中国科技期刊卓越行动计划项目入选期刊

控制与决策

CONTROL AND DECISION

带偏向性轮盘赌的多算子协同粒子群优化算法

于海波, 朱秦娜, 康丽, 乔钢柱, 曾建潮

引用本文:
于海波,朱秦娜,康丽,乔钢柱,曾建潮. 带偏向性轮盘赌的多算子协同粒子群优化算法[J]. 控制与决策, 2024, 39(4): 1167−1176.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2022.1486

带偏向性轮盘赌的多算子协同粒子群优化算法

于海波[1,2†], 朱秦娜[1,2], 康 丽[3], 乔钢柱[1,2], 曾建潮[1,2]

(1. 中北大学 计算机科学与技术学院，太原 030051；
2. 中北大学 大数据与视觉计算研究所，太原 030051；
3. 中北大学 环境与安全工程学院，太原 030051)

摘 要: 针对粒子群优化算法在处理高维、大规模、多变量耦合、多模态、多极值属性优化问题时易早熟收敛等性能和技术瓶颈,基于粒子群优化算法行为学习算子和3种不同学习偏好的差分变异算子,建立带偏向性轮盘赌的多算子选择与融合机制,提出一种带偏向性轮盘赌的多算子协同粒子群优化算法MOCPSO. MOCPSO针对迭代粒子群榜样粒子集,首先通过对迭代种群及其榜样粒子集优劣分组,同时采用轮盘赌分别为每组榜样粒子集选配不同学习偏好的变异算子,并为每组榜样粒子适配差分基向量和最优基向量,预学习并优化迭代种群及其榜样粒子,以权衡算法的全局探索和局部开发;然后通过合并所有子种群,并结合粒子群优化算法行为学习算子,指导迭代种群状态更新,以提高算法的全局收敛性;最后结合精英学习策略,对群体历史最优进行高斯扰动,以提高算法的局部逃生能力,保障算法收敛的多样性. 实验结果表明,MOCPSO算法与5种先进的同类型群智能算法在求解CEC2014基准测试问题上具备竞争力,且有更强的优化特性.

关键词: 粒子群优化；差分演化；多算子协同；榜样竞争；偏向性变异策略；精英学习

中图分类号: TP18 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2022.1486

引用格式: 于海波, 朱秦娜, 康丽, 等. 带偏向性轮盘赌的多算子协同粒子群优化算法[J]. 控制与决策, 2024, 39(4): 1167-1176.

A multi-operator collaborative particle swarm optimization algorithm with biased roulette

YU Hai-bo[1,2†], *ZHU Qin-na*[1,2], *KANG Li*[3], *QIAO Gang-zhu*[1,2], *ZENG Jian-chao*[1,2]

(1. College of Computer Science and Technology，North University of China，Taiyuan 030051，China；2. Institute of Big Data and Visual Computing，North University of China，Taiyuan 030051，China；3. College of Environment and Safety Engineering，North University of China，Taiyuan 030051，China)

Abstract: To address the performance and technical bottlenecks of a particle swarm optimization algorithm in tackling optimization problems of high-dimensional, large-scale, multivariate coupling, multi-modal, multi-extreme attribute vulnerable to premature convergence, a multi-operator selection and fusion mechanism with biased roulette is established based on the behavioral learning operator of particle swarm optimization and three differential mutation operators with different learning preferences and a multi-operator collaborative particle swarm optimization algorithm with biased roulette is proposed (MOCPSO). For balancing the exploration-exploitation trade-off, MOCPSO first groups the iterative swarm into several subswarms with different learning tasks according to the fitness, where each subswarm configures a differential mutation operator that the differential mutation vectors selected among the exemplars of all subswarms through roulette selection, to pre-learn and optimize the iterative swarm and their exemplar particles. Then all subswarms are merged undergoing behavioral learning operation of the particle swarm optimization to improve the global convergence. Finally, for guaranteeing the diversity of algorithm convergence, the MOCPSO incorporates an elitist learning strategy to guide iterative swarm escaping the possible local traps by performing Gaussian perturbation on the current global best. Experimental results show that the proposed MOCPSO algorithm possesses stronger and more competitive optimization properties than five state-of-the-art swarm intelligence algorithms in solving the CEC2014 benchmark test suit.

Keywords: particle swarm optimization；differential evolution；multi-operator coordination；exemplar competition；biased mutation strategy；elitist learning

收稿日期: 2022-08-19；**录用日期:** 2023-01-20.

基金项目: 国家自然科学基金青年基金项目(62106237)；国家自然科学基金联合基金项目(U21A20542)；山西省自然科学基金项目(201901D211237).

责任编委: 夏元清.

†通讯作者. E-mail: tyustyuhaibo@126.com.

0 引 言

粒子群优化算法(particle swarm optimization, PSO)是一种模拟自然界鸟群觅食行为的仿生群智能随机搜索算法,于1995年由Kennedy等[1]提出. PSO算法因其原理简单、参数量小、收敛速度快等优点,自提出伊始便受到广泛关注,已成功应用于如电力系统[2]、无线传感网络[3]、目标分类[4]、特征选择[5]等多种复杂优化问题. 目前,PSO算法的研究主要侧重参数的自适应调优、邻域拓扑的改进设计、学习策略的管理与控制以及与其他群智能算法架构的混合与融合.

为了平衡PSO算法的全局勘探与局部开采能力,算法控制参数的优化配置尤为重要. 在参数设置方面,Shi等[6]设计了惯性权重,Clerc等[7]提出了收缩系数,二者均能有效权衡PSO算法的全局勘探与局部开采,控制算法的收敛态势. Ratnaweera等[8]通过引入时变加速系数,提出HPSO-TVAC,实现了对算法自身认知学习和社会认知共享的自适应控制. Zhan等[9]通过模糊评估种群进化状态,设计了惯性权重和加速因子的动态调优策略,使算法学习参数自适应于种群的演化状态,提出了APSO. 不同于上述对惯性权重和加速因子的独立调控策略,马国庆等[10]直接利用惯性权重调控加速因子,通过建立并加强惯性权重与加速因子间的合作关系来平衡算法的全局勘探和局部开采能力,并据此引入随惯性权重线性调节的时间因子,进一步改善算法迭代后期的局部开采能力和收敛速度,提出了一种围绕惯性权重动态调参的PSO算法.

PSO算法的邻域拓扑尤指单个粒子与其他粒子的邻接互连方式. 邻域结构是决定PSO算法性能的重要因素之一. 配置不同邻域结构的PSO算法,其性能也呈现较大差异. 目前,典型的邻域结构包括星形、环形、球形、金字塔形及冯·诺伊曼结构等[11]. 根据问题特性,配置适当的邻域结构,有利于实现PSO算法勘探与开采的自适应控制. Nasir等[12]提出了DNLPSO算法,该算法从粒子自身邻域的最佳位置选择榜样粒子指导粒子的速度更新,并通过周期性重构粒子邻域,保障种群学习的多样性,降低算法早熟收敛的风险. Mendes等[13]提出了FIPS算法,通过加权平均邻域粒子的个体历史最优位置来更新迭代粒子的位置向量. Parsopoulos等[14]提出了UPSO,利用局部邻域和全局邻域的历史最优榜样粒子来控制算法的勘探和开采动态. 焦重阳等[15]提出了MPSO,通过根据迭代粒子群的多样性反馈信息,为迭代种群粒子选配全局或局部拓扑结构模型进行进化寻优,有效提高了算法的收敛速度.

在标准PSO算法中,个体历史最优和群体历史最优榜样粒子牵引的速度更新算子是调整迭代粒子群全局和局部寻优偏向的唯一行为学习机制. 为加强粒子行为学习的多样性,突破PSO榜样粒子一对一单一引导模式的局限,Liang等[16]提出了CLPSO算法,通过结合粒子群中最优粒子集的榜样粒子,辅助引导特定粒子的行为学习,提高粒子群学习多样性和全局收敛性. Lynn等[17]对CLPSO算法进行了改进,提出了异构CLPSO算法,将迭代种群分别划分为侧重勘探的子种群和倾向局部开采的子种群,并采用CLPSO综合学习算子指导各子种群迭代学习. 为了改善单一学习策略的局限性,融合发挥不同学习策略的优势,Li等[18]集成4种不同的学习策略提出了SLPSO算法,每个粒子可根据自身情况选配不同的学习策略迭代自学习; Wang等[19]基于反向学习策略和柯西变异,提出了GOPSO; Lim等[20]采用精英学习策略和正交学习策略辅助算法避开局部最优,提出了ATLPSO-ELS; Ouyang等[21]通过分类管理演化种群,并搭配不同学习策略进行演化,提出了IGPSO; Zhan等[22]利用正交实验设计定制了一种正交学习策略,并结合粒子个体历史最优和群体历史最优构建较优的榜样粒子作为引领粒子,指导粒子群向更好的方向靠拢,提出了OLPSO. 此外,高卫峰等[23]借鉴动态随机搜索技术,在迭代过程中引入局部搜索算子,围绕迭代粒子群当前最优位置进行局部搜索,加速定位搜索区域的最优位置,同时通过对最优位置选择性的执行倒位操作和简单操作来调控当前最优位置的停滞状态,并结合非一致性变异操作,加快算法跳出局部极值区域,良好平衡了算法的收敛速度和种群的多样性.

在PSO算法和其他群智能算法的架构混合和融合方面,Gong等[24]将遗传算法中的选择、变异和交叉算子引入到PSO算法中设计了GL-PSO算法. 针对群体中的每个粒子,利用遗传算子产生和更新榜样粒子,继而利用速度更新算子指导粒子群的行为学习,有效加速了搜索种群对最优解的定位. Zhang等[25]将差分演化混入PSO算法中,提出了DEPSO,有效提高了PSO的搜索效率. Moore等[26]选用DE/rand/2/差分变异策略和一种改进的“环”状邻域拓扑进行混合策略设计,提出了DEPSO-MV混合算法. Parouha

等[27]通过划分迭代种群,同时应用差分演化和PSO算法分组管理子种群行为学习,以优化调控迭代种群的全局和局部搜索,加快其收敛速度,提出了一种并行混合算法DE-PSO-DE. 为了增加种群多样性,平衡算法的勘探和开采能力, Th Famelis等[28]提出了一种DE-PSO算法,借用差分算子辅助PSO算法突破停滞状态. Mao等[29]提出了DEMPSO算法,借助差分演化缩小搜索范围,并通过改进的PSO算法加快收敛速度. 徐震浩等[30]有机融合PSO算法和基于排序的差分演化算法,提出了一种基于差分进化粒子群优化的间歇调度算法(DEPSO),通过利用差分演化优化迭代种群个体极值位置,增强了粒子的多样性,降低了迭代种群陷入局部极值的风险. 同样,贾时等[31] 基于协同进化理念,介绍了一种双种群多变异差分与粒子群混合算法. 利用搭配不同变异策略的双子种群经差分演化得到的新个体替代迭代种群个体最佳位置,同时结合PSO算法速度和位置更新范式,指导迭代种群的行为更新,有效增强了搜索种群的多样性.

综上所述,为平衡调优PSO算法的全局勘探和局部开采,虽然多种PSO算法的升级版本被提出,但在面对大数据背景下的高维、大规模、多变量耦合、多模态、多极值属性的复杂优化问题时,PSO算法的适用性和全局收敛性仍面临巨大挑战. 此外,在传统PSO算法中,粒子的行为学习对粒子个体历史最优和群体历史最优组建的榜样粒子集具有较强依赖性. 榜样粒子集的优劣分布和方向牵引对权衡PSO算法全局勘探和局部开采,提速迭代种群正确收敛至关重要. 为了修正PSO算法行为学习算子引致的粒子负反馈振荡现象[32],加强榜样粒子对粒子群行为学习方向的正向、正确引领,本文通过偏向性地融合3种不同学习偏好的差分变异算子,分组偏向优化PSO算法粒子群榜样粒子集,设计榜样粒子集的预学习和改进控制策略. 同时配合PSO算法行为学习算子和精英学习算子,综合权衡粒子群的全局勘探和局部开采性能,发展一种带偏向性轮盘赌的多算子协同粒子群优化算法MOCPSO.

1　粒子群优化算法和差分演化算法

1.1　粒子群优化算法

MOCPSO算法主体框架选用PSO算法作为底层搜索引擎. 在PSO算法中,粒子受自身学习经验和粒子群其他粒子学习经验的综合影响,动态调整自身的运动倾向和空间位置,循环迭代探索解空间的最优区域. 不失一般性,对于最小化问题,PSO算法中的任意一个粒子 $\boldsymbol{X}_i^G=[x_{i1}^G,x_{i2}^G,\dots x_{iD}^G]\,(i=1,2,\dots,N)$ 视作待优化问题的一个潜在解向量. 其中: N 表示粒子群规模, D 表示问题维度, G 表示第 G 次迭代. 每个粒子 $\boldsymbol{X}_i^G$ 在迭代状态转移过程中搭配一个速度向量 $\boldsymbol{V}_i^G=[v_{i1}^G,v_{i2}^G,\dots,v_{iD}^G]$,其经历过的历史最佳位置向量记为 $\textbf{pbest}_i^G=[\text{pbest}_{i1}^G,\text{pbest}_{i2}^G,\dots,\text{pbest}_{iD}^G]$. 迭代粒子群迄今为止找到的群体最佳位置向量记为 $\textbf{gbest}^G=[\text{gbest}_1^G,\text{gbest}_2^G,\dots,\text{gbest}_D^G]$. 每次迭代,粒子群中的每个粒子 $\boldsymbol{X}_i^G$ 按下式进行行为学习,实现自身速度和位置的动态更新:

$$v_{id}^{G+1}=w\cdot v_{id}^G+c_1\cdot\text{rand}_1^d\cdot(\text{pbest}_{id}^G-x_{id}^G)+c_2\cdot\text{rand}_2^d\cdot(\text{gbest}_{id}^G-x_{id}^G),\tag{1}$$

$$x_{id}^{G+1}=x_{id}^G+v_{id}^{G+1}.\tag{2}$$

其中: $d=1,2,\dots,D$ 表示第 d 维变量, $w\in[0,1.4]$ 表示惯性权重[6]; rand_1^d 和 rand_2^d 分别表示 $[0,1]$ 范围内的两个随机数; pbest_{id}^G 表示粒子 $\boldsymbol{X}_i^G$ 的个体历史最佳位置的第 d 维分量, gbest_d^G 表示群体历史最佳位置的第 d 维分量; c_1 和 c_2 表示加速系数,用以控制粒子对 $\textbf{pbest}_i$ 和 $\textbf{gbest}$ 的学习强度.

1.2　差分演化算法

差分演化算法(differential evolution, DE)[33]同样是一种基于种群的全局优化算法. 在DE算法的每次迭代,迭代种群经变异、交叉和选择算子操作,实现对种群寻优状态的动态调整,其中变异算子用以产生父代种群的变异种群. MOCPSO算法选用DE算法中3种不同学习偏好的变异算子对粒子群榜样粒子集进行偏向学习,如下列公式所示:

1) DE/rand/1:

$$\boldsymbol{V}_i^G=\boldsymbol{X}_{r_1}^G+F\cdot(\boldsymbol{X}_{r_2}^G-\boldsymbol{X}_{r_3}^G);\tag{3}$$

2) DE/best/1:

$$\boldsymbol{V}_i^G=\boldsymbol{X}_{\text{best}}^G+\boldsymbol{F}\cdot(\boldsymbol{X}_{r_1}^G-\boldsymbol{X}_{r_2}^G);\tag{4}$$

3) DE/current-to-best/1:

$$\boldsymbol{V}_i^G=\boldsymbol{X}_i^G+\boldsymbol{F}\cdot(\boldsymbol{X}_{\text{best}}^G-\boldsymbol{X}_i^G)+F\cdot(\boldsymbol{X}_{r_1}^G-\boldsymbol{X}_{r_2}^G).\tag{5}$$

其中: $\boldsymbol{X}_{r_1}^G$、$\boldsymbol{X}_{r_2}^G$、$\boldsymbol{X}_{r_3}^G$ 分别为迭代种群中随机选取且异于 $\boldsymbol{X}_i^G$ 的互异个体; $\boldsymbol{X}_{\text{best}}^G$ 为第 G 次迭代群体最佳个体,记为最优基向量 $\textbf{best}$; $F\in(0,2]$ 和 $K\in(0,1)$ 为个体扰动因子,用以控制差分向量的学习率.

不同的差分变异算子具有不同的学习偏好. DE/rand/1算子从种群中随机选择3个互异个体构造基向量和差分向量. 虽然随机选择能在一定程度

上提升种群多样性,但其搜索方向和目标较为混乱,导致算法搜索效率较低. DE/best/1算子选用群体中的最优个体作为最优基向量**best**,通过牵引种群个体偏向当前最优方向飞行,加强算法的局部开采性能,提升算法的收敛性. 以当前个体作为基向量的DE/current-to-best/1变异算子,综合了最优个体与当前个体的差分向量和两个随机互异个体的差分向量,良好权衡了算法的局部开采和全局勘探.

父代种群个体经差分变异操作产生变异种群后,运用交叉算子生成试探种群个体$\boldsymbol{U}_i^G = [u_{i1}^G, u_{i2}^G, \dots u_{iD}^G], i = 1, 2, \dots, N$. 常用的二项式交叉算子如下所示:

$$u_{ij}^G = \begin{cases} v_{ij}^G, \ \text{rand}_j \leqslant \text{CR or } j = j_{\text{rand}}; \\ x_{ij}^G, \ \text{otherwise}. \end{cases} \tag{6}$$

其中: $\text{CR} \in [0, 1]$为交叉概率, rand_j为$[0, 1]$内的随机数, $j_{\text{rand}} \in [1, D]$为随机选取的维度数.

最后,采用一对一“贪婪”选择策略,根据下式从新生成的试探种群个体和父代种群个体中优选个体构成子代种群:

$$\boldsymbol{X}_i^{G+1} = \begin{cases} \boldsymbol{U}_i^G, \ f(\boldsymbol{U}_i^G) < f(\boldsymbol{X}_i^G); \\ \boldsymbol{X}_i^G, \ \text{otherwise}. \end{cases} \tag{7}$$

2 MOCPSO算法

2.1 算法描述

MOCPSO算法采用迭代种群分层、分解、学习及合并的算法演化框架,其算法伪码见Algorithm 1. MOCPSO算法在每次迭代将迭代粒子群划分为3个子种群,并针对各个子种群的分布特点,分别选配3种不同学习偏好的差分变异算子,对子种群榜样粒子集进行预学习和优化. 同时结合PSO算法行为学习算子,基于质量更新后的榜样粒子,引导迭代粒子群的行为更新. 图1给出了MOCPSO算法榜样粒子与迭代粒子行为学习示意图. MOCPSO算法主体包含顶层榜样粒子集的偏向学习、更新和底层迭代粒子群的行为学习. 顶层榜样粒子集和底层迭代粒子群按迭代粒子群适应度优劣一对一等分分割为3个子种群. 两层粒子集以序贯方式分别预更新榜样粒子集和更新迭代粒子群. 特别地,对于顶层榜样粒子集,根据底层各组迭代子种群的平均适应值分布情况,对顶层榜样粒子进行勘探和开采任务的划分,为每组榜样粒子集分配相应的差分变异算子,并采用带偏向性轮盘赌的选择策略,分别为每个子种群偏向适配指导榜样粒子集更新的差分变异基向量. 同时,针对不同差分变异算子的构成要素,偏向性地从不同子种群榜样粒子集中,为每个差分变异算子选配最优引导基向量,建立不同子种群间的最优信息共享机制,以加强榜样粒子集学习和最优信息通信的多样性,避免算法陷入局部最优.

Algorithm 1 MOCPSO算法伪代码

输入: N, D, MaxFes, [LB, UB], $[V_{\max}, V_{\min}]$;
输出: **gbest**, fitness(**gbest**).
1) **Pop** $=$ LB $+$ rand$(N, D) \times$ (UB $-$ LB)
2) $\boldsymbol{V} = V_{\min} + \text{rand}(N, D) \times (V_{\max} - V_{\min})$
3) **Pbest** $=$ **Pop**, **gbest** $= \underset{i=1,2,\dots,N}{\arg\min}$ fitness(**pbest**$_i$)
4) while Fes $<$ MaxFes do
5) $(\text{SubPop}_A, \text{SubPop}_B, \text{SubPop}_C) = \text{Split}(\textbf{Pop})$
6) $(\text{Pbest}_A, \text{Pbest}_B, \text{Pbest}_C) = \text{Split}(\textbf{Pbest})$
7) $\boldsymbol{U}\_\textbf{pbest} = \text{Algorithm 2}(\text{Pbest}_{A,B,C}, \text{SubPop}_{A,B,C})$
8) **Pbest** $=$ Algorithm 3($\boldsymbol{U}\_$**pbest**, **Pbest**, D, CR)
9) Merge$(\text{SubPop}_A, \text{SubPop}_B, \text{SubPop}_C)$
10) Merge$(\text{Pbest}_A, \text{Pbest}_B, \text{Pbest}_C)$
11) Update $\boldsymbol{V}$, **Pop** by eqs.(1, 2)
12) Update **gbest**, **Pbest**
13) **Pop**$'$ $=$ **Algorithm 4** (**Pop**, D, **gbest**, **gworst**)
14) **Pop** $=$ **Pop**$'$
15) end while
16) return **gbest**, fitness(**gbest**)

Algorithm 2 偏向性轮盘赌变异操作伪代码

输入: $\text{Pbest}_{A,B,C}, \text{SubPop}_{A,B,C}$;
输出: $\boldsymbol{U}\_$**pbest**.
1) for $i = A \to C$ do
2) if rand$(0, 1) < p_i$ then
3) temp_pbest$_i$ $=$ SubPop$_i$//选择差分基向量集
4) end if
5) end for
6) if Pbest$_A$ then
7) select $=$ find(cum$_A$ $>=$ rand$(0, 1)$)
8) $\boldsymbol{x}_{\text{best}}^G = \text{temp\_pbest}_A(\text{select}(1))$
9) $U\_\text{pbest}_A \leftarrow$ update Pbest$_A$ by eq.(4)
10) else if Pbest$_B$ then
11) select $=$ find(cum$_B$ $>=$ rand$(0, 1)$)
12) $\boldsymbol{x}_{\text{best}}^G = \text{temp\_pbest}_B(\text{select}(1))$
13) $U\_\text{pbest}_B \leftarrow$ update Pbest$_B$ by eq.(5)
14) else
15) $U\_\text{pbest}_C \leftarrow$ update Pbest$_C$ by eq.(3)
16) end if
17) return $\boldsymbol{U}\_$**pbest** $= (U\_\text{pbest}_A, U\_\text{pbest}_B, U\_\text{pbest}_C)$

Algorithm 3 交叉和选择操作伪代码

输入: $\boldsymbol{U}\_$**gbest** $= \{U\_\text{pbest}_A, U\_\text{pbest}_B, U\_\text{pbest}_C\}$,
Pbest $= \{\text{Pbest}_A, \text{Pbest}_B, \text{Pbest}_C\}$;
输出: 更新后的榜样粒子集**Pbest**.
1) for $i = A \to C$ do
2) $U\_\text{pbest}_i = \text{Crossover}(U\_\text{pbest}_i, \text{Pbest}_i)$ //eq.(6)
3) $\text{Pbest}_i = \text{Selection}(U\_\text{pbest}_i, \text{Pbest}_i)$ //eq.(7)
4) end for
5) return **Pbest** $= (\text{Pbest}_A, \text{Pbest}_B, \text{Pbest}_C)$

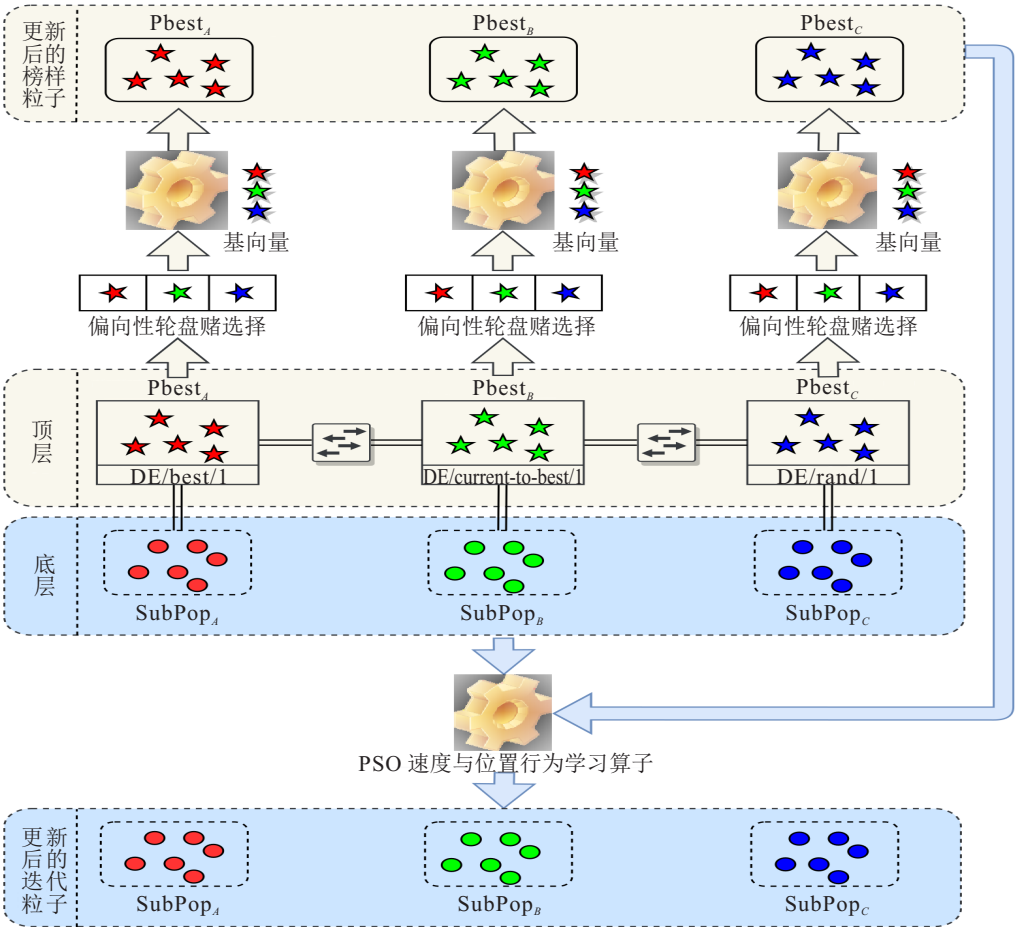

图 1　MOCPSO榜样粒子与迭代粒子行为学习示意图

通过对榜样粒子集进行预学习和优化,一方面,更新后的榜样粒子相比更新前的榜样粒子在质量上更具优势,有助于提高迭代粒子群的学习效率,帮助迭代粒子群快速定位最优区域;另一方面,基于带偏向性的轮盘赌选择策略为各组榜样粒子选择相应差分变异算子结构中不同的差分向量和最优基向量来指导其更新,有助于榜样粒子学习的多元化,助力其定位潜在的最优区域,进而辅助底层迭代粒子群逃离局部极值区域. 再者,该策略为顶层榜样粒子集的学习选配了3种具有不同学习偏好的差分变异算子,有利于深度融合榜样粒子集的分布特点和搜索偏向,协作分工,进一步改善迭代粒子群学习的收敛性和多样性. 此外,MOCPSO算法在每次迭代过程中,采用精英学习算子对当前群体历史最优进行高斯扰动,以规避算法早熟收敛. 事实上,群体历史最优对迭代粒子群的状态转移起着重要牵引作用. 然而,群体历史最优粒子缺乏经验和自我学习能力,其依靠自身在迭代过程中无法获得质量改进. 若当前群体历史最优粒子未能有效迭代更新,则整个种群的寻优能力和种群的多样性将会显著降低,致使PSO算法面临早熟收敛的风险.

2.2　带偏向性轮盘赌的多算子协同学习策略

MOCPSO算法基于3种不同学习偏好的差分变异算子,通过利用迭代粒子群粒子的分布特性,结合轮盘赌选择策略,设计了一种预学习和优化迭代粒子群榜样粒子集的带偏向性轮盘赌的多算子协同学习策略. 不失一般性,以最小化问题为例,MOCPSO算法在每次迭代按适应值优劣,将迭代粒子群等分分割为优、中、劣3组不同角色的子种群SubPop_A、SubPop_B和SubPop_C. 其中: $f_{\mathrm{mean}}(\mathrm{SubPop}_A) < f_{\mathrm{mean}}(\mathrm{SubPop}_B) < f_{\mathrm{mean}}(\mathrm{SubPop}_C)$, $f_{\mathrm{mean}}(\cdot)$表示子种群的平均适应度值.

为了权衡各组子种群的搜索任务,根据各组子种群的平均适应值分布,对3组子种群进行任务划分. 一般而言,平均适应值较优的子种群SubPop_A倾向解空间的局部开采,以充分挖掘局部最优信息,进一步提高榜样粒子集和最优解的质量;相反,对于适应值较劣的子种群SubPop_C,需更侧重解空间的全局勘探,以期望充分探索解空间潜在最优区域,找到更具发展潜力的最优解,以提高榜样粒子分布的多样

性;对于介于两者之间的适应值表现相对平庸的子种群SubPop_B,较为适合平衡迭代种群对解空间的全局勘探和局部开采,为其他两组子种群输送新的较优或较劣的替代个体. 基于不同子种群搜索任务的分配,以偏向性轮盘赌选择的方式,为每组子种群榜样粒子适配搜索偏好相似的差分变异算子和差分基向量. 其中,3组子种群SubPop_A、SubPop_B和SubPop_C对应的榜样粒子集Pbest_A、Pbest_B和Pbest_C分别适配如下3种差分变异算子与参数配置方案:

1) Pbest_A: [DE/best/1, $F=0.6$, CR $=0.1$];

2) Pbest_B: [DE/current-to-best/1, $F=0.8$, CR $=0.3$];

3) Pbest_C: [DE/rand/1, $F=1.0$, CR $=0.9$].

MOCPSO算法分别运行两次轮盘赌选择从相应榜样粒子集中选择差分基向量和最优基向量. Algorithm 2给出了偏向性轮盘赌变异操作伪代码. 首先,根据3组子种群SubPop_A、SubPop_B和SubPop_C的平均适应度值计算选择概率p_A、p_B和p_C,利用轮盘赌为每组子种群相应榜样粒子集Pbest_A、Pbest_B和Pbest_C偏向选择差分基向量集;然后,根据适应值优劣比例,计算各组差分基向量选择概率cum_A和cum_B,并基于相应差分变异算子结构特征,再次利用轮盘赌选择操作为顶层榜样粒子集Pbest_A和Pbest_B选定差分变异算子最优基向量$\boldsymbol{X}_{\text{best}}^G$;最后,对$\text{Pbest}_A$、$\text{Pbest}_B$和$\text{Pbest}_C$执行变异操作生成新的榜样粒子$\boldsymbol{U}$_**pbest**.

榜样粒子集经差分变异后,继续对其执行交叉和选择运算更新,相关运算伪代码见Algorithm 3. 交叉操作更新后的榜样粒子集U_pbest_A、U_pbest_B和U_pbest_C分别与Pbest_A、Pbest_B和Pbest_C一对一竞争选择,更新迭代粒子群榜样粒子集和全局最优. 这里通过选择操作确保了榜样粒子集在每一代中以大概率向最优区域定向学习更新,间接保证了底层迭代粒子群的学习收敛性;同时以小概率向其他榜样粒子所属局部最优区域靠拢拉伸,保证了底层迭代粒子群的学习多样性.

顶层迭代粒子群榜样粒子集更新后,合并所有底层子种群,并利用式(1)和(2)更新底层迭代粒子群,学习产生下一代新的迭代种群.

2.3 精英学习策略

为引导全局最优粒子**gbest**向最优区域飞行,辅助其跳出潜在的局部极值区域,MOCPSO算法采用精英学习算子对其进行高斯扰动,以驱动其状态转移和更新. Algorithm 4给出了精英学习策略伪代码.

Algorithm 4 精英学习策略伪代码

输入: **Pop** $=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_N\}$, D, **gbest**, **gworst**;
输出: **Pop**′.
1) $\boldsymbol{P}=$ **gbest**
2) $d=\text{rand}(D)$
3) $\boldsymbol{P}^d \leftarrow \boldsymbol{P}^d+(x_{\max}^d-x_{\min}^d)\cdot\text{Gaussian}(\mu,\sigma^2)$
4) $f_p=\text{fitness}(\boldsymbol{P})$
5) $f_g=\text{fitness}(\textbf{gbest})$
6) $f_w=\text{fitness}(\textbf{gworst})$
7) if $f_p <= f_g$
8) **gbest** $\leftarrow \boldsymbol{P}$
9) else
10) **gworst** $\leftarrow \boldsymbol{P}$
11) end if
12) return **Pop**′

本文随机选取**gbest**的一个维度进行扰动,计算范式[9]如下所示:

$$P^d=P^d+(X_{\max}^d-X_{\min}^d)\cdot\text{Gaussian}(\mu,\sigma^2), \tag{8}$$

$$\sigma=\sigma_{\max}-(\sigma_{\max}-\sigma_{\min})\cdot t/T. \tag{9}$$

其中: P^d表示**gbest**的第d个维度; $\text{Gaussian}(\mu,\sigma^2)$表示均值为$\mu$、标准差为$\sigma$的高斯随机整数,$\sigma$为随迭代次数动态调整的“精英学习率”,$\sigma_{\max}=1.0$和$\sigma_{\min}=0.1$分别为$\sigma$的上界和下界[9]; $\boldsymbol{X}_{\max}^d$和$\boldsymbol{X}_{\min}^d$分别表示解空间第d维变量的上界和下界. 若经高斯变异后的精英粒子的适应值优于**gbest**的适应值,则更新**gbest**;反之,则替换迭代粒子群中适应度最差的粒子**gworst**.

3 实验设计与结果分析

本节检验MOCPSO算法的有效性和求解效率. 首先,给出所选测试问题;然后,给出相关算法的参数配置和实验环境,分析比较MOCPSO算法与5种同类型代表性算法的性能差异.

3.1 测试问题

实验选用CEC2014测试集[34]验证所提算法的有效性. 表1列举了CEC2014基准测试问题的基本特征,包括单峰、多峰、混合及复合4大类特性.

3.2 参数配置及竞争算法

为验证MOCPSO算法的有效性,本节选取时下具有代表性的4种改进PSO算法(包括CLPSO[16]、TSLPSO[35]、GLPSO[24]、DMSPSO[36])以及一种改进DE算法MRDE[37]展开性能比较实验. 各对比算法的主要参数设置如表2所示,其他参数配置与源文献保持一致. 所有算法最大函数评价次数MaxFes $=300\,000$,且均独立运行20次. 各算法种群规模均设置

表 1　CEC2014基准测试函数

函数特征	No.	函数名称	最优解
unimodal functions	f_1	rotated high conditioned elliptic function	100
	f_2	rotated bent cigar function	200
	f_3	rotated discus function	300
simple multimodal functions	f_4	shifted and rotated Rosenbrock's function	400
	f_5	shifted and rotated Ackley's function	500
	f_6	shifted and rotated Weierstrass function	600
	f_7	shifted and rotated Griewank's function	700
	f_8	shifted Rastrigin's function	800
	f_9	shifted and rotated Rastrigin's function	900
	f_{10}	shifted Schwefel's function	1 000
	f_{11}	shifted and rotated Schwefel's function	1 100
	f_{12}	shifted and rotated Katsuura function	1 200
	f_{13}	shifted and rotated HappyCat function	1 300
	f_{14}	shifted and rotated Hgbat function	1 400
	f_{15}	shifted and rotated expanded Griewank's plus Rosenbrock's function	1 500
	f_{16}	shifted and rotated expanded Scaffer's F6 function	1 600
hybrid functions	f_{17}	hybrid function 1 ($N=3$)	1 700
	f_{18}	hybrid function 2 ($N=3$)	1 800
	f_{19}	hybrid function 3 ($N=4$)	1 900
	f_{20}	hybrid function 4 ($N=4$)	2 000
	f_{21}	hybrid function 5 ($N=5$)	2 100
	f_{22}	hybrid function 6 ($N=5$)	2 200
composition functions	f_{23}	composition function 1 ($N=5$)	2 300
	f_{24}	composition function 2 ($N=3$)	2 400
	f_{25}	composition function 3 ($N=3$)	2 500
	f_{26}	composition function 4 ($N=5$)	2 600
	f_{27}	composition function 5 ($N=5$)	2 700
	f_{28}	composition function 6 ($N=5$)	2 800
	f_{29}	composition function 7 ($N=3$)	2 900
	f_{30}	composition function 8 ($N=3$)	3 000

为$N=30$, 问题维数$D=30$. 实验环境采用Matlab R2017a仿真软件, 各算法均在配置Intel(R) Core(TM) i5-7300HQ CPU @ 2.50 GHz和16.0 GB内存的台式机上运行.

表 2　算法参数配置表

算法	主要参数
CLPSO	$w=0.729\,8, c=1.494\,45$
TSLPSO	$c_1=c_2=1.494\,45, c_3: 0.5\sim 2.5$
GLPSO	$w=0.729\,8, c_1=c_2=1.494\,45, \mathrm{CR}=0.5$
MRDE	$F=[0.6, 0.8, 1.0], \mathrm{CR}=[0.1, 0.5, 0.9]$
DMSPSO	$w=0.729, c_1=c_2=1.494\,45$
MOCPSO	$w=0.729\,8, c_1=c_2=1.494\,45$

3.3　算法性能对比及结果分析

本小节比较MOCPSO算法与5种竞争算法求解30维测试函数的优化性能. 表3给出了各竞争算法所获最优值统计量, 包括各算法所获最优解的均值和标准差、Friedman排序以及在95%置信水平下的双侧Wilcoxon秩和检验结果. 表3中, 符号"+""−"和"="分别表示MOCPSO算法在相关问题的优化求解上显著优于、显著劣于或等同于对比算法的求解性能. 此外, "+/ = /−"表明MOCPSO算法在全部测试问题中优于、劣于和对等于其他对比算法的问题数量.

如表3所示, 对于单模态问题$f_1\sim f_3$, MOCPSO算法较其他算法可获得更优数量级的最优解. 这主要得益于MOCPSO算法的双层序贯学习策略, 通过偏向协同多种变异算子对顶层榜样粒子集预学习分组优化, 使顶层榜样粒子的质量得到明显提升, 尤其是在单模态地貌空间, 对顶层榜样粒子的多算子协同学习更有利于驱使顶层榜样粒子快速靠拢至全局最优解邻域, 从而进一步引导底层迭代粒子群定位全局最优解, 加速算法收敛. 对于f_1, MOCPSO算法性能排名第2, 其获取的最优解略差于GLPSO算法, 但仍优于其余4种对比算法. 对于7个多模态多极值问题$f_5\sim f_{16}$, MOCPSO算法较其他5种竞争算法在求解问题f_5、f_6、f_7、f_9、f_{10}和f_{11}、f_{13}、f_{14}、f_{15}时可获得较优的最优解, 表明了MOCPSO算法中多算

表3 各竞争算法求解30维测试问题的最优值统计量

No.	CLPSO	TSLPSO	GLPSO	MRDE	DMSPSO	MOCPSO
f_1	5.71e+06 (2.26e+06)(+)	4.18e+05(2.93e+05)(=)	**1.21e+05(9.57e+04)(−)**	1.01e+06(6.99e+05)(+)	5.45e+06(2.17e+06)(+)	4.17e+05(3.40e+05)
f_2	2.06e+02(3.40e+02)(+)	2.72e+02(7.82e+02)(+)	3.07e+02(4.33e+02)(+)	1.45e+04(1.05e+04)(+)	6.24e-02(9.99e-02)(+)	**3.84e-14(1.39e-14)**
f_3	1.33e+02(1.62e+02)(+)	1.24e+02(2.36e+02)(+)	5.38e+02(5.26e+02)(+)	6.97e+02(5.28e+02)(+)	1.81e+00(2.58e+00)(+)	**4.18e-08(1.72e-07)**
f_4	7.25e+01(2.04e+01)(+)	5.34e+01(4.26e+01)(+)	**3.70e+00(1.44e+01)(−)**	7.62e+01(4.43e+01)(+)	4.36e+00(1.55e+01)(=)	1.29e+01(2.52e+01)
f_5	2.04e+01(4.45e-02)(+)	2.00e+01(9.86e-04)(−)	2.09e+01(4.16e-02)(+)	2.01e+01(1.94e-01)(=)	2.00e+01(1.41e-03)(−)	2.03e+01(6.13e-02)
f_6	1.12e+01(1.63e+00)(+)	9.28e+00(2.89e+00)(+)	6.51e+00(2.16e+00)(+)	6.54e+00(2.30e+00)(+)	**1.44e+00(1.15e+00)(−)**	3.18e+00(2.65e+00)
f_7	7.35e-07(1.30e-06)(+)	1.55e-05(2.96e-05)(+)	7.50e-03(1.15e-02)(+)	9.85e-02(1.02e-01)(+)	1.01e-03(3.10e-03)(+)	**1.42e-13(8.14e-14)**
f_8	**1.08e-13(2.54e-14)(−)**	2.16e-13(3.50e-14)(−)	1.19e-13(2.54e-14)(−)	1.44e-09(6.42e-09)(−)	1.02e+01(6.98e+00)(+)	1.93e+00(1.09e+00)
f_9	4.80e+01(8.77e+00)(=)	4.78e+01(1.10e+01)(=)	**3.82e+01(8.96e+00)(−)**	4.72e+01(1.03e+01)(=)	4.43e+01(6.40e+00)(=)	4.58e+01(1.15e+01)
f_{10}	1.88e+00(1.71e+00)(−)	1.97e+00(1.56e+00)(−)	**8.75e-01(9.62e-01)(−)**	1.03e+00(7.64e-01)(−)	3.96e+02(1.44e+02)(+)	7.80e+00(7.47e+00)
f_{11}	2.24e+03(2.69e+02)(=)	2.09e+03(4.60e+02)(=)	2.05e+03(3.57e+02)(=)	**2.03e+03(4.54e+02)(=)**	2.60e+03(3.29e+02)(+)	2.22e+03(5.15e+02)
f_{12}	4.03e-01(8.16e-02)(+)	1.50e-01(4.03e-02)(=)	2.51e+00(3.02e-01)(+)	**7.89e-02(3.25e-02)(−)**	4.48e-01(7.85e-02)(+)	1.46e-01(8.31e-02)
f_{14}	2.68e-01(2.44e-02)(−)	2.34e-01(3.70e-02)(−)	3.41e-01(1.02e-01)(=)	2.56e-01(3.54e-02)(−)	**2.00e-01(3.57e-02)(−)**	3.04e-01(3.03e-02)
f_{15}	7.21e+00(1.30e+00)(+)	5.27e+00(1.56e+00)(+)	**2.98e+00(5.76e-01)(−)**	5.84e+00(2.32e+00)(+)	6.24e+00(8.44e-01)(+)	4.00e+00(1.03e+00)
f_{16}	1.03e+01(2.96e-01)(+)	1.00e+01(5.38e-01)(=)	1.06e+01(8.62e-01)(+)	**9.01e+00(7.30e-01)(−)**	1.07e+01(4.40e-01)(+)	9.77e+00(7.12e-01)
f_{17}	5.40e+05(2.65e+05)(+)	1.69e+05(1.34e+05)(+)	**2.26e+04(1.31e+04)(=)**	1.01e+05(1.58e+05)(=)	2.69e+05(1.01e+05)(+)	5.83e+04(9.34e+04)
f_{18}	**1.07e+02(6.56e+01)(−)**	1.89e+02(1.17e+02)(−)	1.49e+03(2.03e+03)(=)	3.54e+03(3.57e+03)(=)	1.00e+03(7.36e+02)(=)	2.74e+03(3.63e+03)
f_{19}	7.04e+00(7.51e-01)(+)	5.53e+00(1.36e+00)(+)	8.12e+00(1.32e+01)(+)	6.39e+00(1.96e+00)(+)	5.97e+00(6.71e-01)(+)	**4.31e+00(1.69e+00)**
f_{20}	1.96e+03(1.15e+03)(+)	7.22e+02(6.83e+02)(=)	5.64e+03(3.30e+03)(+)	4.77e+02(7.30e+02)(=)	4.51e+02(1.30e+02)(+)	**3.11e+02(2.67e+02)**
f_{21}	7.56e+04(3.96e+04)(+)	4.68e+04(4.21e+04)(+)	1.58e+04(9.52e+03)(+)	**1.01e+04(2.90e+04)(−)**	9.37e+04(5.03e+04)(+)	1.34e+04(2.89e+04)
f_{22}	**1.79e+02(7.84e+01)(−)**	2.29e+02(1.02e+02)(−)	2.96e+02(1.17e+02)(=)	2.82e+02(1.81e+02)(=)	2.30e+02(8.24e+01)(−)	3.58e+02(2.14e+02)
f_{23}	3.15e+02(1.00e-08)(+)	3.15e+02(4.64e-12)(+)	3.15e+02(6.74e-13)(+)	3.15e+02(3.36e-03)(+)	3.15e+02(0.00e+00)(+)	**3.15e+02(0.00e+00)**
f_{24}	2.25e+02(5.90e-01)(+)	2.25e+02(1.05e+00)(+)	2.25e+02(1.35e+00)(+)	2.31e+02(4.71e+00)(+)	**2.23e+02(9.93e-01)(=)**	2.24e+02(1.03e+00)
f_{25}	2.07e+02(6.29e-01)(+)	2.06e+02(1.41e+00)(+)	2.10e+02(3.66e+00)(+)	2.05e+02(2.57e+00)(+)	2.08e+02(1.31e+00)(+)	**2.04e+02(9.41e-01)**
f_{26}	**1.00e+02(6.31e-02)(=)**	**1.00e+02(6.43e-02)(=)**	1.45e+02(5.09e+01)(=)	**1.00e+02(6.21e-02)(=)**	1.40e+02(5.02e+01)(=)	1.25e+02(4.43e+01)
f_{27}	4.12e+02(7.13e+00)(−)	4.02e+02(1.04e+01)(=)	4.83e+02(8.87e+01)(+)	4.03e+02(1.49e+00)(=)	**3.76e+02(4.80e+01)(=)**	4.30e+02(6.16e+01)
f_{28}	8.80e+02(6.44e+01)(=)	9.16e+02(6.95e+01)(=)	8.83e+02(5.52e+01)(=)	8.71e+02(6.23e+01)(=)	**8.69e+02(3.03e+01)(=)**	9.42e+02(1.49e+02)
f_{29}	9.81e+02(1.02e+02)(=)	4.23e+03(1.12e+04)(+)	1.44e+03(4.50e+02)(+)	1.44e+03(5.69e+02)(=)	1.55e+03(3.35e+02)(+)	1.17e+03(4.43e+02)
f_{30}	2.47e+03(6.36e+02)(=)	7.83e+03(5.24e+03)(+)	2.26e+03(5.79e+02)(=)	2.29e+03(1.03e+03)(=)	**1.86e+03(4.79e+02)(=)**	2.06e+03(9.26e+02)
Friedman rank	3.95	3.40	3.77	3.60	3.33	**2.95**
+/=/−	17/7/6	14/9/7	15/8/7	11/13/6	17/8/5	−

子协同学习策略在保持种群多样性、稳定解空间全局探索方面的有效性. 然而, 由表3同样可以注意到, 对于F_8, MOCPSO算法所获最优解远不及CLPSO、TSLPSO、 GLPSO和MRDE得到的最优解, 仅相对DMSPSO算法具有显著优势. 其原因在于等分分割子种群在一定程度上限制了迭代种群对解空间部分狭小区域的覆盖和搜索, 同时采用固定参数配置的变异算子, 未能良好适应复杂地貌空间特征, 从而降低了MOCPSO算法的整体效能. 尽管如此, MOCPSO算法在绝大多数多模态多极值测试问题上较其他对比算法具有较强竞争力.

对于混合函数$f_{17}\sim f_{22}$, 由表3可见: MOCPSO算法在f_{17}和f_{21}上排名第2; 在f_{19}和f_{20}上排名第1, 其最优解显著优于其他5种算法, 表明MOCPSO算法通过对顶层榜样粒子集进行偏向预学习和优化, 有效改善了榜样粒子集在解空间分布的多样性, 提升了其对底层迭代粒子群行为学习引导的多样性, 增强了算法对复杂地貌空间的全局探索能力. 对于问题f_{18}, MOCPSO算法获得的最优解较TSLPSO、DMSPSO、 MRDE算法没有显著性差异, 但是略差于CLPSO和TSLPSO算法. 其原因在于CLPSO和TSLPSO算法相比MOCPSO算法具有更强的抵抗局部极值捕获的能力. 事实上, CLPSO和TSLPSO算法通过继承和融合优秀榜样粒子信息, 间接扩大了迭代粒子在解空间的探索步长, 提高了迭代种群的多样性和全局搜索能力. 其中值得注意的是, TSLPSO算法在优化过程中选用CLPSO算法行为学习算子引导子种群粒子的状态更新. 另外, 在问题f_{22}上, MOCPSO算法效果较差, 但对于复杂复合函数$f_{23}\sim f_{30}$, MOCPSO算法获得最优解在f_{23}和f_{25}上位列第1, 在f_{24}、f_{29}、f_{30}位列第2. 特别地, 在f_{26}上, MOCPSO算法的最优解显著优于DMSPSO和GLPSO算法的最优解, 且在f_{27}上, MOCPSO算法优于GLPSO算法. 这些结果表明, MOCPSO算法的求

解性能随着问题复杂度的增加保持了良好的鲁棒性.

此外, 由Friedman排序结果可见, MOCPSO算法获得最小秩, 表明了MOCPSO算法在30维CEC 2014测试集上较对比算法具有更佳的综合性能. 由双侧Wilcoxon秩和检验结果可知, 相比4种PSO变种算法, MOCPSO算法在至少14个问题上优势显著, 在至多7个问题上表现较差. 相比MRDE算法, MOCPSO算法在11个测试问题上具有显著优越性, 而MRDE算法仅在6个问题略优于MOCPSO算法. 整体而言, 本文所提算法可以获得较其他算法相近甚至更优的收敛结果.

4 结 论

本文提出了一种带偏向性轮盘赌的多算子协同粒子群优化算法MOCPSO来修正PSO算法速度更新算子引致的粒子负反馈振荡现象. MOCPSO算法按照迭代粒子群分层、分解、学习及合并的算法设计思想, 结合双重轮盘赌选择策略, 偏向性搭配3种不同学习偏好的差分变异算子, 协同指导迭代粒子群及对应榜样粒子集的行为更新, 制定了带偏向性轮盘赌的多算子协同学习策略, 同时结合精英学习策略辅助MOCPSO算法跳出局部极值. 实验结果表明, 相比其他5种同类型PSO和DE变种算法, MOCPSO算法在求解CEC2014中不同模态测试问题上具有显著优势, 说明了MOCPSO算法的有效性和较强的优化效率.

诚然, 所提算法在求解部分特殊地貌的多极值多模态问题上效果欠佳, 并且不同差分变异算子的调参偏经验化. 在后续工作中将考虑结合强化学习奖惩因子, 辅助设计多算子参数自适应调优的混合算法, 并将其应用于解决实际工程优化问题.

参考文献(References)

[1] Kennedy J, Eberhart R. Particle swarm optimization[C]. Proceedings of ICNN'95 — International Conference on Neural Networks. Perth, 2002: 1942-1948.

[2] del Valle Y, Venayagamoorthy G K, Mohagheghi S, et al. Particle swarm optimization: Basic concepts, variants and applications in power systems[J]. IEEE Transactions on Evolutionary Computation, 2008, 12(2): 171-195.

[3] Kulkarni R V, Venayagamoorthy G K. Particle swarm optimization in wireless-sensor networks: A brief survey[J]. IEEE Transactions on Systems, Man, and Cybernetics — Part C: Applications and Reviews, 2011, 41(2): 262-267.

[4] Tran B, Xue B, Zhang M J. Variable-length particle swarm optimization for feature selection on high-dimensional classification[J]. IEEE Transactions on Evolutionary Computation, 2019, 23(3): 473-487.

[5] Tan T Y, Zhang L, Neoh S C, et al. Intelligent skin cancer detection using enhanced particle swarm optimization[J]. Knowledge-Based Systems, 2018, 158: 118-135.

[6] Shi Y, Eberhart R. A modified particle swarm optimizer[C]. 1998 IEEE International Conference on Evolutionary Computation Proceedings. Anchorage, 2002: 69-73.

[7] Clerc M, Kennedy J. The particle swarm — Explosion, stability, and convergence in a multidimensional complex space[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(1): 58-73.

[8] Ratnaweera A, Halgamuge S K, Watson H C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 240-255.

[9] Zhan Z H, Zhang J, Li Y, et al. Adaptive particle swarm optimization[J]. IEEE Transactions on Systems, Man, and Cybernetics — Part B: Cybernetics, 2009, 39(6): 1362-1381.

[10] 马国庆, 李瑞峰, 刘丽. 学习因子和时间因子随权重调整的粒子群算法[J]. 计算机应用研究, 2014, 31(11): 3291-3294.
(Ma G Q, Li R F, Liu L. A particle swarm algorithm for learning factor and time factor adjustment with weights[J]. Computer Application Research, 2014, 31(11): 3291-3294.)

[11] Kennedy J. Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance[C]. Proceedings of the 1999 Congress on Evolutionary Computation. Washington, 2002: 1931-1938.

[12] Nasir M, Das S, Maity D, et al. A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization[J]. Information Sciences, 2012, 209: 16-36.

[13] Mendes R, Kennedy J, Neves J. The fully informed particle swarm: Simpler, maybe better[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 204-210.

[14] Parsopoulos K E, Vrahatis M N. UPSO: A unified particle swarm optimization scheme[C]. International Conference of Computational Methods in Sciences and Engineering 2004. Boca Raton: CRC Press, 2019: 868-873.

[15] 焦重阳, 周清雷, 张文宁. 混合拓扑结构的粒子群算法及其在测试数据生成中的应用研究[J]. 计算机科学, 2017, 44(12): 249-254.
(Jiao C Y, Zhou Q L, Zhang W N. MPSO and its application in test data automatic generation[J]. Computer Science, 2017, 44(12): 249-254.)

[16] Liang J J, Qin A K, Suganthan P N, et al. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions[J]. IEEE Transactions on Evolutionary Computation, 2006, 10(3): 281-295.

[17] Lynn N, Suganthan P N. Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation[J]. Swarm and Evolutionary Computation, 2015, 24: 11-24.

[18] Li C H, Yang S X, Nguyen T T. A self-learning particle swarm optimizer for global optimization problems[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2012, 42(3): 627-646.

[19] Wang H, Wu Z J, Rahnamayan S, et al. Enhancing particle swarm optimization using generalized opposition-based learning[J]. Information Sciences, 2011, 181(20): 4699-4714.

[20] Lim W H, Mat Isa N A. An adaptive two-layer particle swarm optimization with elitist learning strategy[J]. Information Sciences, 2014, 273: 49-72.

[21] Ouyang H B, Gao L Q, Li S, et al. Improved global-best-guided particle swarm optimization with learning operation for global optimization problems[J]. Applied Soft Computing, 2017, 52: 987-1008.

[22] Zhan Z H, Zhang J, Li Y, et al. Orthogonal learning particle swarm optimization[C]. IEEE Transactions on Evolutionary Computation. Piscataway: IEEE, 2010: 832-847.

[23] 高卫峰, 刘三阳. 一种高效粒子群优化算法[J]. 控制与决策, 2011, 26(8): 1158-1162.
(Gao W F, Liu S Y. An efficient particle swarm optimization[J]. Control and Decision, 2011, 26(8): 1158-1162.)

[24] Gong Y J, Li J J, Zhou Y C, et al. Genetic learning particle swarm optimization[J]. IEEE Transactions on Cybernetics, 2015, 46(10): 2277-2290.

[25] Zhang W J, Xie X F. DEPSO: Hybrid particle swarm with differential evolution operator[C]. 2003 IEEE International Conference on Systems, Man and Cybernetics. Washington, 2003: 3816-3821.

[26] Moore P W, Venayagamoorthy G K. Evolving digital circuits using hybrid particle swarm optimization and differential evolution[J]. International Journal of Neural Systems, 2006, 16(3): 163-177.

[27] Parouha R P, Das K N. DPD: An intelligent parallel hybrid algorithm for economic load dispatch problems with various practical constraints[J]. Expert Systems with Applications, 2016, 63: 295-309.

[28] Th Famelis I, Alexandridis A, Tsitouras C. A highly accurate differential evolution—Particle swarm optimization algorithm for the construction of initial value problem solvers[J]. Engineering Optimization, 2018, 50(8): 1364-1379.

[29] Mao B Y, Xie Z J, Wang Y B, et al. A hybrid strategy of differential evolution and modified particle swarm optimization for numerical solution of a parallel manipulator[J]. Mathematical Problems in Engineering, 2018, 2018: 1-9.

[30] 徐震浩, 李青青, 顾幸生. 基于DEPSO的模糊时间ZW多产品厂间歇调度[J]. 控制与决策, 2015, 30(12): 2275-2279.
(Xu Z H, Li Q Q, Gu X S. Fuzzy time ZW multi-product plant intermittent scheduling based on DEPSO[J]. Control and Decision, 2015, 30(12): 2275-2279.)

[31] 贾时, 梁晓丹, 何茂伟, 等. 一种新的双种群多变异差分与粒子群混合算法[J]. 计算机应用研究, 2020, 37(S2): 44-46.
(Jia S, Liang X D, He M W, et al. A new two-population multivariate difference and particle swarm hybrid algorithm[J]. Computer Application Research, 2020, 37(S2): 44-46.)

[32] van den Bergh F, Engelbrecht A P. A Cooperative approach to particle swarm optimization[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 225-239.

[33] 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.

[34] Liang J J, Qu B Y, Suganthan P N. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization[R]. Zhengzhou: Zhengzhou University and Singapore: Nanyang Technological University, 2013.

[35] Xu G P, Cui Q L, Shi X H, et al. Particle swarm optimization based on dimensional learning strategy[J]. Swarm and Evolutionary Computation, 2019, 45: 33-51.

[36] Liang J J, Suganthan P N. Dynamic multi-swarm particle swarm optimizer[C]. Proceedings 2005 IEEE Swarm Intelligence Symposium. Pasadena, 2005: 124-129.

[37] Gui L, Xia X, Yu F, et al. A multi-role based differential evolution[J]. Swarm and Evolutionary Computation, 2019, 50: 100508.

作者简介

于海波(1990−), 男, 讲师, 博士, 从事进化计算、机器学习、代理模型辅助的群体智能优化算法及其应用等研究, E-mail: tyustyuhaibo@126.com;

朱秦娜(2000−), 女, 博士生, 从事进化计算、深度神经网络、代理模型辅助的进化优化等研究, E-mail: 1790945653@qq.com;

康丽(1990−), 女, 讲师, 博士, 从事环境功能材料的构建、新型可持续能源催化转化新技术等研究, E-mail: kl199047@163.com;

乔钢柱(1975−), 男, 教授, 博士, 从事进化计算、分布式计算、大数据处理技术等研究, E-mail: 20170010@nuc.edu.cn;

曾建潮(1963−), 男, 教授, 博士生导师, 从事计算智能、代理模型优化、复杂系统的维护决策与健康管理等研究, E-mail: 20160002@nuc.edu.cn.