标向量设置 3 个被遗弃的个体及 4 个选择标志（flag=0,1,2,3），第一个个体是目标向量所有丢弃的实验向量中的最佳向量，第二个个体是目标向量所有丢弃的实验向量中的次佳向量，第三个个体从所有成功更新的解中随机选择，在选择操作中，它们分别对应标志（flag=0,1,2），当 3 个备选个体均不能改善解的质量时，从基于目

标向量被遗弃的个体中选择历史最好的个体（flag=3）作为下一代。随机排序是进化约束优化的一种非常成功的方法，通常在生存选择过程中使用，Pulido 等[28]使用随机排序程序研究用于约束处理的进化算法的父代选择机制；Zeng 等[29]通过在选择操作中设置标志统计算法的停顿次数，然后分两步操作，1) 当停顿次数达到预定值时，从非停滞个体中随机选择一个个体替换停滞个体进行变异操作和交叉操作，2) 对停滞个体进行突变操作和交叉操作。

现有文献针对种群状态及 DE 选择策略虽然进行了相关研究，但具有片面性，即仅偏向于对种群状态或 DE 选择策略进行探讨，而没有将两者综合在一起考虑。例如，文献[24]提出用个体维度的标准差研究种群状态，而没有专门设计一种选择策略替换贪心选择。有的仅限于对 DE 的选择策略这一操作进行研究，没有提出自己的 DE 算法，缺乏当算法处于停滞或局部时验证提出选择策略所起的作用，如文献[27]。有的只是把重点放在如何利用个体的变异、交叉操作对种群状态进行分析，如文献[29]。综上所述，本文在已有文献的基础上进行统一的研究，在研究种群状态的同时提出一种新型选择策略，并在实验中验证所提算法 NSPSDE（new selection strategy and population state DE）的有效性。

2　NSPSDE 算法

2.1　判别种群状态信息

算法在迭代中种群出现的状态信息可以分为 2 个类别：局部最优、停滞。DE 陷入局部最优或停滞，理论上是由种群分布过于聚集、种群丢失多样性所致。此时，种群中的个体在当代各个分量的分布基本相同，即使继续迭代下去也无法触发更好的个体，因为此时种群各个个体所对应维度的分量趋于同一个值。

根据上述分析，文献[24]提出用个体各个维度的标准差衡量种群是否处于局部最优、停滞状态。但是，该方法仅限于从个体中的每个维度探索相应分量的多样性情况，相应地，当该维度陷入局部最优时，只针对个体相应的维度进行修理，而没有从种群整体判断其状态及采取有效的处理方法触发更好的下一代。基于这个缺陷，本文从整体水平角度判断种群状态，个体在第 g 代的总体均值和总体标准差分别为

$$m_g = \frac{1}{D}\sum_{d=1}^{D}\frac{1}{\mathrm{NP}}\sum_{i=1}^{\mathrm{NP}}x_{i,d,g} \tag{2}$$

$$\mathrm{sd}_g = \sqrt{\frac{1}{D}\sum_{d=1}^{D}\frac{1}{\mathrm{NP}}\sum_{i=1}^{\mathrm{NP}}(x_{i,d,g}-m_g)^2} \tag{3}$$

其中，m_g 表示种群在第 g 代个体维度的总体均值，sd_g 表示种群在第 g 代个体维度的总体标准差，D 表示个体的维度，NP 表示种群大小，$x_{i,d,g}$ 表示个体 $\boldsymbol{x}$ 在第 g 代对应的第 d 维分量。sd_g 值越小，表示种群分布越紧密，从理论上讲，当 $\mathrm{sd}_g=0$ 时，种群处于 D 维几何空间的一点，即局部最优或停滞状态，实际上 $\mathrm{sd}_g<\partial(\partial=1\times10^{-2})$，本文就这 2 种情况进行进一步研究。

1) 种群处于局部最优的条件为式(4)～式(6)成立。

$$\sigma_g^2 = \frac{1}{\mathrm{NP}}\sum_{i=1}^{\mathrm{NP}}(\mathrm{fit}_{i,g}-\mathrm{fitage}_g)^2 \tag{4}$$

$$\sigma < \min(\sigma_{g-1}, \mathrm{mean}(\sigma_g,\sigma_{g-1})) \tag{5}$$

$$\mathrm{fbest}_g > \min(\varepsilon,\varphi_g) \tag{6}$$

其中，σ_g^2 表示个体在第 g 代函数适应度方差，σ_{g-1} 表示上一代的标准差，$\mathrm{mean}(\sigma_g,\sigma_{g-1})$ 表示 σ_g 与 σ_{g-1} 的平均值，fitage_g 表示整体适应度平均值，fbest_g 表示第 g 代最优适应度值。从式(4)和式(5)可以看出，σ_g 越小，表明种群越处于“族聚集”分布，结合式(6)可判断种群处于局部最优状态。其中，φ_g 由式(7)计算得到。

$$\varphi_g = |\,\mathrm{fbest}_g - \mathrm{fbest}_{g-1}\,|\,\varepsilon \tag{7}$$

其中，fbest_{g-1} 为上一代最优个体的适应度，$\varepsilon=1\times10^{-3}$。

当种群陷入局部最优状态时，若目标为多峰高维函数，从函数适应度外观看，此时种群的个体位于凹区域局部最优处或周边。考虑到种群个体维度较大（$D\geqslant100$ 为大规模优化），根据拟牛顿法原理，利用 LBFGS 方法对个体进行学习实现非凸随机优化，可有效地增加算法的随机优化能力，提高算法全局搜索能力。图 1 为处于局部状态的种群经 LBFGS 方法随机学习后所得个体示意，其中，$\boldsymbol{x}_{r1,g}$、

$\boldsymbol{x}_{r2,g}$ 为局部状态中种群的 2 个个体，$\boldsymbol{x}_{r3,g}$ 为局部最优个体（$r1 \neq r2 \neq r3 \in [1, \mathrm{NP}]$）。由 LBFGS 方法可知，位于图 1 中左边局部状态的个体 $\boldsymbol{x}_{r1,g}$、$\boldsymbol{x}_{r2,g}$ 和 $\boldsymbol{x}_{r3,g}$ 通过学习后生成右边相应的随机个体 $\boldsymbol{x}^L_{r1,g}$、$\boldsymbol{x}^L_{r2,g}$ 和 $\boldsymbol{x}^L_{r3,g}$。注意到，当种群陷入局部状态时，所有的个体虽然不是全部位于搜索空间的局部最低点，但它们在搜索空间上与局部最低点的距离接近，因此，可以对种群的个体进行随机学习来改变这种状态。个体学习计算式为

$$\boldsymbol{x}^L_{i,g} = \mathrm{LBFGS}(\boldsymbol{x}_{i,g}),\ i = 1, 2, \cdots, \mathrm{NP} \tag{8}$$

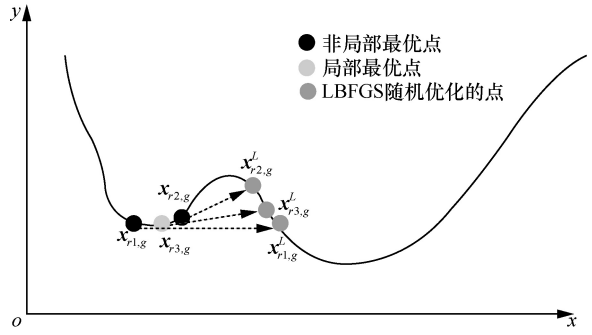


图 1　LBFGS 随机优化示意

通过 LBFGS 方法学习后，为了防止优化后有些个体出现“聚集”现象，本文基于 DE/current-to-best/1 算子，通过高斯局部扰动产生新的个体维持种群多样性。

$$\boldsymbol{x}^{'}_{i,g} = \boldsymbol{x}^L_{i,g} + \omega(\boldsymbol{x}^L_{\mathrm{best},g} - \boldsymbol{x}^L_{i,g}) + \omega(\boldsymbol{x}^L_{r1,g} - \boldsymbol{x}^L_{r2,g}) \tag{9}$$

$$\omega = \mathrm{rand}(F_{\min}, F_{\max})(1 + 0.5\theta) \tag{10}$$

其中，$\boldsymbol{x}^L_{\mathrm{best},g}$ 表示经过 LBFGS 方法学习后最好解对应的个体，rand 表示在 $(F_{\min}, F_{\max})$ 上随机取值，θ 服从均值为 0、方差为 1 的高斯分布，即 $\theta \in \mathrm{Gauss}(0,1)$。式(9)具有良好的平衡能力，在局部状态（即 $(\boldsymbol{x}^L_{\mathrm{best},g} - \boldsymbol{x}^L_{i,g})$）下，该算子围绕 $\boldsymbol{x}^L_{\mathrm{best},g}$ 进行高斯变异，同时保持着随机搜索性（即 $(\boldsymbol{x}^L_{r1,g} - \boldsymbol{x}^L_{r2,g})$），既发挥了 DE 的全局开发能力，也利用了 DE 的局部探测能力。

2) 如果种群在连续 NP（NP 为种群大小）次迭代中最优的适应度 fbest_g 保持不变，即考虑种群处于停滞状态。此时有式(11)成立。

$$\mathrm{fail}_g = \begin{cases} \mathrm{fail}_{g-1} + 1, & \mathrm{abs}(\mathrm{fbest}_g - \mathrm{fbest}_{g-1}) < \varepsilon \\ 0, & \text{其他} \end{cases} \tag{11}$$

其中，初始化 $\mathrm{fail}_0 = 0$，abs 表示取绝对值。

当种群处于停滞状态时，个体的适应度不再随算法的运行发生变化，归根结底是因为种群结构分布处于恒定状态。从几何角度看，该状态反映在目标个体 $\boldsymbol{x}$ 在原定的坐标系中经多次迭代后投影不变。针对这种情况，本文基于旋转坐标系对目标个体进行重组。如图 2 所示，假设种群是由二维个体构成的，向量 $\boldsymbol{x}$、$\boldsymbol{u}$、$\boldsymbol{v}$ 分别表示原始坐标系 xoy 中的目标向量、差分向量、变异变量，向量 $\boldsymbol{x}'$、$\boldsymbol{u}'$、$\boldsymbol{v}'$ 分别表示经原始坐标系 xoy 旋转后 $\boldsymbol{x}$、$\boldsymbol{u}$、$\boldsymbol{v}$ 在坐标系 $x'o'y'$ 获得的相应向量，图 2 中展示了通过坐标旋转能够更好地触发接近全局最优的向量。要对目标向量 $\boldsymbol{x}$ 进行重组，可由 $\boldsymbol{x}' = \boldsymbol{E} \otimes \boldsymbol{x}$ 实现，其中，符号 $\otimes$ 表示向量 $\boldsymbol{x}$ 通过某种数学关系与向量 $\boldsymbol{E}$（本文为特征向量）作用生成重组向量 $\boldsymbol{x}'$。

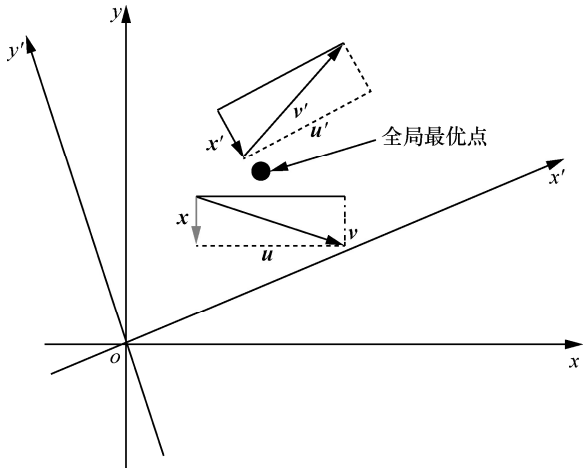


图 2　二维个体通过旋转坐标系随机间接重组

综上可知，种群处于停滞状态是因为其在空间分布中过于聚集，导致所有个体在函数的适应度外观基本一致，而图 2 实质上是通过二维旋转坐标系，利用离散、随机的间接方式产生新的重组向量，这种方法恰好可以克服停滞状态的空间分布聚集问题，提高种群的多样性。由于协方差矩阵具有可正交对角化等代数几何特性，因此可以将这种特性应用到种群个体重组中，即

$$\boldsymbol{E}\boldsymbol{\Sigma}^2\boldsymbol{E}^{\mathrm{T}} = (\boldsymbol{\Sigma}\boldsymbol{E}^{\mathrm{T}})^{\mathrm{T}}(\boldsymbol{\Sigma}\boldsymbol{E}^{\mathrm{T}}) = \boldsymbol{P}^{\mathrm{T}}\boldsymbol{P} \tag{12}$$

其中，$\boldsymbol{E}$ 为种群协方差矩阵的特征向量，$\boldsymbol{\Sigma}$ 为其特征值对角矩阵。令 $\boldsymbol{x} = \boldsymbol{P}$，通过旋转坐标系重组后的目标个体如式(13)所示。

$$\boldsymbol{x}'_{i,g} = \boldsymbol{P}^{\mathrm{T}}\boldsymbol{x}_{i,g} \tag{13}$$

上述过程应用协方差矩阵及其特征值对处于停滞状态的种群进行坐标旋转变换，由于正交投影在旋转中起着重要的作用，因此，变换原理为种群在 D 维空间中沿不同的方向用另一种主要特征表示，打破停滞僵局状态。

2.2 新型选择策略

传统的“贪心选择”策略只有在 DE 算法没有陷入局部最优或停滞时有良好效果，一旦这 2 种状态出现一种，生成的实验个体可能比父代个体在求解问题最优值上质量更差，导致算法无法向全局最优收敛。通过研究发现，当算法处于停滞或局部最优时，由于被遗弃的实验个体没有遗传到下一代，从遗传学原理上说这些个体更有可能帮助算法跳出“陷阱”，因此，这些被遗弃的个体对打破这种现状应该很有帮助。

基于这个事实，本文在算法的选择操作中设置一个大小为 $\frac{\mathrm{NP}}{2}$ 的外部存档 A 用于存储被遗弃的个体，存档中的所有个体根据适应度从小到大排序。如果实验个体在解的质量上劣于目标个体，算法不再遵循“贪心选择”策略，而随机从存档前 m 个个体中选择一个作为下一代新个体，若成功帮助算法“脱困”（对应的适应度优于目标个体），选择该个体作为下一代，否则，剔除该个体，并由存档中的第(m+1)个个体代替，使 m 的大小保持为设定的值，再从当前存档中选择最好的个体作为下一代。新型选择策略如算法 1 所示。

算法 1　新型选择策略
输入　目标个体 $\boldsymbol{x}_{i,g}$，实验个体 $\boldsymbol{u}_{i,g}$
输出　下一代个体 $\boldsymbol{x}_{i,g+1}$

1) 初始化参数设置，$A=\varnothing$，$|A|=\frac{\mathrm{NP}}{2}$，$m$=5；
2) 按函数适应度从小到大排序集合 A 中的个体；
3) for i=1 to NP do
4)　if $f(\boldsymbol{u}_{i,g}) < f(\boldsymbol{x}_{i,g})$ then
5)　　选择实验向量进入下一代，并将目标向量 $\boldsymbol{x}_{i,g}$ 加入外部存档 A；
6)　　if $|A|>\frac{\mathrm{NP}}{2}$ then
7)　　随机从 A 中选择一个个体将其剔除；
8)　　end if
9)　else
10)　　从存档 A 的前 m 个个体中随机选择一个个体 $\boldsymbol{a}$；
11)　　if $f(\boldsymbol{a}) < f(\boldsymbol{x}_{i,g})$ then
12)　　选择 $\boldsymbol{a}$ 进入下一代；
13)　　else
14)　　剔除 $\boldsymbol{a}$，用存档 A 中第(m+1)个个体代替 $\boldsymbol{a}$，并用集合 A 最好的个体作为下一代；
15)　　end if
16)　end if
17) end for

2.3 算法整体流程

NSPSDE 算法与标准 DE 算法的最大区别在于种群状态信息的判断及相应的处理方法、选择操作的执行。这些区别使 DE 具有处理停滞与局部最优的能力，能增强种群多样性。NSPSDE 算法如算法 2 所示。算法 2 在步骤 8)处按 DE/current-to-pbest/1 算子执行变异

$$\boldsymbol{v}_{i,g} = \boldsymbol{x}_{i,g} + F(\boldsymbol{x}_{\mathrm{pbest},g} - \boldsymbol{x}_{i,g}) + F(\boldsymbol{x}_{r1,g} - \boldsymbol{x}_{r2,g}) \tag{14}$$

其中，$\boldsymbol{v}_{i,g}$ 为变异个体，F 为缩放参数，$\boldsymbol{x}_{r1,g}$、$\boldsymbol{x}_{r2,g}$ 为不同于目标个体 $\boldsymbol{x}_{i,g}$ 的个体。设 FES 表示种群评价标量，p 表示当代个体按适应度从小到大排序后选择前 p%个最好的解，本文中 p=6。

算法 2　NSPSDE 算法
输入　当代种群 $\boldsymbol{X}_g$
输出　下一代种群 $\boldsymbol{X}_{g+1}$

1) 初始化参数：$\partial = 1\times10^{-2}$，$\varepsilon = 1\times10^{-3}$，$\mathrm{fail}=0$，$F_{\min}=0.2$，$F_{\max}=0.9$，$\mathrm{CR}_{\min}=0.2$，$\mathrm{CR}_{\max}=0.9$，FES=NP；
2) 在函数取值范围内随机初始化种群；
3) 评价个体的适应度；
4)　while FES<$\mathrm{FES}_{\max}$
5)　　for i=1 to NP do
6)　　找出当代前 p%最好的适应度对应个体 $\mathbf{pbest}_g$；
7)　　$F = F_{\max} - (F_{\max} - F_{\min})\left(\frac{g}{G_{\max}}\right)$，
$\mathrm{CR} = \mathrm{CR}_{\min} + (\mathrm{CR}_{\max} - \mathrm{CR}_{\min})^{\left(\frac{g}{G_{\max}}\right)^2}$；
8)　　根据式(14)执行变异，执行 DE 交叉操作；
9)　　执行算法 1；

10) end for
11) 根据式(2)和式(3)计算种群总体均值和标准差；
12) if $sd_g < \partial$ then
13) if 式(5)和式(6)为真 then
14) 利用 LBFGS 方法对各个个体进行随机学习，执行式(8)和式(9)；
15) else if fail > NP then
16) 执行式(13)；
17) fail 重新设置为 0；
18) end if
19) end if
20) if $abs(fbest_g - fbest_{g-1}) < \varepsilon$ then
21) fail=fail+1；
22) end if
23) g=g+1；
24) FES=FES+NP；
25) end while

3 实验与分析

3.1 测试函数

为了验证 NSPSDE 算法的有效性，本文利用29个标准函数[30]来测试其性能表现。这些函数包含了高维单峰函数（f_1～f_{13}）和高维多峰函数（f_{14}～f_{29}），它们拥有不可分离、可分离、旋转、可规模化、不可规模化、分步、噪声、复合型等重要特征。实验环境和配置如下：CPU-AMD Ryzen 6000，内存为32 GB，操作系统为Windows10 64位，编程语言为MATLAB 2020b。

3.2 算法比较与参数设置

本节将NSPSDE与8个现阶段先进的DE变种算法进行比较，这些DE变种算法分别是CODE[12]、SADE[11]、SinDE[14]、RankDE[13]、AEPDE-JADE[24]、MPEDE[31]、CSDE[32]、ADEDE[33]，其中，CSDE和ADEDE为近几年提出的算法，它们分别通过变异策略与自适应参数提高种群多样性，其余6个算法均为著名的DE变种算法。以上DE变种算法融入了不同的技术，参数的设置与算法原文献一致，它们都是经过反复调试解决相关问题得出的最好设置。NSPSDE 算法参数设置为 $\partial = 1\times10^{-2}$，$\varepsilon = 1\times10^{-3}$，$A$=∅，fail = 0，种群大小设置为NP=50，函数的最大评价次数设置为 FES_{Max}=5 000D，问题的维度设置为 D=30。为了公平起见，所有的算法在每个函数上独立执行 30 次，因此，它们之间存在的性能差异归因于各算法的搜索能力。在实验中，目标函数适应度的平均值和标准差被用来评价算法的性能，非参数统计测试（具有 5%显著水平的 Wilcoxon 符号秩检验）用来测试 NSPSDE 算法与 8 个 DE 变种算法之间的显著差异，实验结果如表 1 所示。其中，加粗部分表示最好的结果，符号“+”“−”“≈”分别表示对比算法的结果比 NSPSDE 算法“好”“差”“大约相等”。基于 Friedman 检验和 Wilcoxon’s rank-sum 检验的测试结果分别如表 2 和表 3 所示。

3.3 数值结果分析

从表 1 可以看到：1) 对于单峰函数（f_1～f_{13}），NSPSDE 在 f_1～f_8、f_{11} 处优于 8 个对比算法，获得了唯一的最优解，且性能偏差稳定（标准差），而在 f_9、f_{13} 处分别被 AEPDE-JADE、MPEDE 击败；NSPSDE 在 f_{13} 处劣于 MPEDE，这可能是 MPEDE 具有多种变异策略的缘故；在 f_{10} 处，除了 AEPDE-JADE 和 RankDE 在稳定性上稍差之外，其余算法均获得了最好解，所有算法在 f_{12} 处均获得了最好解；2) 对于多峰函数（f_{14}～f_{29}），NSPSDE 在 f_{15}～f_{22}、f_{25}～f_{29} 处获得了最优解；CSDE 与 AEPDE-JADE 在 f_{23} 处分别获得了最好的平均值（mean）与最好的标准差（std），而 AEPDE-JADE 和 NSPSDE 在 f_{24} 处分别获得了最好的平均值（mean）与最好的标准差（std），所有算法在 f_{14} 处都获得了最好解；在 f_{17} 处，除了 CSDE 之外，所有算法均获得了最好解，此外，注意到，在 f_{15}、f_{27}～f_{29} 处也有多个算法获得了最好解；3) 基于 Wilcoxon’s signed-rank test（Wilcoxon 符号秩检验）的结果表明，在 29 个函数中，CODE、SADE、SinDE、RankDE、AEPDE-JADE、MPEDE、CSDE、ADEDE 及 NSPSDE 获得的最优解数目分别为 4、6、6、3、8、9、5、8 和 25。进一步分析发现，NSPSDE 相对于 CODE、SADE、SinDE、RankDE、AEPDE-JADE、MPEDE、CSDE 及 ADEDE 在函数表现中胜出的百分比分别为 79.3%、72.4%、72.4%、79.3%、65.6%、65.6%、79.3%和 68.9%，相应胜出的数目分别为 23、21、21、23、19、19、23 和 20。这些实验结果进一步表明 NSPSDE 在优化能力上更具有优势。

表 2 为 NSPSDE 与 8 个 DE 变种算法在 D=30 结果的 Friedman 检验分析，其中，“等级”列是每个

表 1　　NSPSDE 与 8 个变种 DE 算法在 29 个测试函数比较结果

函数	量度	CODE		SADE		SinDE		RankDE		AEPDE-JADE		MPEDE		CSDE		ADEDE		NSPSDE
f_1	mean	2.84×10^{-30}	−	4.06×10^{-39}	−	4.36×10^{-58}	−	1.18×10^{-31}	−	4.16×10^{-163}	−	1.93×10^{-60}	−	2.01×10^{-58}	−	2.9×10^{-34}	−	**5.21×10^{-278}**
	std	2.39×10^{-30}		5.79×10^{-39}		2.40×10^{-58}		7.78×10^{-32}		0		4.21×10^{-60}		2.66×10^{-58}		1.65×10^{-34}		**0**
f_2	mean	8.77×10^{-27}	−	1.41×10^{-36}	−	1.30×10^{-55}	−	1.72×10^{-28}	−	2.03×10^{-159}	−	4.10×10^{-56}	−	2.42×10^{-57}	−	1.47×10^{-30}	−	**2.33×10^{-238}**
	std	6.62×10^{-27}		1.15×10^{-36}		1.05×10^{-55}		1.45×10^{-28}		4.54×10^{-159}		5.45×10^{-56}		3.33×10^{-57}		1.09×10^{-30}		**0**
f_3	mean	4.99×10^{-24}	−	5.53×10^{-34}	−	1.23×10^{-52}	−	4.13×10^{-25}	−	5.17×10^{-150}	−	1.63×10^{-54}	−	6.96×10^{-54}	−	2.65×10^{-28}	−	**2.45×10^{-279}**
	std	5.19×10^{-24}		4.66×10^{-34}		6.81×10^{-53}		7.41×10^{-25}		1.16×10^{-149}		2.04×10^{-54}		1.25×10^{-53}		1.45×10^{-28}		**0**
f_4	mean	3.19×10^{-7}	−	1.11×10^{-4}	−	6.65×10^{0}	−	5.85×10^{-7}	−	1.86×10^{-21}	−	3.19×10^{-39}	−	3.30×10^{-5}	−	8.74×10^{-2}	−	**1.21×10^{-200}**
	std	4.61×10^{-7}		1.16×10^{-4}		2.31×10^{0}		1.62×10^{-7}		4.16×10^{-21}		5.55×10^{-39}		3.17×10^{-5}		5.56×10^{-2}		**0**
f_5	mean	4.54×10^{-15}	−	2.76×10^{-21}	−	2.51×10^{-32}	−	9.07×10^{-15}	−	1.82×10^{-78}	−	4.79×10^{-27}	−	5.34×10^{-28}	−	8.37×10^{-20}	−	**7.00×10^{-149}**
	std	2.68×10^{-15}		1.66×10^{-21}		1.18×10^{-32}		5.10×10^{-15}		3.12×10^{-78}		4.71×10^{-27}		1.17×10^{-27}		2.35×10^{-20}		**3.79×10^{-148}**
f_6	mean	2.50×10^{-7}	−	2.69×10^{-4}	−	5.12×10^{-7}	−	2.39×10^{-4}	−	4.80×10^{-3}	−	7.27×10^{-23}	−	4.51×10^{-10}	−	4.50×10^{-2}	−	**8.97×10^{-137}**
	std	2.13×10^{-7}		5.43×10^{-4}		4.50×10^{-7}		1.59×10^{-4}		5.81×10^{-3}		8.64×10^{-23}		3.39×10^{-10}		9.65×10^{-2}		**7.91×10^{-137}**
f_7	mean	1.11×10^{-27}	−	1.82×10^{-38}	−	5.89×10^{-57}	−	3.39×10^{-30}	−	1.18×10^{-159}	−	1.41×10^{-57}	−	8.75×10^{-56}	−	4.64×10^{-33}	−	**1.80×10^{-277}**
	std	2.21×10^{-27}		3.31×10^{-38}		6.99×10^{-57}		2.68×10^{-30}		2.41×10^{-159}		2.69×10^{-57}		1.42×10^{-55}		2.96×10^{-33}		**0**
f_8	mean	8.37×10^{-30}	−	5.33×10^{-39}	−	7.63×10^{-59}	−	1.94×10^{-31}	−	3.33×10^{-164}	−	3.15×10^{-60}	−	1.90×10^{-56}	−	8.66×10^{-34}	−	**2.49×10^{-277}**
	std	7.58×10^{-30}		7.40×10^{-39}		5.24×10^{-59}		1.78×10^{-31}		0		4.33×10^{-60}		3.32×10^{-56}		5.95×10^{-34}		**0**
f_9	mean	5.03×10^{-26}	−	5.57×10^{-33}	−	1.12×10^{-52}	−	2.27×10^{-27}	−	**1.72×10^{-130}**	+	3.82×10^{-41}	−	2.41×10^{-46}	−	3.12×10^{-31}	−	7.40×10^{-97}
	std	6.28×10^{-26}		2.77×10^{-33}		3.94×10^{-53}		2.80×10^{-27}		**3.28×10^{-130}**		6.84×10^{-41}		4.53×10^{-46}		9.70×10^{-32}		2.07×10^{-98}
f_{10}	mean	**-1.00×10^{0}**	≈	**-1.00×10^{0}**	≈	**-1.00×10^{0}**	≈	-1.00×10^{0}	−	-1.00×10^{0}	−	**-1.00×10^{0}**	≈	**-1.00×10^{0}**	≈	**-1.00×10^{0}**	≈	**-1.00×10^{0}**
	std	**0**		**0**		**0**		1.57×10^{-16}		1.57×10^{-16}		**0**		**0**		**0**		**0**
f_{11}	mean	7.22×10^{-28}	−	9.36×10^{-34}	−	1.95×10^{-45}	−	4.86×10^{-29}	−	3.21×10^{-144}	−	7.08×10^{-63}	−	3.07×10^{-48}	−	2.49×10^{-26}	−	**1.72×10^{-253}**
	std	5.98×10^{-28}		1.07×10^{-33}		1.28×10^{-45}		1.84×10^{-29}		7.19×10^{-144}		1.42×10^{-62}		4.18×10^{-48}		2.10×10^{-26}		**0**
f_{12}	mean	**0**	≈	**0**	≈	**0**	≈	**0**	≈	**0**	≈	**0**	≈	**0**	≈	**0**	≈	**0**
	std	**0**		**0**		**0**		**0**		**0**		**0**		**0**		**0**		**0**
f_{13}	mean	6.16×10^{-3}	+	8.38×10^{-3}	+	5.12×10^{-3}	+	5.70×10^{-3}	+	4.81×10^{-3}	+	**1.25×10^{-3}**	+	3.34×10^{-2}	+	9.43×10^{-3}	+	6.21×10^{-2}
	std	2.18×10^{-3}		2.52×10^{-3}		8.26×10^{-4}		1.57×10^{-3}		1.72×10^{-3}		**3.24×10^{-4}**		6.23×10^{-3}		1.43×10^{-3}		2.32×10^{-2}
f_{14}	mean	**0**	≈	**0**	≈	**0**	≈	**0**	≈	**0**	≈	**0**	≈	**0**	≈	**0**	≈	**0**
	std	**0**		**0**		**0**		**0**		**0**		**0**		**0**		**0**		**0**
f_{15}	mean	2.04×10^{-6}	−	4.62×10^{-15}	−	9.94×10^{1}	−	1.20×10^{2}	−	**0**	≈	**0**	≈	**0**	≈	**0**	≈	**0**
	std	2.01×10^{-6}		6.36×10^{-15}		3.44×10^{0}		3.88×10^{1}		**0**		**0**		**0**		**0**		**0**
f_{16}	mean	4.23×10^{-3}	−	8.94×10^{-6}	−	9.81×10^{-3}	−	2.65×10^{-9}	−	4.44×10^{-17}	−	2.75×10^{-7}	−	1.44×10^{-7}	−	5.51×10^{-5}	−	**2.70×10^{-147}**
	std	2.03×10^{-3}		7.01×10^{-6}		4.33×10^{-3}		5.53×10^{-9}		9.93×10^{-17}		4.07×10^{-7}		2.72×10^{-7}		5.82×10^{-5}		**4.05×10^{-147}**
f_{17}	mean	**0**	≈	**0**	≈	**0**	≈	**0**	≈	**0**	≈	**0**	≈	2.22×10^{-17}	−	**0**	≈	**0**
	std	**0**		**0**		**0**		**0**		**0**		**0**		4.97×10^{-17}		**0**		**0**
f_{18}	mean	1.99×10^{-1}	−	1.60×10^{-1}	−	1.99×10^{-1}	−	1.99×10^{-1}	−	1.99×10^{-1}	−	1.19×10^{-1}	−	1.99×10^{-1}	−	1.99×10^{-1}	−	**1.06×10^{-133}**
	std	1.13×10^{-9}		5.47×10^{-2}		7.71×10^{-10}		3.68×10^{-9}		1.69×10^{-9}		4.47×10^{-2}		3.10×10^{-13}		1.36×10^{-10}		**4.94×10^{-134}**
f_{19}	mean	2.77×10^{0}	−	3.63×10^{0}	−	1.52×10^{1}	−	1.99×10^{1}	−	2.45×10^{-1}	−	3.59×10^{-1}	−	3.78×10^{-1}	−	4.29×10^{0}	−	**2.45×10^{-80}**
	std	1.57×10^{0}		6.48×10^{-1}		1.02×10^{0}		2.28×10^{0}		4.48×10^{-2}		1.71×10^{-1}		3.40×10^{-1}		9.99×10^{-1}		**0**
f_{20}	mean	5.51×10^{-15}	−	2.66×10^{-15}	−	3.38×10^{-15}	−	4.09×10^{-15}	−	6.22×10^{-15}	−	2.66×10^{-15}	−	2.04×10^{-14}	−	2.66×10^{-15}	−	**8.88×10^{-16}**
	std	1.59×10^{-15}		0		1.59×10^{-15}		1.95×10^{-15}		0		0		2.15×10^{-14}		0		**0**
f_{21}	mean	1.78×10^{0}	−	1.52×10^{1}	−	2.68×10^{1}	−	3.75×10^{1}	−	1.09×10^{-1}	−	2.02×10^{-1}	−	8.79×10^{0}	−	1.33×10^{1}	−	**0**
	Std	2.47×10^{-1}		4.05×10^{-1}		1.32×10^{0}		2.07×10^{0}		8.16×10^{-2}		4.91×10^{-2}		1.94×10^{0}		9.85×10^{-1}		**0**
f_{22}	mean	1.61×10^{-3}	−	1.11×10^{-1}	−	3.33×10^{-1}	−	1.55×10^{0}	−	2.65×10^{-5}	−	1.75×10^{-4}	−	3.09×10^{-2}	−	1.17×10^{-1}	−	**0**
	std	2.51×10^{-4}		1.24×10^{-2}		5.18×10^{-2}		2.14×10^{-1}		5.40×10^{-5}		7.55×10^{-5}		4.90×10^{-3}		2.58×10^{-2}		**0**
f_{23}	mean	3.15×10^{-1}	+	3.59×10^{-1}	+	3.54×10^{-1}	+	3.76×10^{-1}	+	2.37×10^{-1}	+	2.30×10^{-1}	+	**2.18×10^{-1}**	+	3.39×10^{-1}	+	6.82×10^{-1}
	std	6.92×10^{-2}		5.81×10^{-2}		3.43×10^{-2}		5.74×10^{-2}		**1.56×10^{-2}**		1.71×10^{-2}		2.90×10^{-2}		2.24×10^{-2}		5.74×10^{-1}
f_{24}	mean	2.77×10^{-1}	≈	3.64×10^{-1}	≈	3.13×10^{-1}	≈	3.80×10^{-1}	≈	**2.13×10^{-1}**	≈	3.62×10^{-1}	≈	4.39×10^{-1}	≈	3.50×10^{-1}	≈	5.00×10^{-1}
	std	5.31×10^{-2}		3.50×10^{-2}		5.07×10^{-2}		1.63×10^{-1}		4.73×10^{-2}		9.12×10^{-2}		1.67×10^{-1}		6.77×10^{-3}		**0**
f_{25}	mean	1.55×10^{0}	−	1.71×10^{0}	−	4.02×10^{0}	−	6.57×10^{0}	−	5.16×10^{-1}	−	7.57×10^{-1}	−	5.76×10^{-1}	−	2.09×10^{0}	−	**0**
	std	3.55×10^{-1}		2.24×10^{-1}		6.89×10^{-1}		2.84×10^{-1}		5.26×10^{-2}		6.37×10^{-2}		1.10×10^{-1}		2.81×10^{-1}		**0**
f_{26}	mean	1.25×10^{0}	−	1.71×10^{0}	−	4.21×10^{0}	−	6.57×10^{0}	−	5.16×10^{-1}	−	6.96×10^{-1}	−	4.47×10^{-1}	−	2.01×10^{0}	−	**0**
	std	3.16×10^{-1}		1.29×10^{-1}		4.03×10^{-1}		4.01×10^{-1}		5.26×10^{-2}		6.01×10^{-2}		1.09×10^{-1}		2.60×10^{-1}		**0**
f_{27}	mean	9.85×10^{0}	−	3.32×10^{-9}	−	5.84×10^{1}	−	9.98×10^{1}	−	**0**	≈	**0**	≈	4.60×10^{0}	−	**0**	≈	**0**
	std	4.07×10^{0}		3.99×10^{-9}		6.11×10^{0}		2.20×10^{1}		**0**		**0**		1.52×10^{0}		**0**		**0**
f_{28}	mean	1.32×10^{-31}	−	**1.57×10^{-32}**	≈	**1.57×10^{-32}**	≈	2.01×10^{-32}	−	**1.57×10^{-32}**	≈	**1.57×10^{-32}**	≈	4.38×10^{-31}	−	**1.57×10^{-32}**	≈	**1.57×10^{-32}**
	std	1.09×10^{-31}		**0**		**0**		8.44×10^{-33}		**0**		**0**		3.05×10^{-31}		**0**		**0**
f_{29}	mean	9.57×10^{-30}	−	**1.35×10^{-31}**	≈	**1.35×10^{-31}**	≈	4.03×10^{-31}	−	**1.35×10^{-31}**	≈	**1.35×10^{-31}**	≈	**1.35×10^{-31}**	≈	**1.35×10^{-31}**	≈	**1.35×10^{-31}**
	std	9.19×10^{-30}		**0**		**0**		2.20×10^{-31}		**0**		**0**		**0**		**0**		**0**
+、−、=的数目		2、23、4		2、21、6		2、21、6		2、23、4		3、19、7		2、19、8		2、23、4		2、20、7		—
最优解数目		4		6		6		3		8		9		5		8		25

算法在 29 个测试函数上独立运行 30 次排名的平均值，“排名”列是基于“等级”列各个算法的最终名次。从表 2 中可以看到，NSPSDE 在所有算法中获得了最好的等级与排名，相应的卡方统计量为 96.02，表明各个算法在性能上存在显著的差异。注：排名越小代表算法综合能力越好，卡方统计量大于 5.99 表示差异显著（$\alpha = 0.05$）。

表 2 NSPSDE 与 8 个 DE 变种算法在 *D*=30 结果的 Friedman 检验分析

算法	等级	排名
CODE	6.379 3	8
SADE	5.620 7	5
SinDE	5.672 4	6
RankDE	7.172 4	9
AEPDE-JADE	3.206 9	2
MPEDE	3.465 5	3
CSDE	4.982 8	4
ADEDE	6.034 5	7
NSPSDE	2.465 5	1

表 3 为 NSPSDE 与 8 个 DE 变种算法在 *D*=30 结果的 Wilcoxon’s rank-sum 检验分析。从表 3 中可以看到，NSPSDE 对比其他算法的正秩（R^+）分别为 383.0、376.5、383.5、388.0、347.0、342.0、354.0 和 369.0，相对应的负秩（R^-）分别为 52.0、58.5、51.5、47.0、88.0、93.0、52.0 和 66.0。根据统计学知识，R^+越大，代表所提算法 NSPSDE 的性能越好，且在 $\alpha = 0.05$ 处优势显著（用 Yes 表示）。因此，各算法的性能表现为 NSPSDE>MPEDE> AEPDE-JADE> CSDE>ADEDE>SinDE>SADE>> CODE>RankDE。

表 3 NSPSDE 与 8 个 DE 变种算法在 *D*=30 结果的 Wilcoxon’s rank-sum 检验分析

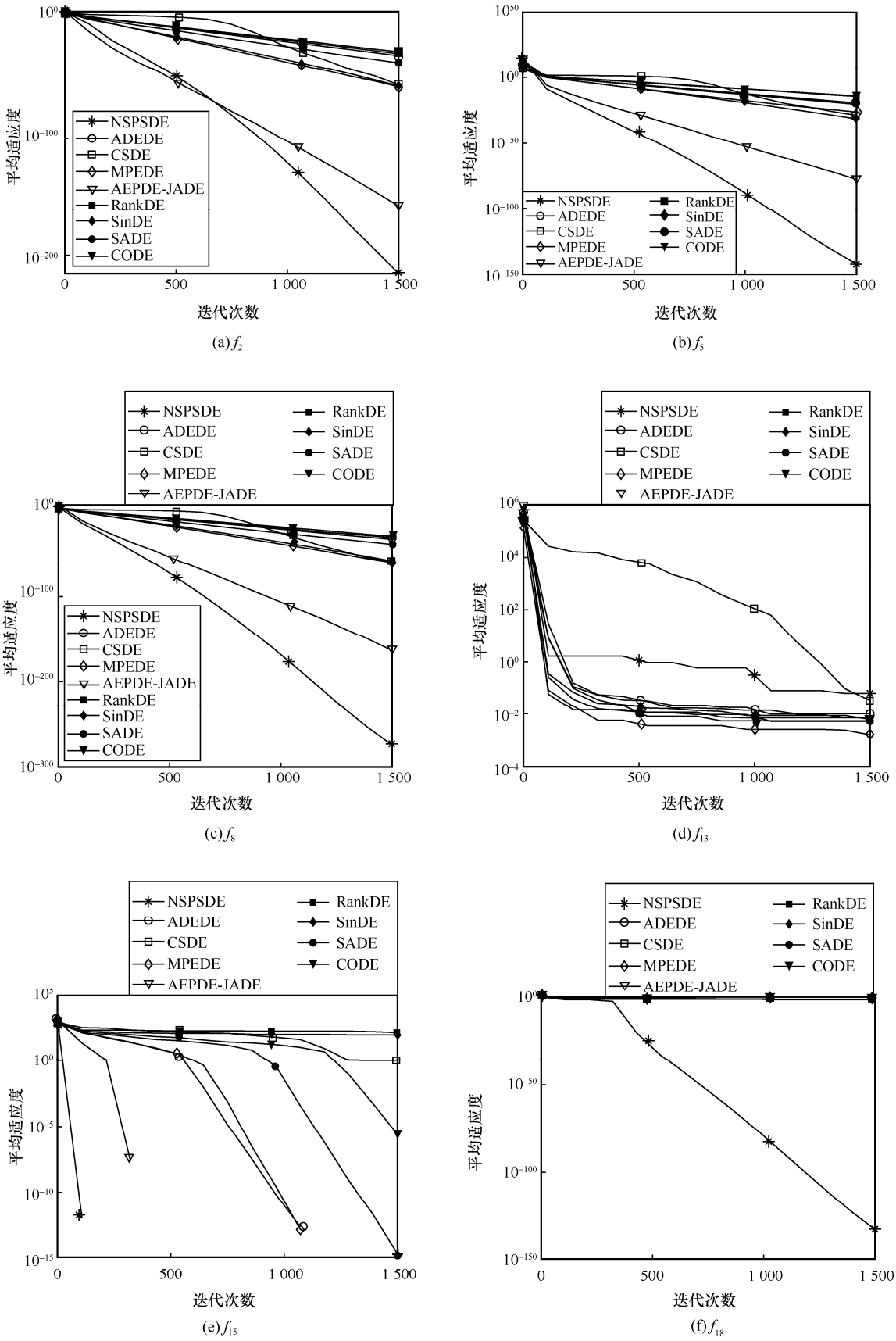
对比算法	R^+	R^-	p	$\alpha = 0.05$
CODE	383.0	52.0	1.39×10^{-4}	Yes
SADE	376.5	58.5	2.84×10^{-4}	Yes
SinDE	383.5	51.5	1.36×10^{-4}	Yes
RankDE	388.0	47.0	7.73×10^{-5}	Yes
AEPDE-JADE	347.0	88.0	4.10×10^{-3}	Yes
MPEDE	342.0	93.0	5.99×10^{-3}	Yes
CSDE	354.0	52.0	2.74×10^{-4}	Yes
ADEDE	369.0	66.0	6.07×10^{-4}	Yes

综上可知，通过与 8 个 DE 变种算法的实验数据比较，不论是基于 Wilcoxon’s rank-sum 检验分析还是基于 Friedman 检验分析，实验结果从统计学上进一步验证了 NSPSDE 的优越性。

3.4 收敛性分析

基于篇幅限制，本文在 29 个测试函数中选取 12 个具有代表性的函数来分析 NSPSDE 算法的收敛性，其中，前 4 个为单峰函数(f_2、f_5、f_8、f_{13})，后 8 个为多峰函数(f_{15}、f_{18}、f_{20}、f_{22}、f_{23}、f_{26}、f_{27}、f_{29})，特别地，f_{26} 为复合函数。NSPSDE 与 8 个 DE 变种算法在 *D*=30 处的收敛曲线如图 3 所示，其中，横坐标表示算法的迭代次数，纵坐标表示迭代过程中算法在当次迭代获得的平均适应度，为了清晰显示，纵坐标采用对数表示。

从单峰函数 f_2、f_5、f_8 收敛曲线可以看到，无论是初期还是后期，NSPSDE 的收敛速率都是最快的，其稳定性也是最好的，这是因为 NSPSDE 算法在搜索空间中拥有良好的全局与局部的开拓、利用能力，这也归结于新型的选择策略的结果。注意到，在 f_{13} 中，NSPSD 表现出缓慢的收敛趋势，尽管在最后的迭代（g=1 500）收敛性略劣于一些其他算法，而当 $g \approx 400$、650、950 种群出现了局部最优状态时，NSPSD 还是跳出了局部最优进行向前优化，这是由于算法的局部状态评估机制起到了良好的作用，但是由于迭代次数有限，NSPSD 在 f_{13} 未获得最好解。对于多峰函数，NSPSD 除了在 f_{23} 之外均获得最好的收敛性，具体分析，在处理 f_{15}、f_{22}、f_{26}、f_{27} 时，NSPSD 在前 300 次迭代均已经获得了全局最优解，其余算法即使迭代到最后（g=1 500）均未获得全局最优解，特别在处理复合函数 f_{26} 时，当其他算法一直处于停滞时，NSPSD 依然具有稳定的演化性能，表明 NSPSD 具有优化复杂多峰问题的出色能力。注意到，在 f_{18} 中所有算法在开始不久（大约在 g=100）均陷入了停滞，说明种群处于停滞状态，然而 NSPSDE 大约从 g=300 开始就已经跳出停滞，直到最后获得了非常接近全局最优解的值，其他算法直到最后均未跳出停滞，这表明 NSPSDE 算法的停滞机制起到了明显的作用。在优化 f_{20}、f_{23}、f_{29} 中，NSPSDE 也表现出良好结果，特别在 f_{20} 上获得了最快的收敛速度。注意到，在处理 f_{23} 时，尽管 NSPSDE 最后的表现不佳，但 NSPSDE 仍能继续前进，观察到 $g \approx 1\,400$ 之后收敛曲线仍呈下降趋势，相信如果增加迭代次数会有好的表现。

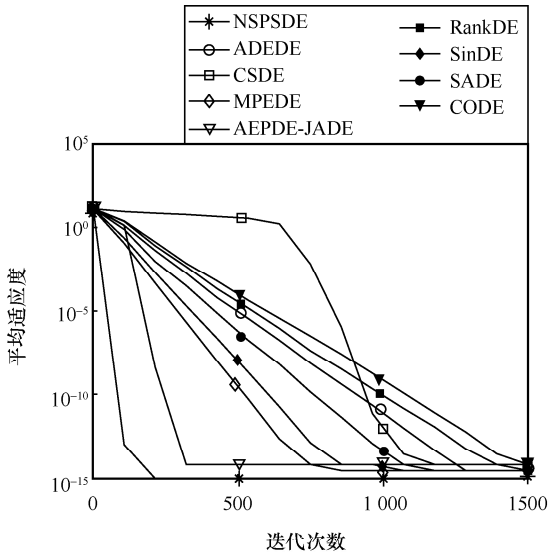


(a) f_2　(b) f_5

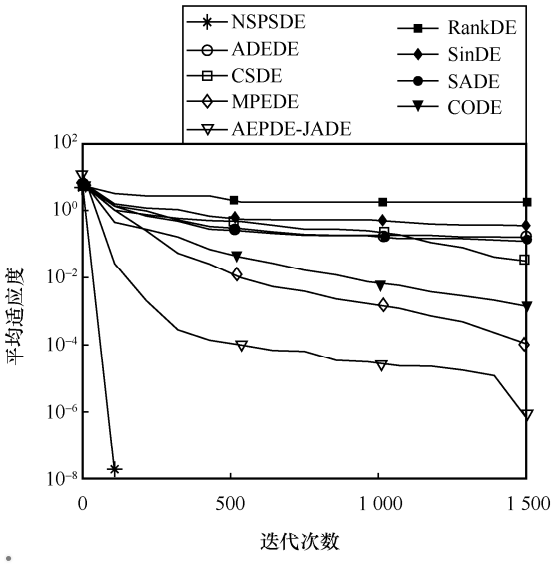
(c) f_8　(d) f_{13}

(e) f_{15}　(f) f_{18}

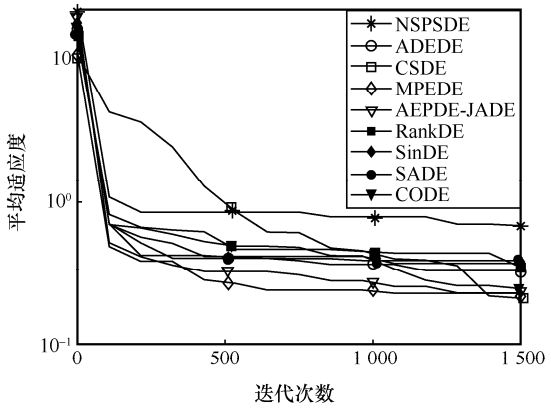
图 3　NSPSDE 与 8 个 DE 变种算法在 D=30 处的收敛曲线



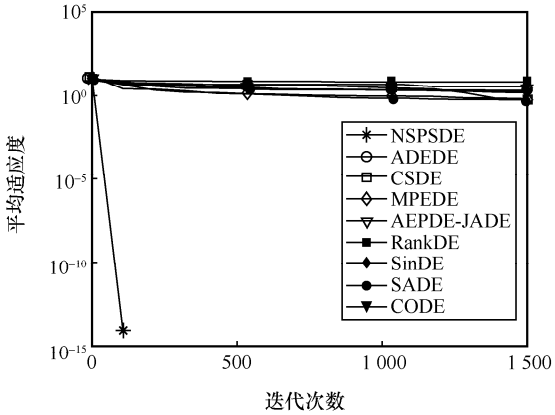
(g) f_{20}



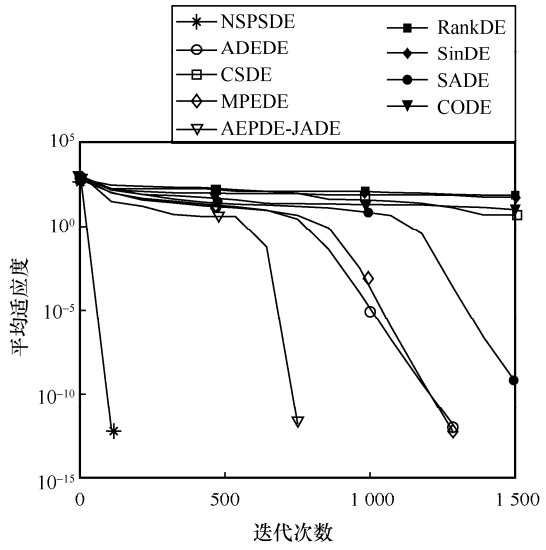
(h) f_{22}



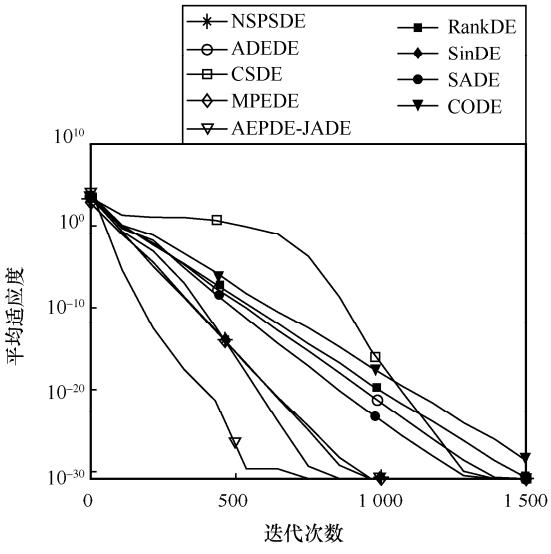
(i) f_{23}



(j) f_{26}



(k) f_{27}



(l) f_{29}

图 3　NSPSDE 与 8 个 DE 变种算法在 D=30 处的收敛曲线（续）

4　结束语

本文基于种群状态信息与 DE 的选择策略提出了一种改进的差分进化算法，该算法根据种群在演化中所处的不同状态采取不同的处理措施。针对局部最优，运用 LBFGS 方法与高斯变异探讨局部最优点附近的新个体，加强算法全局搜索性能。针对停滞状态，运用种群的协方差矩阵，基于旋转坐标系前后的映射对目标个体进行重组，抑制停滞。此外，本文针对 DE“贪心选择”策略缺点提出了一种新型的选择策略，该策略能更智能地针对种群状态信息进行不同的操作，增加算法的全局开发与局部利用能力。通过与现阶段 8 个 DE 变种算法在 29 个标准测试函数测试，实验结果中的数值结果分析与收敛性分析均证实了 NSPSDE 的有效性。

参考文献：

[1] 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.

[2] BREST J, GREINER S, BOSKOVIC B, et al. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems[J]. IEEE Transactions on Evolutionary Computation, 2006, 10(6): 646-657.

[3] ISLAM S M, DAS S, GHOSH S, et al. An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization[J]. IEEE Transactions on Systems, Man, and Cybernetics Part B, Cybernetics, 2012, 42(2): 482-500.

[4] ZHOU X G, ZHANG G J. Abstract convex underestimation assisted multistage differential evolution[J]. IEEE Transactions on Cybernetics, 2017, 47(9): 2730-2741.

[5] ZHOU S C, XING L N, ZHENG X, et al. A self-adaptive differential evolution algorithm for scheduling a single batch-processing machine with arbitrary job sizes and release times[J]. IEEE Transactions on Cybernetics, 2021, 51(3): 1430-1442.

[6] 王亚辉, 吴金妹, 贾晨辉. 基于动态种群多策略差分进化模型的多目标进化算法[J]. 电子学报, 2016, 44(6): 1472-1480.
WANG Y H, WU J M, JIA C H. Multi-objective evolutionary algorithm based on dynamic population multi-strategy differential models[J]. Acta Electronica Sinica, 2016, 44(6): 1472-1480.

[7] WANG B C, LI H X, LI J P, et al. Composite differential evolution for constrained evolutionary optimization[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2019, 49(7): 1482-1495.

[8] CAI Y Q, WU D W, ZHOU Y, et al. Self-organizing neighborhood-based differential evolution for global optimization[J]. Swarm and Evolutionary Computation, 2020, 56:100699.

[9] WANG X, MA S Q, GOLDFARB D, et al. Stochastic quasi-Newton methods for nonconvex stochastic optimization[J]. SIAM Journal on Optimization, 2017, 27(2): 927-956.

[10] DAS S, SUGANTHAN P N. Differential evolution: a survey of the state-of-the-art[J]. IEEE Transactions on Evolutionary Computation, 2011, 15(1): 4-31.

[11] QIN A K, HUANG V L, SUGANTHAN P N. Differential evolution algorithm with strategy adaptation for global numerical optimization[J]. IEEE Transactions on Evolutionary Computation, 2009, 13(2): 398-417.

[12] WANG Y, CAI Z X, ZHANG Q F. Differential evolution with composite trial vector generation strategies and control parameters[J]. IEEE Transactions on Evolutionary Computation, 2011, 15(1): 55-66.

[13] GONG W Y, CAI Z H. Differential evolution with ranking-based mutation operators[J]. IEEE Transactions on Cybernetics, 2013, 43(6): 2066-2081.

[14] DRAA A, BOUZOUBIA S, BOUKHALFA I. A sinusoidal differential evolution algorithm for numerical optimisation[J]. Applied Soft Computing, 2015, 27: 99-126.

[15] SHENG W G, WANG X, WANG Z D, et al. A differential evolution algorithm with adaptive niching and K-means operation for data clustering[J]. IEEE Transactions on Cybernetics, 2022, 52(7): 6181-6195.

[16] WEI Z F, GAO W F, LI G H, et al. A penalty-based differential evolution for multimodal optimization[J]. IEEE Transactions on Cybernetics, 2022, 52(7): 6024-6033.

[17] 陈宗淦, 詹志辉. 面向多峰优化问题的双层协同差分进化算法[J]. 计算机学报, 2021, 44(9): 1806-1823.
CHEN Z G, ZHAN Z H. Two-layer collaborative differential evolution algorithm for multimodal optimization problems[J]. Chinese Journal of Computers, 2021, 44(9): 1806-1823.

[18] 刘会宇, 韩继红, 袁霖, 等. 基于双变异策略的自适应骨架差分进化算法[J]. 通信学报, 2017, 38(8): 201-212.
LIU H Y, HAN J H, YUAN L, et al. Self-adaptive bare-bones differential evolution based on bi-mutation strategy[J]. Journal on Communications, 2017, 38(8): 201-212.

[19] ZHAN Z H, WANG Z J, JIN H, et al. Adaptive distributed differential evolution[J]. IEEE Transactions on Cybernetics, 2019, 50(11): 4633-4647.

[20] WANG Z J, ZHOU Y R, ZHANG J. Adaptive estimation distribution distributed differential evolution for multimodal optimization problems[J]. IEEE Transactions on Cybernetics, 2022, 52(7): 6059-6070.

[21] CHENG M Y, TRAN D H. Two-phase differential evolution for the multiobjective optimization of time–cost tradeoffs in resource-constrained construction projects[J]. IEEE Transactions on Engineering Management, 2014, 61(3): 450-461.

[22] 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[J]. 自动化学报, 2020, 46(4): 752-766.
WANG L J, ZHANG G J, ZHOU X G. Strategy self-adaptive differen-

tial evolution algorithm based on state estimation feedback[J]. Acta Automatica Sinica, 2020, 46(4): 752-766.

[23] ZHANG C J, QIN A K, SHEN W M, et al. ε-constrained differential evolution using an adaptive ε-level control method[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2022, 52(2): 769-785.

[24] YANG M, LI C H, CAI Z H, et al. Differential evolution with auto-enhanced population diversity[J]. IEEE Transactions on Cybernetics, 2015, 45(2): 302-315.

[25] ZHAO F Q, ZHAO L X, WANG L, et al. An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion[J]. Expert Systems with Applications, 2020, 160: 113678.

[26] GUO S M, YANG C C, HSU P H, et al. Improving differential evolution with a successful-parent-selecting framework[J]. IEEE Transactions on Evolutionary Computation, 2015, 19(5): 717-730.

[27] ZENG Z Q, ZHANG M, CHEN T, et al. A new selection operator for differential evolution algorithm[J]. Knowledge-Based Systems, 2021, 226: 107150.

[28] PULIDO G T, LANDA R, LÁRRAGA G, et al. On the use of stochastic ranking for parent selection in differential evolution for constrained optimization[J]. Soft Computing, 2017, 21(16): 4617-4633.

[29] ZENG Z Q, ZHANG M, HONG Z Y, et al. Enhancing differential evolution with a target vector replacement strategy[J]. Computer Standards & Interfaces, 2022, 82:103631.

[30] ZHONG X X, CHENG P. An elite-guided hierarchical differential evolution algorithm[J]. Applied Intelligence, 2021, 51(7): 4962-4983.

[31] WU G H, MALLIPEDDI R, SUGANTHAN P N, et al. Differential evolution with multi-population based ensemble of mutation strategies[J]. Information Sciences, 2016, 329: 329-345.

[32] SUN G J, YANG B, YANG Z Q, et al. An adaptive differential evolution with combined strategy for global numerical optimization[J]. Soft Computing, 2020, 24(9): 6277-6296.

[33] CUI L Z, LI G H, ZHU Z X, et al. A novel differential evolution algorithm with a self-adaptation parameter control method by differential evolution[J]. Soft Computing, 2018, 22(18): 6171-6190.

[作者简介]

麦伟杰（1985− ），男，广东广州人，华南理工大学博士生，主要研究方向为演化计算、机器学习。

刘伟莉（1985− ），女，广东广州人，博士，广东技术师范大学讲师，主要研究方向为演化计算、群体智能。

钟竞辉（1982− ），男，广东广州人，博士，华南理工大学教授，主要研究方向为演化计算、机器学习及多智能体建模与仿真。