软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2023,34(4):1850−1869 [doi: 10.13328/j.cnki.jos.006432]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于密度峰值的依维度重置多种群粒子群算法*

陶新民, 郭文杰, 李向可, 陈 玮, 吴永康

(东北林业大学 工程技术学院, 黑龙江 哈尔滨 150040)

通信作者: 郭文杰, E-mail: clockworkor@outlook.com

摘 要: 针对粒子群算法无法有效兼顾开采与勘探的问题, 提出一种基于密度峰值的依维度重置多种群粒子群算法. 首先采用密度峰值聚类中相对距离的思想并结合适应度值将种群分为两个子种群: 顶层群和底层群. 之后为顶层群设计专注于开采的学习策略而为底层群设计倾向于勘探的学习策略, 以均衡种群的勘探与开采. 最后依维度将陷入局部最优的粒子与全局最优粒子交叉重置, 在有效避免早熟收敛的同时也显著减少了无效计算次数. 将提出的算法与其他改进的优化算法在基础优化问题与 CEC2017 测试集上进行实验对比, 实验结果均值的统计检验证明了提出算法的改进具有统计学显著性.

关键词: 粒子群算法; 密度峰值聚类; 多种群; 依维度重置

中图法分类号: TP18

中文引用格式: 陶新民, 郭文杰, 李向可, 陈玮, 吴永康. 基于密度峰值的依维度重置多种群粒子群算法. 软件学报, 2023, 34(4): 1850–1869. http://www.jos.org.cn/1000-9825/6432.htm

英文引用格式: Tao XM, Guo WJ, Li XK, Chen W, Wu YK. Density Peak Based Multi Subpopulation Particle Swarm Optimization with Dimensionally Reset Strategy. Ruan Jian Xue Bao/Journal of Software, 2023, 34(4): 1850–1869 (in Chinese). http://www.jos.org.cn/1000-9825/6432.htm

Density Peak Based Multi Subpopulation Particle Swarm Optimization with Dimensionally Reset Strategy

TAO Xin-Min, GUO Wen-Jie, LI Xiang-Ke, CHEN Wei, WU Yong-Kang

(College of Engineering and Technology, Northeast Forestry University, Harbin 150040, China)

Abstract: In order to solve the dilemma that particle swarm optimization (PSO) cannot well balance the exploration and exploitation, a density peak based multi subpopulation particle swarm optimization algorithm is proposed with dimensionally reset strategy (DPMPSO). In the proposed DPMPSO, the idea of relative distance originated from density peak clustering is firstly adopted and then it is combined with the fitness value of particles to divide the whole swarm into two subpopulations: the top subpopulation and the bottom subpopulation. Secondly, the learning strategy is designed, focusing on local search for the top subpopulation and the learning strategy paying more attention to global search for the bottom subpopulation, which can well balance the exploration and exploitation. Finally, particles that fall into local optima will be reset by crossover with the global optima dimensionally, which can not only effectively avoid premature but also significantly reduce invalid iteration. The experiment results on 10 benchmark problems and CEC2017 optimization problems demonstrate that DPMPSO performs better than some representative PSOs and other optimization algorithms with significant difference.

Key words: particle swarm optimization (PSO); density peak clustering; multi subpopulation; dimensionally reset

粒子群算法 (particle swarm optimization, PSO) 是由 Kennedy 和 Eberhart[1]提出的一种基于群体智能的优化算法, 由于其具有较快的收敛速度且在实际问题求解中易于实现, 现已被广泛应用在工程优化[2,3], 云计算[4,5]以及模

* 基金项目: 国家自然基金面上项目 (62176050); 中央高校基本科研业务费专项资金 (2572017EB02); 东北林业大学双一流科研启动基金 (411112438); 哈尔滨市科技局创新人才基金 (2017RAXXJ018)
收稿时间: 2020-10-10; 修改时间: 2021-04-13, 2021-07-10; 采用时间: 2021-08-13; jos 在线出版时间: 2022-09-30
CNKI 网络首发时间: 2022-11-15

© 中国科学院软件研究所 http://www.jos.org.cn

式识别[6,7]等科学领域. 然而, 与差分进化算法[8]和人工蜂群算法[9]等大多数群智能优化算法一样, 粒子群算法的优化结果很大程度上取决于其是否能在优化问题的解空间中实现勘探与开采的平衡. 实验表明, 传统的粒子群算法在处理部分优化问题, 尤其是高维复杂优化问题时常常无法兼顾勘探与开采, 从而导致早熟收敛或收敛精度较低等问题[10].

针对这一问题, 研究者对传统粒子群算法进行了多方面的改进, 主要包括以下几个方向.

(1) 参数的调整: Chen 等人[11]在传统粒子群算法中引入正余弦加速因子, 并通过正弦映射来调整加速系数以实现搜索的平衡. 而 Tian 等人[12]通过构造一个类 Sigmoid 的惯性权重, 使算法自适应地利用线性和非线性递减策略之间的惯性权重来平衡勘探与开采. 然而实验证明单独的参数调整实际上很难保证粒子具有较强的开采能力[13].

(2) 学习策略的改进: Liang 等人[14]提出了全面学习的粒子群算法 CLPSO (comprehensive learning particle swarm optimization), 该算法通过对整个种群全面学习的随机策略实现了较强的全局勘探能力, 但是文中的实验表明由于其损失了局部开采能力, CLPSO 在处理单峰问题时效果不理想. Zhang 等人[15]则设计了自适应学习策略, 该策略通过进化过程中粒子位置来自适应分配学习策略, 取得了不错的效果.

(3) 拓扑结构的设计: Ning 等人[16]设计了一种可以快速发现更多潜在最优解的拓扑结构, 适用于处理多峰问题. 而 Li 等人[17]将动态子群与多维综合学习策略相结合, 使得算法收敛速度更快. 遗憾的是, 该算法在提高收敛性能的同时, 全局搜索能力有所减弱. 基于同样的思想, 文献 [18] 在传统 CLPSO 中加入多种群拓扑与改进学习方法的概念, 在单模态问题上优于传统 CLPSO 算法.

近年来, 诸多学者尝试着将上述 3 个方向相结合并相继提出一些改进的粒子群算法, 部分算法取得了较好的优化效果. 如文献 [19] 在经典遗传学习粒子群算法的基础上, 提出一种新的环形拓扑结构取代传统全局拓扑结构用以提高整个种群的多样性. 此外, 算法还将线性参数调节的全局学习策略引入到算法中实现自适应搜索, 用以平衡搜索过程中的勘探与开采性能. 文献 [20] 将上述基于改进学习策略的 CLPSO 算法与多种群拓扑相结合, 并引入一种基于全局最优粒子的学习策略用以提高传统 CLPSO 算法的局部开采能力. 实验结果表明其优化效果明显优于 CLPSO 算法, 尤其在处理单模态优化问题时. 文献 [21] 将拓扑结构与学习策略相结合, 提出了一种基于局部最优拓扑的全面学习粒子群算法, 该算法能够在进化过程中动态地调整拓扑空间, 从而在不影响算法收敛精度的前提下提升了算法的收敛速度.

除上述算法之外, 为了兼顾算法的勘探和开采能力, 大部分粒子群的改进算法都会以多种群的拓扑结构作为基础, 并通过为不同种群的粒子分配不同学习策略的方式来实现两者的均衡. 然而文献 [22] 指出, 不适合的种群划分策略往往在改进算法的同时破坏了其原有的拓扑结构, 导致种群多样性降低进而陷入早熟收敛等问题. 而导致这个问题的原因主要有两个.

(1) 不恰当的分群策略忽略了每个子群本身在搜索空间的分布信息. 如目前大部分多种群算法[17,18]都采用随机策略或依据适应度值策略划分种群. 随机策略虽然保证了子群粒子的多样性, 但该策略的针对性较弱, 并没有将不同适应度的粒子用于不同的搜索任务而降低了算法的收敛性能; 相比较而言, 依据适应值策略虽然可以利用较优粒子识别种群中潜在的局部最优区域并加以开采, 但这却会导致多种群中适应度较高的用于开采最优解的子群在初始化的时候就密集分布在某一个或几个最优区域, 从而在迭代过程中很容易陷入早熟收敛且无法实现对其他最优区域的开采.

(2) 忽略了种群间的信息交互从而导致每个子种群多样性降低与勘探能力减弱. 将规模较大的种群划分为多个小规模的子群, 总会或多或少的降低每个子群的多样性[22]. 而现存多数多种群算法并没有设置合适的信息交互策略以弥补种群多样性的损失, 最终导致了子种群易于陷入局部最优.

针对以上问题, 本文结合 3 个基本的改进方向, 设计了一种基于密度峰值的依维度重置多种群粒子群算法 (DPMPSO). 首先, DPMPSO 借用密度峰值聚类[23]中相对距离的思想, 以其定义的相对距离与粒子适应度值作为分群依据, 将种群分为初始时顶层粒子较少且底层粒子较多的两个子种群, 保证进化过程中每个子群粒子的适应度值处于相同水平且均匀分散在空间的不同位置, 加强子种群的多样性以提升每个子种群的勘探能力. 在此基础上对不同层次的粒子采取不同的学习策略. 其中顶层专注于开采最优解以提高解的精度, 底层则用于勘探新的最优

© 中国科学院软件研究所 http://www.jos.org.cn

区域以防止早熟收敛. 此外, 随着迭代次数的增加底层粒子逐渐向顶层流动, 使得算法前期倾向于全局的勘探, 而后期专注于局部的开采, 同时实现了种群之间的信息流动. 最后, 算法为每个粒子定义线性递减的维度概率, 并依此概率将每个种群中适应度最差且多代不更新的停滞粒子与全局最优解交叉重置, 由于全局最优解可能处于顶层或底层群, 交叉重置策略有助于两个种群间的信息交互, 进而增强种群的多样性. 在 10 个基准优化函数与 CEC2017 (congress on evolutionary computation 2017) 测试集的测试环境下将 DPMPSO 与其他优化算法进行实验对比, 并通过统计检验对结果进行检验证明了 DPMPSO 的改进具有显著性.

1 粒子群算法

作为一种群智能优化算法, 粒子群算法 (PSO) 通过模拟鸟群觅食的过程来求解优化问题. 求解过程中, 粒子群算法将每个粒子作为一个潜在解且每个粒子由两个重要成分组成, 即速度与位置. 在进化过程中, 每个粒子的速度根据其自身的历史最优以及全局的历史最优进行更新, 针对 D 维问题其具体形式如下:

$$V_i^d = w \times V_i^d + c_1 \times rand1_i^d \times \left(pbest_i^d - X_i^d\right) + c_2 \times rand2_i^d \times \left(gbest^d - X_i^d\right) \tag{1}$$

$$X_i^d = X_i^d + V_i^d (d = 1, 2, \ldots, D) \tag{2}$$

其中, V_i^d, X_i^d 分别代表第 i 个粒子第 d 维的速度值与位置. $pbest_i^d, gbest^d$ 分别代表第 i 个粒子的第 d 维历史最优值与整个种群的第 d 维历史最优值. w 代表惯性权重, 用于调节算法在勘探与开采间的平衡. c_1, c_2 是两个加速因子, 用于控制粒子向自身最优和全局最优移动的幅度. $rand1_i^d$ 和 $rand2_i^d$ 是从 [0, 1] 之间产生的随机数. 同时算法定义速度边界 $V_{\max}$, 每当粒子的速度 V_i^d 超过边界时则重置, 公式如下:

$$V_i^d = sign\left(V_i^d\right) \times V_{\max} \tag{3}$$

2 基于密度峰值聚类的动态多种群粒子群算法

本文设计一种基于密度峰值的依维度重置多种群粒子群算法. 该算法改进主要包括以下几点.

(1) 基于密度峰值的分群策略将种群分为顶层群和底层群, 且随着迭代次数的增加通过再分群将底层的粒子逐渐流向顶层; (2) 为每个子群设计不同的学习策略以均衡开采与勘探的任务; (3) 采用依维度重置策略, 将多代不更新且适应度差的粒子依维度概率与全局最优交叉重置.

第 2.1 节首先介绍基于密度峰值的动态分群策略. 第 2.2 节在分群策略的基础上介绍了针对不同分群的不同学习进化策略. 之后在第 2.3 节中引入进化过程中的依维度重置策略. 最后将上述策略结合并给出算法的整体流程. 具体分析如下.

2.1 基于密度峰值聚类的动态多种群策略

如上文所述, 大多数多种群算法会单独依据粒子的适应度值来划分种群, 并在此基础上为不同子种群分配不同任务, 实际上这种分群策略会导致每个子群中的粒子位置分布不均且适应值处于不同水平, 进而易陷入早熟收敛. 为了显式地说明问题, 我们以 Schwefels 函数最小优化问题为例, 如图 1(a) 所示 (图 1 中紫色、蓝色、绿色与橙色的等高线依次代表了适应度由低到高的分布情况), 多个适应度值较低的粒子 (蓝色) 会集中在同一局部最优的范围中而导致单一的在该范围内进行开采, 这就会使得该种群在迭代过程中迅速收敛至该区域而忽略其他的峰值. 同时由于这些区域的粒子均被划分为适应度较低 (蓝色) 的群, 使得适应度较高的粒子 (绿色) 无法遍布整个搜索空间而使其勘探能力减弱.

针对这一问题, 本文引入密度峰值聚类的思想, 该方法定义如下: 通过数据点的坐标计算其局部密度, 再通过局部密度针对数据点进行排序, 依次计算出数据点的相对距离, 该距离一般使用欧氏距离衡量. 最后通过局部密度与相对距离绘制决策图来确定聚类中心. 而本文借助其相对距离的概念, 以粒子的适应度与欧氏距离作为计算其相对距离的依据, 并将粒子适应度与相对距离进行排序, 最终将每个粒子的两者排序的乘积作为分种群的依据, 以最小优化问题为例, 其具体公式如下:

© 中国科学院软件研究所 http://www.jos.org.cn

$$f_{q_1} \leqslant f_{q_2} \leqslant \ldots \leqslant f_{q_m} \tag{4}$$

$$Rank_{\text{fit}} = [Rank_{\text{fit}}(i) = j | q_j = i] \tag{5}$$

$$\delta_{q_i} = \begin{cases} \min\limits_{q_j: j<i} \left(d_{q_i q_j}\right), & i \geqslant 2 \\ \max\limits_{j \geqslant 2} (\delta_{q_j}), & i = 1 \end{cases} \tag{6}$$

$$\delta_{p_1} \geqslant \delta_{p_2} \geqslant \ldots \geqslant \delta_{p_m} \tag{7}$$

$$Rank_{\text{dis}} = [Rank_{\text{dis}}(i) = j | p_j = i] \tag{8}$$

$$Div_i = Rank_{\text{fit}_i} \times Rank_{\text{dis}_i} \tag{9}$$

其中, f_{q_i} 代表了粒子 q_i 的适应度值, $Rank_{\text{fit}}$ 与 $Rank_{\text{dis}}$ 可理解如下: 在公式 (4) 将所有粒子依据适应度 f_{q_i} 排列后, 提取其排序后标号 q_i 的 i 值作为每个粒子的排序号, 如第一个粒子的适应度 f_1 在种群中排序第 3, 即 q_3=1, 那么在 $Rank_{\text{fit}}$ 中将 3 放在第一个位置. 同理, 如第一个粒子的相对距离排序第 2, 即 p_2=1, 那么则将 2 放在 $Rank_{\text{dis}}$ 的第一位置. Div_i 是两者的乘积结果, 也是粒子 i 分群时的依据, 即对每个粒子而言, 取其在公式 (5) 与公式 (8) 中的两个排序结果的乘积作为下一步分群时的依据. 我们在 $Rank_{\text{fit}}$ 中将适应度值较低的粒子排序靠前, 而在 $Rank_{\text{dis}}$ 中将相对距离较大粒子的排序靠前, 以实现选取的用于开采的粒子既有较好的适应度值, 又互相保持一定的距离. 这样直接对排序值进行处理, 省去了对适应度以及相对距离先进行标准化再统一指标的步骤. 此外, 在解决最大化问题时, 只需要对适应度值排序进行处理即可. 基于密度峰值划分种群的伪代码如算法 1 所示.

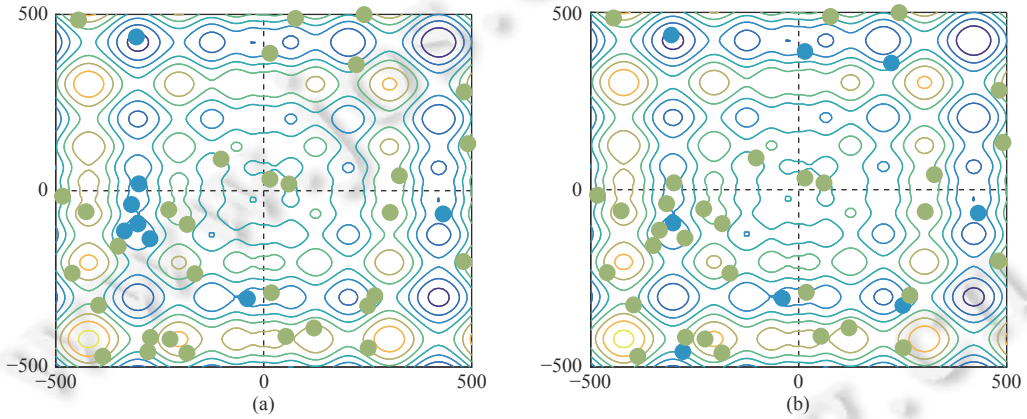

图 1 不同策略分种群结果

算法 1. 基于密度峰值划分种群策略.

输入: 种群 X, 顶层群规模 N_u;

输出: 顶层群 $X_{<1>}$和底层群 $X_{<2>}$.

1. 依输入的种群 X, 计算出粒子的适应度值 f.
2. 根据公式 (4)、公式 (5) 对适应度值升序排列.
3. 根据公式 (6)–公式 (8) 计算相对距离, 并降序排列.
4. 根据公式 (9) 计算 Div_i, 并将其升序排序, 将前 N_u 的粒子划分到顶层种群 $X_{<1>}$, 其余划分至底层 $X_{<2>}$.

值得注意的是, 该策略在划分种群时, 由公式 (6) 可见, 在相对距离的计算过程中, 如果粒子的适应度最优, 那么分配其相对距离亦最大, 即排序最靠前, 借此保证适应度最优的粒子分配在顶层. 而对于其后的粒子, 其相对距离定义为其与适应度优于该粒子且距离该粒子最近的粒子间的欧式距离. 具体如下: 其首先对适应度排序 2 – n 的粒子进行处理, 即排序为 2 的粒子的相对距离为其与排序为 1 的粒子的欧氏距离, 而排序为 3 的粒子的相对距离则为其与排序为 1、2 的粒子之间的欧氏距离的最小值, 以此类推. 遍历完 2 – n 后, 为排序为 1 粒子的相对距离赋

© 中国科学院软件研究所 http://www.jos.org.cn

予其他所有粒子的相对距离 δq_i 中的最大值. 这种计算方式不仅保证了顶层粒子的适应度值较好, 而且选取的顶层粒子由于受相对距离的影响会相对处于不同的局部最优解的范围内. 即粒子 i 具有较好的适应度 (在适应度升序排序中靠前), 且与优于它的粒子间的相对距离又较大 (在相对距离升序排序中靠前) 才有机会进入到顶层群中参与进化. 这样就可以最大程度地保证分配到顶层群的粒子既具有较好的适应度, 同时又保证粒子间有一定的分散性, 对于底层群而言同样如此. 如图 1(b) 所示, 对比图 1(a), 蓝色顶层粒子在整个搜索空间中找到了更多的局部最优区域以供开采, 同时也为绿色的底层粒子留下了分布更广泛的种群以进行全局的勘探. 需要解释的是, 我们将种群中的每个粒子的适应度值代替密度峰值聚类算法中的局部密度以进一步求解相对距离, 实际上就是考虑到在聚类过程中, 算法对局部密度的描述潜在意义上是希望找到周围具有较多点的数据点, 因为这类点本身具有较大的概率位于数据集的聚类中心, 从而保证在此基础上计算的相对距离能较好地识别处于不同区域的聚类中心. 因此, 我们使用粒子的适应度值直接代替局部密度, 实际上就是希望找出种群中具有较大概率靠近局部最优解的个体, 并在此基础上进一步计算个体间的相对距离, 这就保证了处于不同局部最优区域且适应度较好个体 (类似于不同的聚类中心) 能够被有效识别出来, 从而使得进入顶层群的粒子适应度较优且彼此间分散.

以上提出的基于密度峰值的分种群策略保证了进化过程中每个子群粒子的适应度值在搜索空间中处于相同水平且均匀分散在空间的不同位置, 从而加强子种群的多样性以提升每个子种群的勘探能力. 然而, 考虑到优化算法在搜索过程中的最佳收敛方式, 即在搜索过程的前期算法应该更多地倾向于全局的勘探, 而后期则更应该专注于局部的开采. 本文设计了种群间的动态流动策略, 即在最初只有 N/S (S 为流动系数, N 为整个种群规模) 顶层粒子的基础上, 在计算次数超过总计算次数的一半时, 每过固定的计算次数就将整体种群重新进行分群操作, 且每次重新分群时顶层群粒子个数增加 N/S, 直到最终底层群粒子占整体种群粒子的 N/S 为止, 即通过 S–2 次重新分群实现粒子的流动与子群间信息交互.

同时, 考虑到在一段迭代后每个子种群中的个体在搜索空间中所处的相对位置会发生变化, 因此选择使用再分群策略实现种群的流动, 这样不仅保证了整个种群在算法后期有更多的个体用于开采, 同时又能维护每个子种群的多样性. 种群流动策略的具体方式如算法 2 所示.

算法 2. 种群流动策略.

输入: 顶层群 X_u 和底层群 X_l, 整体种群规模 N, 计算次数 fes, 计算总次数 $Maxfes$, 流动系数 S, 缺省为 5;
输出: 顶层群 X_u 和底层群 X_l.

1. **if** $Fes \geqslant Maxfes/2$ & 底层种群规模 > N/S
2. **if** mod(Fes, $Maxfes/(2\times S)$)==0)
3. $N_u = N_u + N/S$
4. 依据算法 1 重新分群
5. **end if**
6. **end if**

2.2 改进的全面学习策略

有效的分群策略可以保证多个子种群具备单独完成搜索任务的条件, 而为了确保在单独搜索中每个子种群能够实现不同的搜索任务, 就必须为其提供有针对性的学习策略. 具体地说, 对于用于勘探全局的底层群而言需要注重对整个空间的搜索, 增强种群多样性; 而对于用于局部开采的顶层群而言, 其学习策略则需要关注对局部空间的挖掘, 提升收敛性能. 鉴于此, 本文采用两种不同的全面学习策略分别分配给顶层与底层群. 全面学习粒子群算法 (CLPSO) 是由 Liang 等人[14]提出的一种适用于多峰优化问题的优化算法, 公式如下:

$$P_{ci} = a + b \times \frac{\left(\exp\left(\frac{10(i-1)}{N-1}\right)-1\right)}{(\exp(10)-1)} \tag{10}$$

© 中国科学院软件研究所 http://www.jos.org.cn

$$V_i^d = w \times V_i^d + c \times rand_i^d \times \left(pbest_{fi(d)}^d - X_i^d\right) \tag{11}$$

其中, P_{ci} 代表了粒子 i 的学习概率, 根据文献 [14] 的建议学习概率的两个控制参数设置为: a=0.05 , b=0.45 以保证每个粒子的交叉概率在大于 0.05 的基础上各不相同且线性递减, N 为种群规模, $pbest_{fi(d)}^d$ 即为改进后的全面学习个体. 全面学习策略的目标更新如算法 3 所示.

算法 3. 全面学习目标更新策略.

输入: 种群 X, 适应度值 f, 种群规模 N, 问题维度 D;
输出: 学习目标 $pbest_f$.

1. 按公式 (10) 对种群每个粒子分配线性递增的学习概率 p_{ci}.
2. **for** i = 1:N
3. **for** d=1:D
4. **if** *rand* (0, 1) < P_{ci}
5. 种群中随机选取两个粒子 ra, rb
6. **if** $f(pbest_{ra})$< $f(pbest_{rb})$
7. $pbest_{fi(d)}^d = pbest_{ra}^d$
8. **else**
9. $pbest_{fi(d)}^d = pbest_{rb}^d$
10. **end if**
11. **else**
12. $pbest_{fi(d)}^d = pbest_i^d$
13. **end if**
14. **end for**
15. **if** $pbest_{fi} == pbest_i$
16. $pbest_{fi}^R = pbest_i^R$, R=*round*(*rand*×D)
17. **end if**
18. **end for**

通过伪代码可以看出, 全面学习策略就是在经典粒子群算法速度更新公式的基础上, 舍弃了全局最优项 $gbest^d$ 的影响, 取而代之的是通过在种群中随机选取的两个粒子 ra, rb 中适应度较好的粒子作为候选点, 再依据参数 P_{ci} 将候选点与粒子自身的历史最优解交叉组合从而得到全面学习算子 $pbest_{fi(d)}^d$ 来进行速度更新, 从而保证其全面学习的能力提升粒子群间的分散性. 其中文献 [14] 中 P_{ci} 定义为根据种群个体差异而线性递减的参数以保证每个粒子的交叉概率不同. 上文提到, 文献 [14] 中的实验结果证明全面学习策略具有较好的全局勘探能力且在处理多峰问题时表现良好, 但其随机性较强且局部开采能力较弱, 导致其在面对复杂问题时无法取得较高的最优解精度. 为此, 本文将经典的全面学习策略用于底层群实现全局勘探, 而将增加全局最优项的改进全面学习策略用于顶层群实现局部开采, 这样既可以提升种群的全局勘探能力又能有效兼顾其开采性能. 公式如下:

$$V_i^d = w \times V_i^d + c_1 \times rand1_i^d \times \left(pbest_{fi(d)}^d - X_i^d\right) + c_2 \times rand2_i^d \times \left(gbest^d - X_i^d\right) \tag{12}$$

由公式 (12) 可见, 对比公式 (11), 加入全局最优项的全面学习策略是为了降低全面学习策略的随机性, 加强顶层粒子的局部开采能力. 同时, 由于全局最优项可能处于任何一个子群, 因此增加了全局最优项的学习策略也有助于信息交互, 增强子群的多样性防止早熟收敛.

© 中国科学院软件研究所 http://www.jos.org.cn

2.3 基于维度概率的粒子交叉重置策略

为了避免分群策略导致的子群多样性降低的风险, 合适的子群信息交互策略显得尤为重要. 实验证明, 在粒子群算法的收敛过程中, 部分粒子会在一定迭代次数后陷入局部最优而导致其在整个种群的迭代过程中失去作用进而浪费算法的计算次数[22]. 针对这一特性, 同时结合多种群策略需要在迭代过程中进行信息交互的需求, 本文提出一种基于维度概率的粒子重置策略, 该策略依据维度概率将停滞粒子与全局最优解交叉重置, 有效地将陷入局部最优解的粒子重新加入到搜索任务中. 由于停滞粒子和最优粒子可能存在于两个不同的种群中, 因此重置策略可以实现种群间的信息交互, 进而提升了种群多样性. 同时, 依据线性递减的维度重置概率对粒子进行重置, 使得其每个维度在多次重置中均有机会向全局最优学习, 且可以避免某一维多次无效重置导致的种群多样性降低的不足.

图 2 和图 3 分别给出了依维度重置策略的流程图与提出的 DPMPSO 的整体流程图. 图 2 中 $X(max)$: 适应度最大的粒子对应位置; p_{ri}^{d} : 粒子第 d 维对应的重置概率; *stay*: 种群的停滞次数 (图 3 中相同). 如图 3 可见, DPMPSO 首先依据上述算法 1 对初始化以后的种群进行分子群操作. 之后对底层群与顶层群分别进行更新: 首先依据算法 3 为两个群中的粒子寻找对应的全面学习目标, 再依据不同的更新策略分别更新两个种群中粒子的位置与速度. 之后, 计算每个种群中所有粒子的适应度值, 通过与自身历史最优比较来确定是否需要更新历史最优并计算停滞次数. 最后对停滞次数超过阈值的粒子依据流程 1 进行依维度重置. 此外, 算法在进化过程中满足条件后即依据算法 2 进行重新分群. 图 3 中 k: 进化次数; w_0, w_1: 惯性权重上界 0.9, 下界 0.4; p: 对应子群标签 (1, 2); $X(min)$: 适应度最小的粒子对应位置; *max_gen*: 最大迭代次数; m: 停滞阈值 (30).

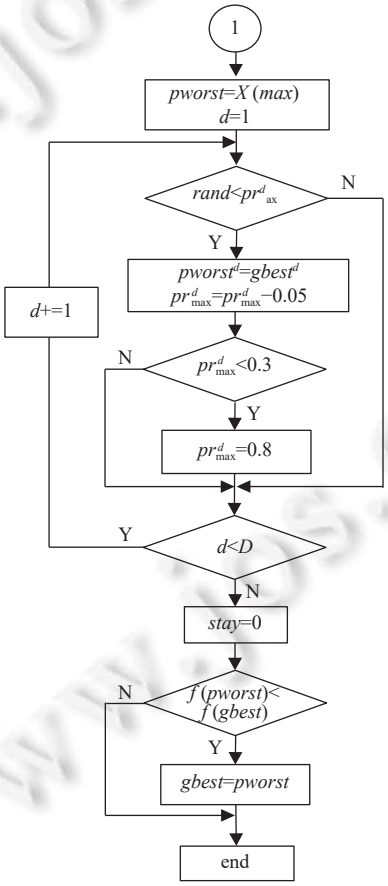

图 2 粒子的依维度交叉重置策略

© 中国科学院软件研究所 http://www.jos.org.cn

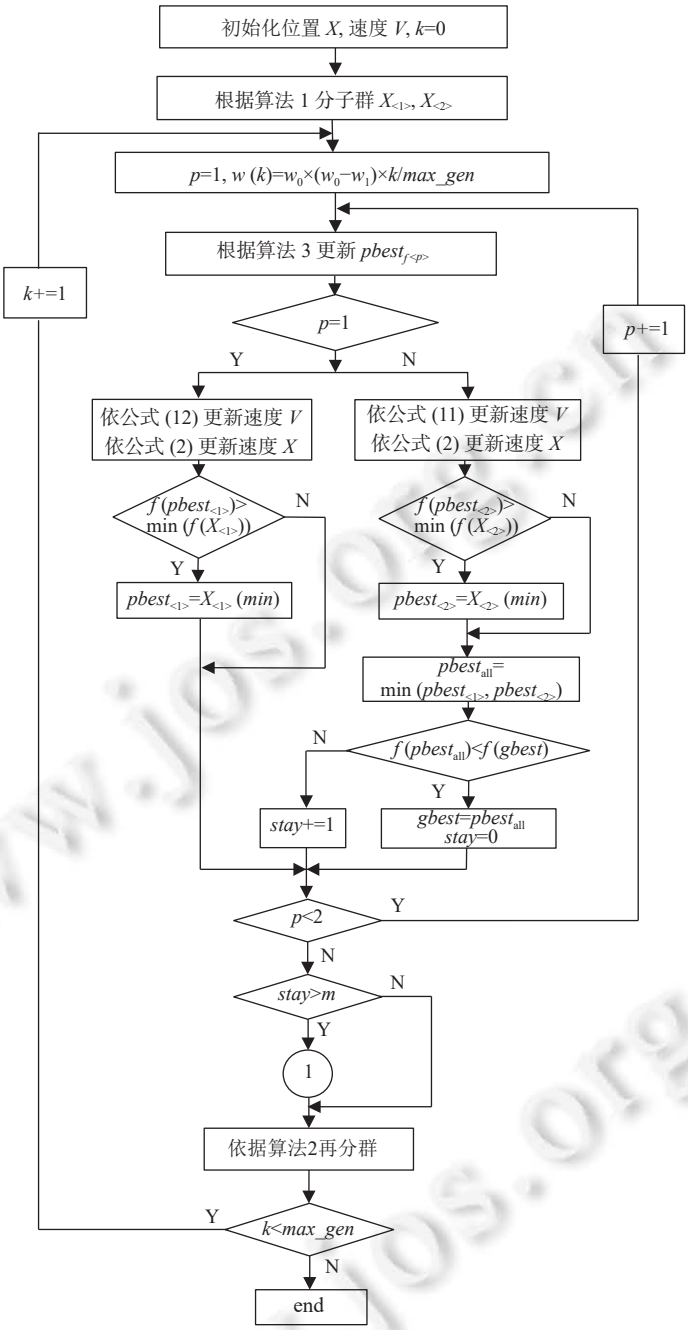

图 3 DPMPSO 算法流程图

3 实验结果与分析

为了证明 DPMPSO 的改进在处理各类优化问题时的有效性, 本文采用 10 个基准测试函数与 CEC2017 优化标准测试集分别对算法性能进行测试. 为了对比提出的算法在解决优化问题时的竞争力, 本文将传统的 PSO[1]、多种群 MSPSO (multi-swarm particle swarm optimization)[24]、全面学习策略 CLPSO[14]以及其效果良好

© 中国科学院软件研究所 http://www.jos.org.cn

的多种群变体 HCLPSO (heterogeneous comprehensive learning particle swarm optimization)[18]加入对比. 此外, 还包括两个改进其他优化算法: 多种群樽海鞘群算法 MSNSSA (multi-subpopulation based symbiosis and non-uniform Gaussian mutation salp swarm algorithm)[25]与改进的灰狼算法 HCOAG (hybrid coyote optimization algorithm with grey wolf optimizer)[26], 以及两个在优化问题上取得良好结果的 CEC 冠军算法: 改进的差分进化算法 L.SHADE (success-history based adaptive DE using linear population size reduction)[27]与多种群协方差矩阵自适应进化算法 BiPop.CMAES (bi-population covariance matrix adaptation evolution strategy)[28]. 对比算法对应的参数设计详见表 1. 为了比较的公平性, 所有比较算法的种群规模均为 40, 优化问题的维度设置为 30, 最大计算次数为 300 000 次. 所有比较算法均在 Matlab 2016b 中实现, 运行环境为 Windows 10 操作系统, Intel 2.4 GHz, 内存 8 GB.

表 1　对比算法参数定义

算法	参数设定	文献
PSO	$\omega=0.4, c1=c2=2$	[1]
MSPSO	$\chi=0.7298, c1=c2=1.49445, Rn=10, N=30$	[24]
CLPSO	$\omega=0.9\sim0.4, c1=c2=1.49445, m=7$	[14]
HCLPSO	$\omega=0.2\sim0.99, c1=2.5\sim0.5$ $c2=0.5\sim2.5, c=3\sim1.5$	[18]
MSNSSA	$C=2*r_3-a$	[25]
HCOAG	$N_C=5, N_P=20$	[26]
L.SHADE	$r^{N_{\text{init}}}=\{15,16,\ldots,24,25\}, r^{arc}=\{1.0,1.1,\ldots,2.9,3.0\}$ $p=\{0.05,\ldots,0.14,0.15\}, H=\{2,3,\ldots,9,10\}$	[27]
BiPop.CMAES	$\mu=\frac{\lambda}{2}, \omega_i=\sum_{i=1}^{\mu}\omega_i^2, d_\sigma=1+c_\sigma+2\max\left(0,\sqrt{\frac{\mu_W-1}{D+1}}-1\right)$	[28]
DPMPSO	$w=0.9\sim0.4, c1=c2=1.49445$ $Pr_{\text{init}}=0.8, m=30, S=5$	—

3.1　算法策略检验与参数设定

为了验证本文算法中一些关键参数的设定对算法性能的影响, 我们选取 4 个经典的基准函数来对比不同参数设定下的算法性能, 其中包括两个单峰函数: Sphere 和 NoisyQuadric 以及两个多峰函数: Dminima 和 Schewfel. 为了避免实验中随机性的影响, 本节每个实验中每个对比算法均单独运行 30 次并记录结果的均值 (Mean)、方差 (Dev) 以及达到一定阈值所需要的计算次数的均值 (FEs). 为了记录算法达到最优阈值的适应度计算次数, 首先通过初步实验确定每个函数的阈值 FEs 并记录在结果对比表中函数名下方.

3.1.1　分群策略的检验与参数设定

由于基于密度峰值的分群策略涉及两个子群的初始规模与迭代过程中子群的流动, 因此需要首先验证流动系数 S 对算法性能的影响. 表 2 是算法在不同流动系数下取得的结果, 由表 2 可以看出, 当流动系数过小时算法在处理多峰问题时的效果很差, 这是由于算法初期为顶层群分配过多的粒子用于开采, 容易导致整个种群缺乏充足的勘探能力进而易陷入早熟收敛. 反之当流动系数过大时算法在多峰问题上虽然取得了较好的结果, 但在单峰问题上性能却明显下降且需要更多的计算次数才能达到相应的阈值, 这是由于算法前期用于开采的粒子过少, 导致其开采能力不足进而降低了其收敛速度. 为了保证进化过程中的勘探和开采能力的均衡, 需设置适中的流动系数, 因此本文选取流动系数 S=5.

© 中国科学院软件研究所 http://www.jos.org.cn

确定流动系数 S 后, 为了验证本文提出的基于密度峰值的种群划分策略相比随机划分策略与单独依据适应度划分策略具有更好的性能, 我们将算法的分群策略分别变为随机划分策略与单独依据适应度划分策略, 而算法的其他策略则保持不变, 将两个变体分别定义为版本 1 (Version 1) 与版本 2 (Version 2) 并进行测试. 表 3 记录了两个变体与原算法在 S=5 时的实验结果. 由表中结果可见, 对比两种常见的分群策略, 本文提出的基于密度峰值的种群划分策略在求解多峰问题上的性能明显优于随机划分策略与单独依据适应度划分策略, 说明基于密度峰值的分群策略有效提升了种群的多样性. 在求解单峰问题上, 基于密度峰值种群划分策略的性能虽然与其他两个分群策略没有明显差异, 但是其达到一定阈值所需要计算的适应值次数却明显低于其他两个分群策略, 这说明基于密度峰值的分群策略具有更好的收敛性能. 这是由于本文提出的基于密度峰值的种群划分策略不仅考虑到种群的适应度信息以保证处于不同适应度水平的粒子可以专注于不同的搜索任务, 同时通过相对距离的思想将每个子群的分布信息加以考虑, 以确保划分后的子群能够均匀地分布在整个搜索空间的不同位置, 从而在保证收敛性能的同时增强了整个种群的多样性, 避免了早熟收敛.

表 2 不同流动系数下的 DPMPSO 实验结果对比

函数	S	Mean ± Dev	FEs	函数	S	Mean ± Dev	FEs
Sphere (1E–50)	3	**3.06E–94±5.11E–95**	176490	Dminima (4.6E–10)	3	4.57E–10±1.83E+00	158493
	4	9.81E–91±1.36E–91	**170379**		4	**4.57E–10±0**	156031
	5	1.08E–89±3.76E–90	189432		5	**4.57E–10±0**	**124676**
	6	5.19E–82±7.47E–83	232670		6	**4.57E–10±0**	127443
	7	2.39E–66±3.87E–66	260842		7	**4.57E–10±0**	160625
NoisyQuadric (1E–2)	3	**1.05E–03±2.61E–04**	**78619**	Schewfel (0)	3	4.44E+01±8.81E+01	272895
	4	1.29E–03±1.37E–04	80951		4	4.44E+01±8.81E+01	257045
	5	1.62E–03±7.19E–04	80788		5	**0±0**	193190
	6	1.78E–03±8.74E–04	85436		6	**0±0**	**190561**
	7	1.85E–03±6.41E–04	85406		7	**0±0**	223588

表 3 不同种群划分策略的 DPMPSO 实验结果对比

函数	版本	Mean ± Dev	FEs	函数	版本	Mean ± Dev	FEs
Sphere (1E–50)	Version 1	5.61E–82±4.32E–82	249158	Dminima (4.6E–10)	Version 1	4.57E–10±1.83E+00	170135
	Version 2	1.60E–82±3.95E–82	247699		Version 2	4.57E–10±1.83E+00	156747
	DPMPSO	**1.08E–89±3.76E–90**	**189432**		DPMPSO	**4.57E–10±0**	**124676**
NoisyQuadric (1E–2)	Version 1	1.78E–03±8.56E–04	84134	Schewfel (0)	Version 1	2.96E+01±5.48E+01	236734
	Version 2	**1.48E–03±9.33E–04**	81080		Version 2	1.48E+01±4.18E+01	249431
	DPMPSO	1.62E–03±7.19E–04	**80788**		DPMPSO	**0±0**	**193190**

3.1.2 重置策略的检验与参数设定

除了分群策略, 本文提出的依维度重置策略中也包含了两个关键参数: 粒子交叉重置策略中的停滞阈值 m 与初始的维度重置概率 Pr_{init}, 同样需通过实验确定其最优参数范围. 表 4 记录了不同停滞阈值下算法的实验结果, 由结果可见, 当阈值设置较小时算法在单峰问题上的结果较好而在处理多峰问题时效果较差且频繁重置导致适应度的计算次数上升, 反之当停滞阈值设置稍大时算法则在多峰问题中取得了较好的效果. 这可能是由于阈值较小时停滞粒子的交叉重置较为频繁, 使得算法整体更加专注于最优解的开采, 因此在求解单峰问题时表现出了优良的性能. 然而过多的交叉重置也会破坏部分粒子的原有拓扑结构进而降低种群多样性, 导致其在处理多峰问题时性能较差. 反之当阈值较大时粒子不易进入交叉重置过程, 虽然保证了种群多样性, 但因其无法有效地向全局最优学习导致其开采能力较弱, 降低了算法收敛速度. 而表 5 记录了不同初始维度重置概率下算法的实验结果, 由结果可见, 较大的维度概率更适合于处理单峰问题, 这是由于粒子中更多的维度与全局最优交叉, 可有效提升种群的开采能力; 而较小的维度概率可以在与最优解交叉重置时一定程度上保留原有粒子的拓扑结构, 维护了种群多样性,

© 中国科学院软件研究所 http://www.jos.org.cn

因此有利于多峰问题的求解. 综合上述实验, 为了保证算法能够均衡开采与勘探, 我们建议设定交叉重置策略停滞阈值 m=30, 初始维度重置概率 Pr_{init}=0.8.

表 4 不同停滞阈值下的 DPMPSO 实验结果对比

函数	m	Mean ± Dev	FEs	函数	m	Mean ± Dev	FEs
Sphere(1E–50)	10	1.37E–57±3.94E–58	225 479	Dminima (4.6E–10)	10	4.57E–10±1.83E+00	193 295
	20	4.61E–89±1.54E–90	**190 543**		20	4.57E–10±1.83E+00	194 703
	30	**4.01E–89±9.33E–90**	193 197		30	**4.57E–10±0**	**139 460**
	40	5.94E–82±1.86E–83	234 890		40	**4.57E–10±0**	145 483
	50	3.67E–77±3.30E–78	269 741		50	**4.57E–10±0**	147 096
NoisyQuadric (1E–2)	10	9.79E–03±1.65E–03	149 643	Schewfel (0)	10	7.89E+00±3.05E+01	231 066
	20	1.46E–03±6.30E–04	96 790		20	1.45E–12±7.53E–13	189 450
	30	**1.25E–03±8.73E–04**	**85 470**		30	**0±0**	168 746
	40	3.67E–03±9.05E–04	117 634		40	**0±0**	**166 612**
	50	7.84E–03±6.95E–04	159 117		50	**0±0**	169 784

表 5 不同维度概率下的 DPMPSO 实验结果对比

函数	Pr_{init}	Mean ± Dev	FEs	函数	Pr_{init}	Mean ± Dev	FEs
Sphere (1E–50)	0.9	1.08E–89±7.81E–91	225 479	Dminima (4.6E–10)	0.9	4.57E–10±1.83E+00	184 397
	0.8	**2.37E–90±6.40E–91**	**184 370**		0.8	**4.57E–10±0**	140 593
	0.7	3.15E–90±1.30E–90	193 197		0.7	**4.57E–10±0**	**139 745**
	0.6	2.79E–87±6.27E–88	234 890		0.6	**4.57E–10±0**	147 693
NoisyQuadric (1E–2)	0.9	**1.31E–03±5.46E–04**	**85 830**	Schewfel (0)	0.9	1.45E–12±7.53E–13	186 437
	0.8	1.42E–03±3.35E–04	94 397		0.8	**0±0**	**149 083**
	0.7	2.09E–03±8.17E–04	106 509		0.7	**0±0**	150 974
	0.6	5.41E–03±1.97E–04	117 315		0.6	**0±0**	167 340

3.1.3 算法策略的自身消融实验

在确定交叉重置策略停滞阈值与初始维度重置概率后, 为了证明本文中提出的 3 种策略的有效性, 将算法的分种群策略、改进的全面学习策略与重置策略分别移除且同时保持算法其他策略不变, 将其定义为 Version 3–6 (即 Version 3 中不包括分群策略, Version 4 与 Version 5 中分别在所有子群中均使用经典的全面学习策略与改进的全面学习策略, Version 6 中不包含重置策略) 并与原有算法进行对比分析, 表 6 记录了 4 个版本与原算法的对比结果. 由表 6 的结果可见, 针对移除了分群策略的 Version 3, 其整体结果无论是在收敛精度还是收敛速度上结果都较差. 总的来说, 其结果在单峰问题上略优于经典的 CLPSO, 在多峰问题上基本与 CLPSO 相似. 而对于 Version 4 与 Version 5 而言, 两个种群均采用经典的全面学习策略时, 算法在处理单峰问题时明显结果较差, 反之当两个子群均采用改进的添加全局项的学习策略时, 算法在多峰问题上的结果则明显较差. 而在移除依维度重置策略的 Version 6 中, 无论是处理单峰还是多峰函数, 算法的计算次数均明显增加且最终结果亦劣于原算法. 表 6 中的结果也证明了上面几节中对每个策略作用的分析.

3.2 基准函数实验结果

本文首先选取了 10 个优化领域常用的基准测试函数进行实验, 其中 F1–F4 为单峰函数, F5–F10 为多峰函数, 函数详细定义可见表 7. 所有对比算法在各优化问题的运行结果记录在表 8 中, 其中最优结果加粗表示, 且表中 function 一栏下括号内为每个函数的阈值, 用以确定算法是否成功找到优化函数的最优解与成功找到最优解时所需计算次数, 该值通过所有算法初步实验后的结果确定. 为了排除随机性的影响, 每个对比算法会独立运行 30 次并统计结果. 为了全面考虑算法的优化效果、稳定性以及计算成本, 统计结果包括: 30 次运行结果的均值 (Mean)、方差 (Dev)、算法均值的排序 (rank)、达到一定阈值所需要的计算次数的均值 (FEs) 以及算法达到该阈值的成功

© 中国科学院软件研究所 http://www.jos.org.cn

率 (*SR*), 其中成功率 *SR* 的计算公式如下:

$$SR = 100 \times \frac{(\text{number of successful runs})}{(\text{total runs})} \tag{13}$$

表 6 有无 3 种策略的 DPMPSO 实验结果对比

函数	版本	Mean ± Dev	FEs	函数	版本	Mean ± Dev	FEs
Sphere (1E–50)	Version 3	7.51E–51±9.34E–52	296 733	Dminima (4.6E–10)	Version 3	4.57E–10±6.93E–15	163 743
	Version 4	5.04E–45±1.22E–45	—		Version 4	**4.57E–10±0**	150 976
	Version 5	9.11E–76±3.16E–76	190 146		Version 5	3.11E–07±6.97E–08	—
	Version 6	1.43E–81±1.98E–81	251 440		Version 6	4.57E–10±1.83E+00	179 040
	DPMPSO	**2.37E–90±6.40E–91**	**184 370**		DPMPSO	**4.57E–10±0**	**140 593**
NoisyQuadric (1E–2)	Version 3	7.51E–03±1.09E–03	163 732	Schewfel (0)	Version 3	4.01E+00±1.95E+01	—
	Version 4	2.91E–03±4.27E–04	124 617		Version 4	1.46E+00±2.61E–01	—
	Version 5	2.36E–03±6.05E–04	107 643		Version 5	6.01E+00±3.10E+00	—
	Version 6	3.49E–03±7.23E–04	136 160		Version 6	**0±0**	292 640
	DPMPSO	**1.42E–03±3.35E–04**	**94 397**		DPMPSO	**0±0**	**167 340**

表 7 基准测试函数定义

名称	定义	阈值	$f_{\min}$
Sphere (F1)	$f_1(x) = \sum_{i=1}^{D} x_i^2$	[−100, 100]	0
NoisyQuadric (F2)	$f_2(x) = \sum_{i=1}^{D} i x_i^4 + random[0,1)$	[−1.28, 1.28]	0
Schwefel 2.22 (F3)	$f_3(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \Pi_{i-1}^{D} \lvert x_i \rvert$	[−10, 10]	0
High Conditioned Elliptic (F4)	$f_4(x) = \sum_{i=1}^{D} \left(10^6\right)^{\frac{i-1}{D-1}} * x_i^2$	[−100, 100]	0
Dminima (F5)	$f_5(x) = 78.332331408 + \sum_{i=1}^{D} \frac{x_i^4 - 16x_i^2 + 5x_i}{D}$	[−5, 5]	0
Rastrigin10 (F6)	$f_6(x) = \sum_{i=1}^{D} \left((a_i x_i)^2 - 10\cos(2\pi a_i x_i) + 10\right), a_i = \frac{i-1}{10D-1}, i = 1,\ldots,D$	[−5, 5]	0
Weierstrass (F7)	$f_7(x) = \sum_{i=1}^{D} \left(\sum_{k=0}^{kmax} \left[a^k \cos\left(2\pi b^k \cdot (x_i + 05)\right)\right]\right) - \sum_{k=0}^{kmax} \left[a^k \cos\left(2\pi b^k \cdot 0.5\right)\right]$	[−0.5, 0.5]	0
Griewank (F8)	$f_8(x) = \sum_{i=1}^{D} {x_i}^2 / 4000 - \prod_{i=1}^{D} \cos\left(x_i / \sqrt{i}\right) + 1$	[−600, 600]	0
Schewfel (F9)	$f_9(x) = 418.982887273 \times D - \sum_{i=1}^{D} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	[−500, 500]	0
Ackley (F10)	$f_{10}(x) = -20\exp\left(-0.2\sqrt{\sum_{i=1}^{D} \frac{x_i^2}{D}}\right) - \exp\left(\sum_{i=1}^{D} \cos(2\pi\, x_i)/D\right) + 20 + \mathrm{e}$	[−32, 32]	0

© 中国科学院软件研究所 http://www.jos.org.cn

表 8 基准函数测试结果对比

算法		PSO	MSPSO	CLPSO	HCLPSO	MSNSSA	HCOAG	L.SHADE	BiPop.CMAES	DPMPSO
F1	Mean	3.44E–21	5.79E–18	8.85E–35	8.37E–57	2.07E–49	5.74E–-23	**5.79E–102**	1.39E–51	3.44E–91
	Dev	5.68E–21	1.23E–17	8.32E–35	1.80E–56	4.23E–50	5.03E–24	**9.17E–102**	3.12E–52	6.72E–91
	SR	0	0	0	100	66.7	0	100	86.7	100
	FEs	—	—	—	266951	257496	—	**149638**	182280	193295
	Rank	8	9	6	3	5	7	**1**	4	2
F2	Mean	5.12e-03	4.03E–03	3.56E–03	4.90E–03	4.93E–03	2.66E–03	3.24E–03	2.89E–03	**1.38E–03**
	Dev	1.91E–03	1.11E–04	6.59E–04	1.61E–03	2.41E–03	9.15E–04	1.66E–03	7.97E–04	**2.37E–03**
	SR	100	100	100	100	100	100	100	100	**100**
	FEs	118320	141967	162225	90971	127520	134520	110764	140040	**109326**
	Rank	9	6	5	7	8	2	4	3	**1**
F3	Mean	4.78E–06	2.93E–10	8.70E–22	2.28E–28	7.47E–24	1.17E–26	2.10E–43	2.10E–35	**3.76E–47**
	Dev	8.12E–06	8.45E–10	4.59E–22	4.50E–28	1.89E–25	6.18E–27	5.92E–43	5.92E–36	**9.38E–47**
	SR	0	0	0	0	0	0	100	83.3	**100**
	FEs	—	—	—	—	—	—	210889	255952	**186794**
	Rank	9	8	7	4	6	5	2	3	**1**
F4	Mean	6.64E–10	2.08E–13	8.14E–32	1.36E–54	2.79E–40	8.09E–38	**4.46E–103**	2.88E–40	3.96E–88
	Dev	1.18E–09	4.60E–13	6.89E–32	2.66E–54	5.91E–40	3.57E–38	**5.68E–104**	6.44E–41	8.75E–88
	SR	0	0	0	100	0	0	**100**	100	100
	FEs	—	—	—	264376	—	—	**183526**	218399	193681
	Rank	9	8	7	3	4	6	**1**	5	2
F5	Mean	1.11E+01	8.54E+00	**4.57E–10**	4.57E–10	1.25E+00	7.54E–01	**4.57E–10**	1.88E–01	**4.57E–10**
	Dev	2.33E+00	0	**0**	6.93E–15	2.51E–01	8.65E–01	**0**	3.97E–01	**0**
	SR	0	0	**100**	70	16.7	6.67	**100**	53.3	**100**
	FEs	—	—	158009	214930	297146	174659	256442	139560	**123767**
	Rank	9	8	**1**	4	7	6	**1**	5	**1**
F6	Mean	1.88E–14	3.55E–15	**0**	**0**	6.63E–14	**0**	**0**	**0**	**0**
	Dev	3.07E–14	3.24E–15	**0**	**0**	1.14E–13	**0**	**0**	**0**	**0**
	SR	26.7	20	**100**	**100**	16.7	**100**	**100**	**100**	**100**
	FEs	137280	—	123348	91247	159710	179960	**69179**	104360	72418
	Rank	8	7	**1**	**1**	9	**1**	**1**	**1**	**1**
F7	Mean	1.16E+00	1.07E+00	**0**	9.47E–16	1.02E+00	3.19E–13	3.31E–15	**0**	**0**
	Dev	7.07E–01	8.80E–01	**0**	2.50E–15	7.55E–01	2.55E–14	4.54E–15	**0**	**0**
	SR	0	0	**100**	86.7	0	36.7	60	**100**	**100**
	FEs	—	—	215162	280525	—	324476	**183840**	224964	198954
	Rank	9	8	**1**	4	7	6	5	**1**	**1**
F8	Mean	4.30E–02	1.09E–02	**0**	**0**	3.74E–02	2.98E–04	**0**	**0**	**0**
	Dev	7.18E–02	2.04E–02	**0**	**0**	3.63E–02	7.64E–05	**0**	**0**	**0**
	SR	0	0	**100**	**100**	16.7	50	**100**	**100**	**100**
	FEs	—	—	212664	178739	292720	246927	**141480**	187450	146211
	Rank	9	7	**1**	**1**	8	1	**1**	**1**	**1**
F9	Mean	8.67E+03	2.63E+03	7.89E+00	6.31E+01	9.06E+03	9.76E+03	1.45E–12	1.80E+02	**0**
	Dev	1.71E+03	3.70E+02	3.05E+01	7.57E+01	7.87E+02	5.57E+02	7.53E–13	1.92E+02	**0**
	SR	0	0	86.7	13.3	0	0	20	26.6	**100**
	FEs	—	—	162839	201780	—	—	218566	164680	**160027**
	Rank	7	6	3	4	8	9	2	5	**1**
F10	Mean	1.40E–08	1.24E–08	1.06E–14	2.81E–14	6.30E–10	3.76E–12	3.00E–14	**7.10E–15**	8.04E–15
	Dev	4.51E–08	3.27E–08	2.32E–15	5.10E–15	8.56E–10	1.12E–12	6.56E–15	**0**	3.66E–15
	SR	0	0	66.7	13.3	0	0	0	**100**	86.7
	FEs	—	—	293737	341964	—	—	—	258080	**218094**
	Rank	9	8	3	4	7	6	5	**1**	2

© 中国科学院软件研究所 http://www.jos.org.cn

如表 8 所示, 在处理 4 个单峰问题时, 仅有提出的 DPMPSO 与 L.SHADE 取得了较优的结果, 其中 L.SHADE 在处理 F1 与 F4 时取得了最好的性能, 而 DPMPSO 则在 F2 与 F3 上得到了最好的结果. 紧随其后的便是 BiPop. CMAES 与 HCLPSO. 而通过达到指定阈值所需要的计算次数亦能看出提出的 DPMPSO 在处理单峰问题时不仅能够取得好的结果, 且收敛速度较快. 这主要是由于本文提出的分群策略和依维度概率重置策略有效提高了种群的局部开采能力, 使得算法能快速收敛至最优区域. 在处理多峰问题时, 提出的 DPMPSO 在 F5-F9 中均取得了最优的性能, 且在 F10 中的运行结果仅次于 BiPop. CMAES. 此外可以看出, 经典的 CLPSO 在处理多峰问题时表现出其强大的全局勘探能力. 但值得一提的是, 其多种群改进算法 HCLPSO 虽然在单峰问题上对 CLPSO 有优化, 但是在处理多峰问题时效果却劣于 CLPSO, 这可能是因为其分种群策略并没有照顾到粒子的全局分布且种群间缺少足够的信息交互进而导致勘探能力下降. 而本文提出的基于密度峰值的分群策略则确保了底层种群在前期可以分布在搜索空间的各个位置, 从而使得全面学习策略能够最大限度地发挥作用. 实验结果亦表明提出的 DPMPSO 具有良好的勘探能力.

为了直观对比所有对比算法在 10 个基准函数上的表现结果, 图 4 统计了各种算法在所有函数上取得结果的均值的平均排序. 同时为了验证所有算法的平均排序的差异不是随机事件. 本文首先对均值结果进行 Friedman 检验[29], 零假设为在显著性水平 α=0.05 的条件下, 对比算法在所有实验函数中所得的均值排序没有显著性差异. 检验结果 p=1.0531E–09 拒绝了原假设. 从图 4 可以看出, 在 10 个基准函数的实验中, DPMPSO 取得了最好的均值排序, 而 L.SHADE 与 BiPop. CMAES 同样取得了不错的结果. 值得一提的是, 由于全面学习策略在单峰问题上的表现不理想, CLPSO 整体排序并不高. 同样, 其多种群变体 HCLPSO 由于破坏了 CLPSO 的全局勘探能力, 使得其在多峰问题上表现不佳, 整体排序第 4. 为了进一步体现算法的收敛性质, 图 5 展示所有对比算法在处理优化函数时的收敛图, 其中包含两个单峰函数与两个多峰函数. 由图 5 可见, 在处理单峰函数时, DPMPSO 能通过较少的计算次数快速收敛且继续开采, 这主要是由于分群策略中的顶层群配合依维度概率与全局最优的交叉重置, 大幅提高了顶层群的局部开采能力; 而在处理多峰函数时, DPMPSO 前期收敛较缓以倾向于勘探全局, 而后期能够持续收敛至最小值. 这主要是由于基于密度峰值的分群策略在初始化时就保证了子群的多样性, 同时配合子群的流动策略与重置时的信息交互, 保证了子群在迭代过程中的多样性而使得其不易陷入早熟收敛. 而子群流动策略使整个种群在后期能专注于开采工作, 保证了处理多峰问题时具有较高的精度.

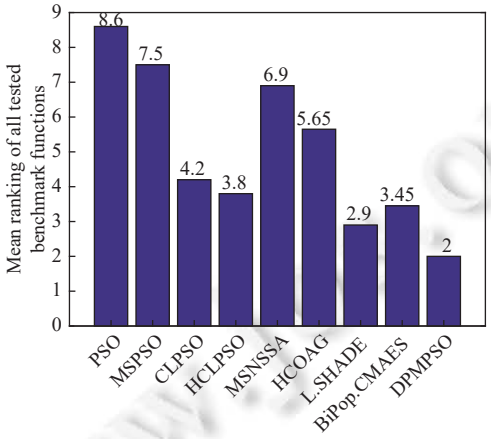

图 4 不同算法在基准函数结果均值 (Mean) 排序

为了证明表 8 中 DPMPSO 与其他算法的对比结果具有统计显著性, 本文对所有函数结果的均值项进行 Holm 检验, 零假设为在 α=0.05 的显著性水平下, DPMPSO 与其他对比算法的均值结果不具有显著性差异, 表 9 记录了 Holm 检验的实验结果. 由表 9 结果可见, 所有对比算法中, 仅有 L.SHADE 拒绝了原假设, 即 L.SHADE 与提出的 DPMPSO 取得了较为相似的结果. 而与其他算法相比, DPMPSO 的改进效果均有显著性.

© 中国科学院软件研究所 http://www.jos.org.cn

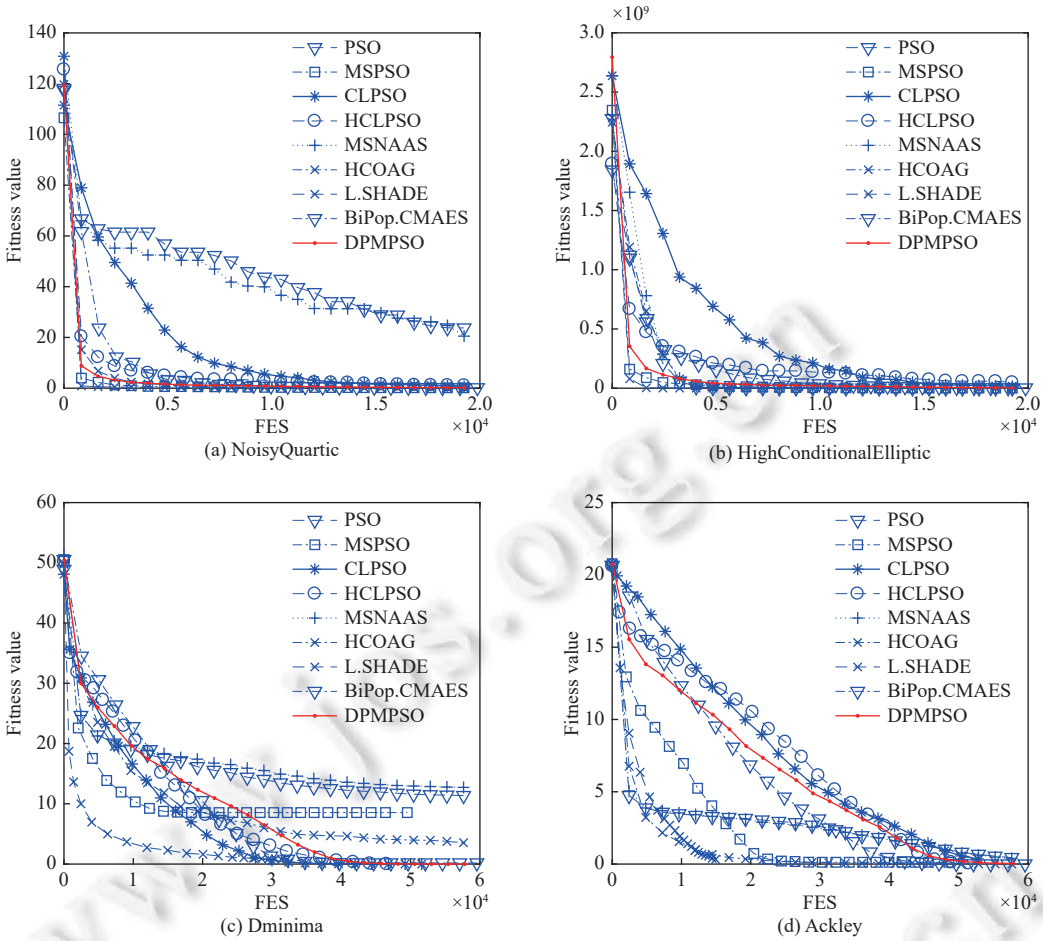

图 5 基准测试函数收敛图

表 9 DPMPSO 与其他对比算法的 Holm 检验结果

算法	$\alpha_{0.05}$	p-value
PSO	0.0063912	3.6815E–17
MSPSO	0.0073008	1.1703E–13
CLPSO	0.012741	0.00061179
HCLPSO	0.016952	0.0045633
MSNSSA	0.0085124	9.7546E–12
HCOAG	0.010206	7.5492E–08
L.SHADE	0.05	0.14848
BiPop.CMAES	0.025321	0.021191

3.3 CEC2017 函数实验结果

为了进一步证明 DPMPSO 在处理各类优化问题时的改进效果, 采用 CEC2017 优化的基准对算法进行测试. CEC2017 测试集共有 30 个测试函数, 所有函数具体定义可见文献 [30], 且本文的实验过程均依据文献 [30] 定义的标准过程进行, 最大适应度评价次数为优化问题的维度 30×10000=300000 次, 每个算法单独计算 51 次然后记录均值 (Mean) 方差 (Dev) 与均值的排序 (Rank), 实验结果记录在表 10, 同样将最优结果加粗表示. 由表 10 中记录的结果可以看出, 在 3 个单峰函数的对比中, 提出的 DPMPSO 取得了两次第 1、一次第 3, 对比另一个针对 CLPSO

© 中国科学院软件研究所 http://www.jos.org.cn

的多种群改进算法 HCLPSO 取得了更好的结果, 这证明了依维度重置策略有效提高了全面学习策略的局部开采能力. 而在简单多峰函数的实验中, DPMPSO 仍然取得了 3 次第 1 与 3 次第 2, 相同情况下, HCLPSO 的表现则不是很理想, 这主要是由于本文算法采用的分群与流动策略在提高种群局部开采能力的同时亦保持了其多样性, 从而使其在面对多峰问题时不易陷入早熟收敛. 结合余下的混合函数与复合函数结果来看, 虽然提出的 DPMPSO 在处理少数函数如 F14, F18 时取得的结果不太理想. 但是其结果在整体上具有更强的鲁棒性. 同样为了直观对比所有算法的均值结果, 图 6 给出了均值排序. 由图 6 可见, 本文提出的 DPMPSO 在结果上明显优于 CLPSO 与其改进算法 HCLPSO. 同时在与两个冠军算法的对比中亦具备一定的竞争力, 这也反映了 DPMPSO 在处理各类优化问题时能够有效地平衡局部开采与全局勘探.

表 10　CEC2017 函数测试结果对比

算法		PSO	MSPSO	CLPSO	HCLPSO	MSNSSA	HCOAG	L.SHADE	BiPop.CMAES	DPMPSO
F1	Mean	4.97E+09	3.66E+06	4.21E+03	7.75E+02	4.87E+05	2.08E+06	6.91E+02	3.18E+03	**4.61E+02**
	Dev	3.06E+09	7.09E+05	6.39E+02	2.37E+01	6.41E+05	9.84E+05	1.11E+01	5.09E+02	**1.94E+01**
	Rank	9	8	5	3	6	7	2	4	**1**
F2	Mean	2.34E+30	9.81E+23	5.17E+06	3.76E+06	5.67E+19	1.22E+10	4.67E+04	2.68E+06	**3.66E+04**
	Dev	7.60E+29	3.55E+23	4.21E+06	9.86E+06	2.57E+20	3.09E+09	5.01E+05	6.23E+06	**9.73E+03**
	Rank	9	8	5	4	7	6	2	3	**1**
F3	Mean	7.19E+06	2.67E+01	6.45E−03	9.43E−04	6.34E+00	2.71E−03	5.23E−04	**9.68E−06**	6.34E−04
	Dev	2.00E+05	3.95E+00	7.72E−04	5.39E−04	1.04E+00	3.19E−04	1.54E−04	**5.14E−06**	3.84E−05
	Rank	9	8	6	4	7	5	2	**1**	3
F4	Mean	9.81E+01	5.77E+01	2.97E+01	4.31E+01	6.74E+01	2.51E+01	1.39E+01	8.61E+00	**8.07E+00**
	Dev	6.10E+01	1.48E+01	9.14E+00	1.99E+01	3.08E+01	8.07E+00	1.07E+01	4.98E+00	**3.94E+00**
	Rank	9	7	5	6	8	4	3	2	**1**
F5	Mean	6.51E+02	1.17E+02	6.85E+01	5.76E+01	9.17E+01	6.08E+01	**2.64E+01**	3.94E+01	3.85E+01
	Dev	3.35E+02	1.31E+02	5.40E+01	4.39E+01	6.91E+01	4.14E+01	**1.08E+01**	2.16E+01	3.01E+01
	Rank	9	8	6	4	7	5	**1**	3	2
F6	Mean	5.98E+01	3.68E+01	5.05E−13	4.82E−13	6.09E−02	3.72E−05	5.61E−13	**4.72E−13**	5.24E−13
	Dev	2.62E+01	1.05E+01	2.67E−13	6.04E−14	8.68E−02	4.26E−06	3.30E−13	**1.85E−13**	1.57E−13
	Rank	9	8	3	2	7	6	5	**1**	4
F7	Mean	9.09E+02	2.40E+02	7.21E+01	5.75E+01	1.06E+02	8.71E+01	**2.13E+01**	6.29E+01	3.71E+01
	Dev	3.51E+02	1.05E+02	2.15E+01	2.43E+01	7.94E+01	6.69E+01	**9.56E+00**	2.65E+01	1.39E+01
	Rank	9	8	5	3	7	6	**1**	4	2
F8	Mean	6.01E+02	5.89E+02	5.69E+01	4.89E+01	3.59E+01	8.21E+01	4.96E+01	**2.85E+01**	3.07E+01
	Dev	4.76E+02	2.81E+02	3.05E+01	2.81E+01	9.71E+00	3.02E+01	8.24E+00	**8.90E+00**	9.66E+00
	Rank	9	8	6	4	3	7	5	**1**	2
F9	Mean	3.02E+04	7.81E+00	8.18E+00	7.95E+00	5.87E+01	9.08E+01	1.91E+00	4.68E+00	**8.96E−01**
	Dev	7.61E+03	2.51E+00	5.21E+00	2.06E+00	4.47E+01	5.81E+01	1.08E+00	3.23E+00	**1.94E−01**
	Rank	9	4	6	5	7	8	2	3	**1**
F10	Mean	3.52E+03	2.87E+03	2.45E+03	2.19E+03	2.81E+03	2.33E+03	2.37E+03	1.93E+03	**1.85E+03**
	Dev	8.61E+02	6.20E+02	4.13E+02	3.61E+00	4.27E+02	3.62E+02	4.90E+02	2.19E+02	**3.21E+02**
	Rank	9	8	6	3	7	4	5	2	**1**
F11	Mean	5.86E+02	3.20E+02	4.37E+01	5.59E+01	6.82E+01	8.71E+01	**2.16E+01**	3.64E+01	4.55E+01
	Dev	1.38E+02	6.83E+01	2.32E+01	1.73E+01	5.61E+01	6.28E+01	**1.46E+01**	2.44E+01	2.21E+01
	Rank	9	8	3	5	6	7	**1**	2	4
F12	Mean	6.85E+08	8.05E+05	3.61E+04	2.99E+04	5.16E+04	6.36E+04	2.33E+04	3.41E+04	**1.81E+04**
	Dev	3.08E+08	4.18E+05	1.05E+04	1.19E+04	2.49E+04	7.65E+04	1.38E+04	2.07E+04	**1.06E+04**
	Rank	9	8	5	3	6	7	2	4	**1**
F13	Mean	1.22E+05	5.61E+03	4.34E+02	**4.07E+01**	5.16E+03	6.47E+03	5.03E+02	4.73E+02	4.91E+02
	Dev	3.64E+05	1.85E+03	6.35E+02	**8.44E+01**	3.64E+03	7.42E+03	2.84E+02	4.62E+02	2.37E+02
	Rank	9	7	2	**1**	6	8	5	3	4

© 中国科学院软件研究所 http://www.jos.org.cn

表 10 CEC2017 函数测试结果对比 (续)

算法		PSO	MSPSO	CLPSO	HCLPSO	MSNSSA	HCOAG	L.SHADE	BiPop.CMAES	DPMPSO
F14	Mean	7.03E+04	5.15E+04	1.55E+03	3.48E+03	9.01E+03	8.10E+03	**1.25E+03**	1.50E+03	1.32E+03
	Dev	9.80E+04	2.66E+04	2.36E+03	5.15E+03	5.15E+03	4.37E+03	**1.71E+03**	2.88E+03	1.08E+03
	Rank	9	8	4	5	7	6	**1**	3	2
F15	Mean	7.41E+03	5.38E+02	2.97E+02	3.55E+02	5.43E+02	1.74E+03	3.32E+02	4.23E+02	3.44E+02
	Dev	9.44E+03	9.61E+01	4.12E+02	2.48E+02	3.06E+02	9.41E+02	1.77E+02	2.57E+02	1.57E+02
	Rank	9	6	1	4	7	8	2	5	3
F16	Mean	1.92E+03	2.64E+03	4.79E+02	5.13E+02	8.64E+02	3.19E+02	2.67E+02	3.40E+02	**2.49E+02**
	Dev	3.07E+03	1.39E+03	2.05E+02	3.11E+02	4.22E+02	1.37E+02	3.62E+02	1.27E+02	**7.02E+02**
	Rank	8	9	5	6	7	3	2	4	**1**
F17	Mean	6.38E+02	2.67E+02	1.65E+02	1.25E+02	4.81E+02	2.33E+02	**1.08E+02**	1.36E+02	1.28E+02
	Dev	1.61E+02	9.20E+01	5.43E+00	7.61E+01	3.27E+01	8.62E+01	**4.90E+01**	6.19E+01	5.91E+01
	Rank	9	7	5	2	8	6	**1**	4	3
F18	Mean	4.02E+05	**3.62E+04**	7.61E+04	8.95E+04	1.81E+05	8.03E+04	3.88E+04	4.79E+04	5.18E+04
	Dev	5.17E+05	**4.20E+04**	1.83E+04	2.61E+04	4.27E+04	1.62E+04	2.90E+04	3.19E+04	4.31E+04
	Rank	9	**1**	5	7	8	6	2	3	4
F19	Mean	4.06E+05	5.21E+04	2.17E+02	3.19E+02	4.82E+03	2.71E+02	**5.16E+01**	7.64E+01	6.15E+01
	Dev	3.18E+05	3.03E+04	1.32E+02	2.73E+02	6.68E+03	3.28E+02	**4.46E+01**	6.44E+01	3.21E+01
	Rank	9	8	4	6	7	5	**1**	3	2
F20	Mean	6.15E+02	3.05E+02	1.35E+02	1.99E+02	3.19E+02	1.36E+02	1.49E+02	**1.31E+02**	1.35E+02
	Dev	2.08E+02	6.28E+01	4.05E+01	2.19E+01	2.47E+01	5.61E+01	4.98E+01	**6.07E+01**	5.16E+01
	Rank	9	7	3	6	8	4	5	**1**	3
F21	Mean	4.22E+02	2.61E+02	2.34E+02	**2.17E+01**	3.56E+02	2.47E+02	1.91E+02	2.11E+02	1.85E+02
	Dev	5.10E+01	1.80E+01	9.35E+01	**7.44E+01**	2.56E+01	8.62E+01	6.84E+01	4.67E+01	1.32E+01
	Rank	9	7	5	**1**	8	6	3	4	2
F22	Mean	5.15E+03	6.17E+02	1.00E+02	1.00E+02	3.02E+02	4.10E+02	1.62E+02	1.50E+02	1.00E+02
	Dev	6.40E+03	3.54E+03	1.16E−01	3.95E−01	7.13E+00	3.21E+00	1.81E+01	4.81E+01	1.68E−01
	Rank	9	8	2	2	6	7	5	4	2
F23	Mean	6.14E+02	4.08E+02	4.17E+02	3.59E+02	6.43E+02	5.44E+02	**1.34E+02**	2.44E+02	1.46E+02
	Dev	3.72E+01	4.87E+01	2.25E+01	1.03E+01	1.74E+01	2.13E+01	**2.75E+01**	4.97E+01	6.30E+01
	Rank	8	5	6	4	9	7	**1**	3	2
F24	Mean	5.92E+02	7.64E+02	4.79E+02	4.55E+02	5.64E+02	5.29E+02	3.27E+02	3.14E+02	**3.00E+02**
	Dev	7.92E+01	1.39E+02	9.05E+01	7.58E+01	3.27E+01	3.94E+01	6.15E+01	5.71E+01	**6.02E+01**
	Rank	8	9	5	4	7	6	3	2	**1**
F25	Mean	4.98E+02	3.95E+02	3.89E+02	3.91E+02	4.01E+02	3.93E+02	**3.87E+02**	3.91E+02	3.89E+02
	Dev	2.16E+01	3.20E+01	2.13E+01	4.61E+01	5.27E+01	6.62E+01	**2.90E+01**	3.19E+01	3.21E+01
	Rank	9	7	3	5	8	6	**1**	5	3
F26	Mean	5.96E+03	5.20E+03	3.37E+03	3.59E+03	4.12E+03	5.01E+03	2.31E+03	2.14E+03	**1.85E+03**
	Dev	6.44E+02	4.83E+02	2.32E+02	2.73E+02	5.61E+02	3.29E+02	7.46E+02	5.47E+02	**2.21E+02**
	Rank	9	8	4	5	6	7	3	2	**1**
F27	Mean	5.85E+02	5.35E+02	5.18E+02	**5.13E+02**	5.22E+02	5.16E+02	**5.13E+02**	5.21E+02	**5.13E+02**
	Dev	3.14E+01	6.10E+00	4.99E+00	**7.02E+00**	6.49E+00	7.65E+01	**4.98E+01**	6.17E+01	**8.95E+01**
	Rank	9	8	5	**2**	7	4	**2**	6	**2**
F28	Mean	5.02E+02	4.61E+02	3.34E+02	3.47E+02	3.56E+02	4.41E+02	3.91E+02	**3.21E+02**	3.29E+02
	Dev	4.23E+01	1.80E+01	1.95E+01	5.44E+01	9.16E+01	7.34E+01	6.81E+01	**4.17E+01**	1.36E+01
	Rank	9	8	3	4	5	7	6	**1**	2
F29	Mean	9.13E+02	7.15E+02	6.57E+02	5.41E+02	6.64E+02	5.91E+02	5.21E+02	5.39E+02	**5.19E+02**
	Dev	3.66E+02	7.65E+01	2.01E+01	6.15E+01	1.59E+01	4.03E+01	6.88E+01	2.88E+01	**1.64E+01**
	Rank	9	8	6	4	7	5	2	3	**1**
F30	Mean	9.71E+04	4.75E+03	3.97E+03	3.78E+03	4.43E+03	5.44E+03	**2.32E+03**	3.13E+03	3.44E+03
	Dev	3.05+04	2.26E+04	7.52E+02	6.88E+02	5.88E+03	3.41E+02	**1.77E+02**	1.67E+02	6.17E+02
	Rank	9	7	5	4	6	8	**1**	2	3

© 中国科学院软件研究所 http://www.jos.org.cn

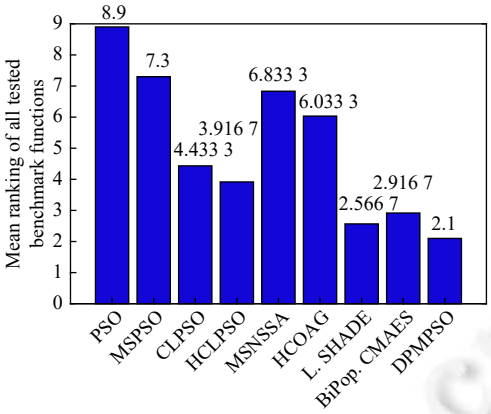

图 6 不同算法在 CEC2017 实验结果均值排序

同样为了检验表 10 中 DPMPSO 与其他算法在 CEC2017 测试集中的对比结果是否具有统计显著性, 对所有函数结果的均值项进行 Holm 检验, 表 11 记录了 Holm 检验的实验结果. 由表 11 可见, 所有对比算法中仅有 L.SHADE 没有拒绝原假设, 表明了 DPMPSO 的改进与其他算法之间对比具有显著性. 而对于冠军算法 L.SHADE 而言, 其取得了与提出的 DPMPSO 较为相似的均值排序, 在整体效果上也具备不错的竞争力.

表 11 DPMPSO 与其他对比算法的 Holm 检验结果

算法	$\alpha_{0.05}$	p-value
PSO	0.006 391 2	1.9607E–55
MSPSO	0.007 300 8	8.4836E–39
CLPSO	0.012 741	2.9647E–11
HCLPSO	0.016 952	1.4342E–07
MSNSSA	0.008 512 4	6.3646E–34
HCOAG	0.010 206	9.7268E–26
L.SHADE	0.05	0.165 89
BiPop.CMAES	0.025 321	0.015 711

通过对 L.SHADE 的自适应策略分析可以看出, 该算法通过自适应参数策略使得其在处理各类问题时均能保持较为优秀的结果. 而提出的 DPMPSO 所采用的固定的线性概率在处理部分不同的优化问题中则会出现重置后无效等问题, 这为我们下一步设计更为灵活的自适应重置策略提供了思路. 然而即便如此, 实验结果证明提出的 DPMPSO 依然取得了更好的均值排序, 这也印证了其基于密度峰值的分群与流动策略以及改进的学习策略有效的帮助算法平衡了种群的局部开采与全局勘探.

4 结 论

本文提出了基于密度峰值的依维度重置多种群粒子群算法. 该算法首先通过基于密度峰值的分群策略将整个种群划分为两个子群: 顶层群和底层群, 保证了子群内粒子的适应度处于相同水平且均匀分布在搜索空间. 然后结合改进的全面学习策略分别为两个动态子群分配不同的搜索任务: 采用带全局项全面学习策略的顶层群用于开采; 采用全面学习策略的底层群用于勘探, 进而实现整个种群在搜索过程中开采与勘探的平衡. 在此基础上设计合适的子群流动策略以保证子群在迭代中的多样性. 最后以一种线性递减的概率将适应度差的停滞粒子与全局最优解交叉重置, 实现了种群之间的信息交互并增强了每个子种群的多样性. 实验结果显示 DPMPSO 在处理多峰以及单峰问题时均取得了较好的结果, 证明本文所采用的策略不仅保证了算法具有较强的全局勘探能力, 同时也很大程度上提高了算法的局部开采能力. 然而, 在针对 CEC2017 的实验中我们发现, 在处理部分问题时尤其是复杂的复合函数时, 维度重置概率的线性递减效果并不理想, 分析结果如上文所述, 固定的线性递减维度重置概率在处理复

© 中国科学院软件研究所 http://www.jos.org.cn

杂问题时会难以兼顾不同优化问题在收敛时存在的不同收敛趋势, 从而导致算法开采精度的效果受到影响. 因此我们下一步将尝试设计具有依据种群的适应度值的变化而自适应改变的维度重置概率, 以提高算法针对不同复杂问题时的适应性与准确性.

References:

[1] Kennedy J, Eberhart R. Particle swarm optimization. In: Proc. of the 1995 Int'l Conf. on Neural Networks. Perth: IEEE, 1995. 1942–1948. [doi: 10.1109/ICNN.1995.488968]

[2] Diao X, Jiang JC, Shen GD, Chi ZZ, Wang ZR, Ni L, Mebarki A, Bian HT, Hao YM. An improved variational mode decomposition method based on particle swarm optimization for leak detection of liquid pipelines. Mechanical Systems and Signal Processing, 2020, 143: 106787. [doi: 10.1016/j.ymssp.2020.106787]

[3] Zhao XG, Liang J, Meng J, Zhou Y. An improved quantum particle swarm optimization algorithm for environmental economic dispatch. Expert Systems with Applications, 2020, 152: 113370. [doi: 10.1016/j.eswa.2020.113370]

[4] Li JH, Zong TC, Gu JP, Hua L. Parameter estimation of wiener systems based on the particle swarm iteration and gradient search principle. Circuits, Systems, and Signal Processing, 2020, 39(7): 3470–3495. [doi: 10.1007/s00034-019-01329-1]

[5] Tao XM, Liu FR, Liu Y, Tong ZJ. Multi-scale cooperative mutation particle swarm optimization algorithm. Ruan Jian Xue Bao/Journal of Software, 2012, 23(7): 1805–1815 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4128.htm [doi: 10.3724/SP.J.1001.2012.04128]

[6] Pang H, Liu F, Xu ZR. Variable universe fuzzy control for vehicle semi-active suspension system with MR damper combining fuzzy neural network and particle swarm optimization. Neurocomputing, 2018, 306: 130–140. [doi: 10.1016/j.neucom.2018.04.055]

[7] Farshi TR, Drake JH, Özcan E. A multimodal particle swarm optimization-based approach for image segmentation. Expert Systems with Applications, 2020, 149: 113233. [doi: 10.1016/j.eswa.2020.113233]

[8] 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. [doi: 10.1023/A:1008202821328]

[9] Gao WF, Liu SY, Huang LL. A novel artificial bee colony algorithm based on modified search equation and orthogonal learning. IEEE Trans. on Cybernetics, 2013, 43(3): 1011–1024. [doi: 10.1109/TSMCB.2012.2222373]

[10] Beyer HG, Schwefel HP. Evolution strategies-a comprehensive introduction. Natural Computing, 2002, 1(1): 3–52. [doi: 10.1023/A:1015059928466]

[11] Chen K, Zhou FY, Yin L, Wang SQ, Wang YG, Wan F. A hybrid particle swarm optimizer with sine cosine acceleration coefficients. Information Sciences, 2018, 422: 218–241. [doi: 10.1016/j.ins.2017.09.015]

[12] Tian DP, Shi ZZ. MPSO: Modified particle swarm optimization and its applications. Swarm and Evolutionary Computation, 2018, 41: 49–68. [doi: 10.1016/j.swevo.2018.01.011]

[13] Li NJ, Wang WJ, Hsu CCJ. Hybrid particle swarm optimization incorporating fuzzy reasoning and weighted particle. Neurocomputing, 2015, 167: 488–501. [doi: 10.1016/j.neucom.2015.04.045]

[14] Liang JJ, Qin AK, Suganthan PN, Baskar S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans. on Evolutionary Computation, 2006, 10(3): 281–295. [doi: 10.1109/TEVC.2005.857610]

[15] Zhang YF, Liu XX, Bao FX, Chi J, Zhang CM, Liu PD. Particle swarm optimization with adaptive learning strategy. Knowledge-based Systems, 2020, 196: 105789. [doi: 10.1016/j.knosys.2020.105789]

[16] Ning Y, Peng ZS, Dai YX, Bi DQ, Wang J. Enhanced particle swarm optimization with multi-swarm and multi-velocity for optimizing high-dimensional problems. Applied Intelligence, 2019, 49(2): 335–351. [doi: 10.1007/s10489-018-1258-3]

[17] Li W, Meng X, Huang Y, Fu ZH. Multipopulation cooperative particle swarm optimization with a mixed mutation strategy. Information Sciences, 2020, 529: 179–196. [doi: 10.1016/j.ins.2020.02.034]

[18] Lynn N, Suganthan PN. Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation. Swarm and Evolutionary Computation, 2015, 24: 11–24. [doi: 10.1016/j.swevo.2015.05.002]

[19] Lin AP, Sun W, Yu HS, Wu GH, Tang HW. Global genetic learning particle swarm optimization with diversity enhancement by ring topology. Swarm and Evolutionary Computation, 2019, 44: 571–583. [doi: 10.1016/j.swevo.2018.07.002]

[20] Xu GP, Cui QL, Shi XH, Ge HW, Zhan ZH, Lee HP, Liang YC, Tai R, Wu CG. Particle swarm optimization based on dimensional learning strategy. Swarm and Evolutionary Computation, 2019, 45: 33–51. [doi: 10.1016/j.swevo.2018.12.009]

[21] Zhang K, Huang QJ, Zhang YM. Enhancing comprehensive learning particle swarm optimization with local optima topology. Information Sciences, 2019, 471: 1–18. [doi: 10.1016/j.ins.2018.08.049]

© 中国科学院软件研究所 http://www.jos.org.cn

[22] Lynn N, Ali MZ, Suganthan PN. Population topologies for particle swarm optimization and differential evolution. Swarm and Evolutionary Computation, 2018, 39: 24–35. [doi: 10.1016/j.swevo.2017.11.002]

[23] Rodriguez A, Laio A. Clustering by fast search and find of density peaks. Science, 2014, 344(6191): 1492–1496. [doi: 10.1126/science.1242072]

[24] Xia XW, Gui L, Zhan ZH. A multi-swarm particle swarm optimization algorithm based on dynamical topology and purposeful detecting. Applied Soft Computing, 2018, 67: 126–140. [doi: 10.1016/j.asoc.2018.02.042]

[25] Chen ZY, Zhang DM, Xin ZY. Multi-subpopulation based symbiosis and non-uniform Gaussian mutation salp swarm algorithm. Acta Automatica Sinica, 2022, 48(5): 1307–1317 (in Chinese with English abstract). [doi: 10.16383/j.aas.c190684]

[26] Zhang XM, Jiang Y, Liu SW, Liu GQ, Dou Z, Liu Y. Hybrid coyote optimization algorithm with grey wolf optimizer and its application to clustering optimization.Acta Automatica Sinica, 2022, 48(11): 2757−2776 (in Chinese with English abstract). [doi: 10.16383/j.aas.c190617]

[27] Tanabe R, Fukunaga AS. Improving the search performance of SHADE using linear population size reduction. In: Proc. of the 2014 IEEE Congress on Evolutionary Computation (CEC). Beijing: IEEE, 2014. 1658–1665. [doi: 10.1109/CEC.2014.6900380]

[28] Hansen N. Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In: Proc. of the 11th Annual Conf. Companion on Genetic and Evolutionary Computation Conf.: Late Breaking Papers. Montreal: ACM, 2009. 2389–2396. [doi: 10.1145/1570256.1570333]

[29] Tao XM, Guo WJ, Li Q, Ren C, Liu R. Multiple scale self-adaptive cooperation mutation strategy-based particle swarm optimization. Applied Soft Computing, 2020, 89: 106124. [doi: 10.1016/j.asoc.2020.106124]

[30] Awad NH, Ali MZ, Suganthan PN, Liang JJ, Qu BY. Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization. Technical Report, Nanyang Technological University Singapore, 2016.

附中文参考文献:

[5] 陶新民, 刘福荣, 刘玉, 童智靖. 一种多尺度协同变异的粒子群优化算法. 软件学报, 2012, 23(7): 1805–1815. http://www.jos.org.cn/1000-9825/4128.htm [doi: 10.3724/SP.J.1001.2012.04128]

[25] 陈忠云, 张达敏, 辛梓芸. 多子群的共生非均匀高斯变异樽海鞘群算法. 自动化学报, 2022, 48(5): 1307–1317. [doi: 10.16383/j.aas.c190684]

[26] 张新明, 姜云, 刘尚旺, 刘国奇, 窦智, 刘艳. 灰狼与郊狼混合优化算法及其聚类优化. 自动化学报, 2022, 48(11): 2757−2776. [doi: 10.16383/j.aas.c190617]

陶新民(1973－), 男, 博士, 教授, 主要研究领域为智能信号处理, 软计算方法, 模式识别.

陈玮(1997－), 女, 硕士, 主要研究领域为不均衡数据分类, 模式识别.

郭文杰(1996－), 男, 硕士, 主要研究领域为智能优化算法, 模式识别.

吴永康(1995－), 男, 硕士, 主要研究领域为数据降维.

李向可(1993－), 男, 硕士, 主要研究领域为智能优化算法.

© 中国科学院软件研究所 http://www.jos.org.cn