基于多尺度选择性学习和探测-收缩机制的PSO算法

夏学文[1,2],桂 凌[3],戴志锋[4],谢承旺[1,2],魏 波[1,2]
(1.华东交通大学软件学院,江西南昌,330013;2.华东交通大学智能优化与信息处理研究所,江西南昌,330013;
3.华东交通大学经济管理学院,江西南昌,330013;4.湖北经济学院信息管理学院,湖北武汉,430205)

摘 要: 针对粒子群算法逃离局部最优能力差、易早熟收敛、求解精度低等缺点,提出了一种具有多尺度选择性学习和探测-收缩机制的PSO算法.在多尺度选择性学习机制中,粒子根据其自身进化状态在拓扑结构、邻居个体、目标变量维等多个尺度上进行选择性学习,提升粒子个体的学习效率;在探测-收缩机制中,算法利用历史信息指导种群最优解进行探测,提高其逃离局部最优的能力,当判断种群历史最优解处于全局最优解附近时,执行空间收缩策略,将种群的搜索空间限定在较小的一个区域,增强算法的开采能力,提高算法的求解精度.通过和其它PSO算法在22个典型测试函数的实验对比表明,本算法能有效克服早熟收敛、加快收敛速度、提高求解精度.

关键词: 粒子群算法; 早熟收敛; 多尺度学习; 探测策略
中图分类号: TP301 **文献标识码:** A **文章编号:** 0372-2112 (2016)05-1090-11
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2016.05.012

A PSO Algorithm Based on Multiscale-Selective-Learning and Detecting-Shrinking Strategies

XIA Xue-wen[1,2], GUI Ling[3], DAI Zhi-feng[4], XIE Cheng-wang[1,2], WEI Bo[1,2]
(1. *School of Software, East China Jiaotong University, Nanchang, Jiangxi* 330013, *China*;
2. *Intelligent Optimization & Information Processing Lab, East China Jiaotong University, Nanchang, Jiangxi* 330013, *China*;
3. *School of Economics and Management, East China Jiaotong University, Nanchang, Jiangxi* 330013, *China*
4. *School of Information Management, Hubei University of Economics, Wuhan, Hubei* 430205, *China*)

Abstract: To overcome the shortcomings the traditional particle swarm optimization algorithm (PSO), such as poor ability to escape a local optimal, premature convergence and low precision, we proposed a new PSO based on multiscale-selective-learning and detecting-shrinking strategies, which called MDPSO in short. In the multiscale-selective-learning strategy, a particle executes a multiscale learning process to improve its studying efficiency by adopting its topology, selecting neighbors, and choosing target variable dimensions. In the detecting-shrinking strategy, particles′ historical best solutions are periodic sampling and some useful information, which extracting from the sampling results, is used to direct the best solutions to carry out a detecting operation. The aims of the strategy are to improve PSO′s global searching ability and to help the population escape a local optimal solution. While the best solution situating around a global optimal solution, the algorithm implements the shrinking strategy to confine the search space to a small one the aims of which are to improve the PSO′s exploitation ability and to increase the accuracy of the solutions. The proposed method was applied to twenty-two typical benchmark functions, and the comparisons of the performance between MDPSO and other eight PSO algorithms were experimented. The results suggest that the proposed strategies can effectively overcome the premature convergence, speed up the convergence and improve solutions accuracy.

Key words: particle swarm optimization; premature convergence; multiscale learning; detecting strategy

收稿日期:2014-09-05;修回日期:2014-11-12;责任编辑:马兰英
基金项目:国家自然科学基金(No.41231174,No.61165004,No.61562028);华东交通大学校立科研项目(No.14JG03);江西省教育厅科研项目(No.GJJ150539);江西省自然科学基金(N0.2015BAB207022);新疆维吾尔自治区高校科研计划青年教师科研启动基金(No.2014JYT041606)

1 引言

粒子群算法(Particle Swarm Optimization,PSO)是 Kennedy[1]等人于 1995 年提出的一种群智能优化算法,由于其理论简单、易于实现,因此,PSO 算法在提出后被迅速应用于许多领域. 影响 PSO 算法性能的主要因素是粒子位置迭代公式中的参数以及粒子邻域的拓扑结构,因此众多学者在这两方面进行了广泛的研究和改进. 第一类改进是通过调整 PSO 的参数来提升算法性能. 如 Shi 和 Eberhart[2]对 PSO 算法的速度项引入了惯性权重来平衡全局搜索性能和收敛速度,Asanga Ratnaweera[3]则引入时变的加速因子(Time-Varying Acceleration Coefficients)调节粒子的自我学习和社会学习的强度. Ioan[4]和 Jiang[5]先后对标准粒子群算法进行了收敛性分析,并提出了收敛性和稳定性较好的一组参数选择. Zhan[6]则对惯性因子、加速因子及其它多个参数进行自适应调整以改善算法综合性能;第二类改进是通过采用不同类型的邻域拓扑结构来提高种群的多样性、改善算法性能. 如 Suganthan[7]和 Peram[8]分别利用粒子间欧式距离和粒子适应值来确定粒子的学习模式,通过这种动态选择学习对象的策略,改善种群的多样性. Rui[9]和 Liang[10]先后提出了完全感知 PSO 算法和综合学习 PSO 算法,这两种算法的共同特点是通过丰富粒子的社会学习模式,改善种群多样性. 众多研究结果也表明动态、多样的邻域拓扑结构对于多峰函数优化具有很好的效果[11,12]. 需要指出的是,对 PSO 算法的改进并非局限于某一方面的改进,很多时候是对参数、拓扑结构等同时进行改进[13~17].

上述对 PSO 算法的改进的目的主要是在防止早熟收敛的同时尽量提高求解精度,但当种群已经陷入局部最优时,却没有先验知识指导种群跳出局部最优. 因此,本文提出了一种具有多尺度选择性学习和探测-收缩机制的 PSO 算法(Multiscale-selective-learning hybrid with Detecting-shrinking PSO,MDPSO),MDPSO 算法根据不同进化阶段时种群和个体生存环境的不同,在多个尺度上进行了学习模式的选择. 同时,MDPSO 还利用探测机制对粒子个体的历史信息进行周期采样与统计,利用统计结果指导种群进行探测和搜索空间的收缩. 实验结果表明,上述改良机制有效地提升了 PSO 算法的综合性能.

2 PSO 算法

在 PSO 算法中,每个粒子的位置可视为 D 维问题空间中的一个候选解,粒子种群的飞行过程即为算法求解的过程. 假设 D 维搜索空间中,粒子群的规模为 N,则第 i 个($i<N$)粒子在第 t 代时可用两个指标来描述:位置向量 $\boldsymbol{X}_i^t=(x_{i,1}^t,x_{i,2}^t,\cdots,x_{i,D}^t)$ 和速度向量 $\boldsymbol{V}_i^t=(v_{i,1}^t,v_{i,2}^t,\cdots,v_{i,D}^t)$. 若第 i 个粒子搜索至第 t 代时的个体历史最优解为 $\mathrm{pBest}_i=(pbest_{i,1},pbest_{i,2},\cdots,pbest_{i,D})$、种群历史最优解为 $\mathrm{gBest}=(gbest_1,gbest_2,\cdots,gbest_D)$,则在第 $t+1$ 代时,第 i 个粒子第 j 维的速度和位置更新公式如下:

$$v_{i,j}^{t+1}=\omega\cdot v_{i,j}^t+c_1\cdot r_1\cdot(pbest_{i,j}-x_{i,j}^t)+c_2\cdot r_2\cdot(gbest_j-x_{i,j}^t) \tag{1}$$

$$x_{i,j}^{t+1}=x_{i,j}^t+v_{i,j}^{t+1} \tag{2}$$

其中,ω 为惯性权重,表示前一时刻的速度对本次移动的影响,用于平衡算法的收敛速度和全局搜索能力,较大的 ω 有利于全局搜索,而较小的 ω 则可以提高算法的局部开采能力和求解精度[2,6];c_1 和 c_2 为学习因子,表示粒子自我学习和社会学习的强度,即用来调节粒子向 pBest_i 和 gBest 的学习强度;r_1 和 r_2 为[0,1]内均匀分布的随机数,用来增强算法搜索的随机性.

3 MDPSO 算法

PSO 算法在复杂多峰函数优化中较易出现早熟收敛,因此很多学者通过不同的策略来增强种群多样性,从而避免算法陷入局部最优,但这也使得算法的收敛速度变慢. 同时,当种群已陷入局部最优后,缺少具有指导意义的逃逸策略来帮助种群跳出局部最优并找到更优位置. 为此,本文提出了一种具有多尺度选择性学习和探测-收缩机制的 PSO 算法(Multiscale-selective-learning hybrid with Detecting-shrinking PSO,MDPSO). 在 MDPSO 中,种群中的粒子个体将种群进化过程和自身进化状态相结合,在拓扑结构、领域个体以及不同的变量维等多个尺度上进行选择性学习. 此外,为了增强算法跳出局部最优的能力,还赋予了种群历史最优解 gBest 探测学习的能力. 同时,根据 gBest 以及 pBest_i 的统计信息,对种群的搜索空间进行收缩,以提高算法的求解精度.

3.1 多尺度选择性学习机制

3.1.1 邻域拓扑结构的选择

为了改善 PSO 算法综合性能,必须保证算法在进化初期具有更好的全局搜索能力,尽可能地"勘探"到全局最优解所在区域,避免"早熟"收敛;而在进化后期,则希望种群能以较快的速度收敛到最优解附近,提高求解精度. 因此,本文提出了一种随进化过程依概率选择不同邻域拓扑结构的策略. 该策略可描述如下:

$$TOP_{neighbor}=\begin{cases}top_{star}, & \text{if } rand\leqslant\sqrt{t/T}\\ top_{ring}, & \text{else}\end{cases} \tag{3}$$

其中,$TOP_{neighbor}$ 表示种群的邻域拓扑结构;top_{star} 和 top_{ring} 分别为星型和环形邻域拓扑结构;$rand$ 是[0,1]间均匀分布的随机数. 从式(3)可看出:在进化初期,种群选择

top_{ring}的概率较大,有利于算法保持种群多样性,避免"早熟"收敛;而在进化后期,种群选择 top_{star} 的概率变大,种群的收敛速度得到提升,从而提高了算法的收敛速度和求解精度.

3.1.2 邻域粒子与目标变量维的选择

本文对粒子个体设置了一个最大连续"停滞"代数 $MaxStag_{ind}$,当粒子 i 在进化过程中连续"停滞"次数 $stay_i$ 达到该阈值时,将重新选择学习对象及学习方式,具体策略如下:

(1)当 pBest_i 是其邻域内粒子的历史最优解中最优,则粒子 i 在进行社会学习时,从现有邻域粒子中获取有益信息的概率较低,此时应考虑重新选择合适的粒子作为其邻域粒子,选择策略可描述如下:

$$Neighbor_i^{t+1}=\begin{cases}\boldsymbol{X}_{best1}^t\cup\boldsymbol{X}_{best2}^t, & \text{if } rand\leqslant\sqrt{t/T}\\ \boldsymbol{X}_{far1}^t\cup\boldsymbol{X}_{far2}^t, & \text{else}\end{cases}\tag{4}$$

其中,$Neighbor_i^t$ 为粒子 i 在第 t 代的邻域粒子;$\boldsymbol{X}_{best1}^t$ 和 $\boldsymbol{X}_{best2}^t$ 分别为种群中除粒子 i 之外最优的 2 个粒子;$\boldsymbol{X}_{far1}^t$ 和 $\boldsymbol{X}_{far2}^t$ 分别为种群中距离粒子 i 最远的 2 个粒子. 在为某粒子重新选择邻域粒子后,整个种群的邻域拓扑结构不再是严格意义上的环形结构,为了描述方便,本文将其统称为环形邻域拓扑结构.

(2)当 $pBes\ t_i$ 不是其邻域粒子的历史最优解中最优,则表明其邻域粒子所蕴含的有益信息未被粒子 i 有效学习,因此,粒子 i 应当有选择地向其邻域粒子进行学习,即只学习那些能改善自身适应值的变量维,放弃对那些使自身劣化的变量维的学习. 选择变量维学习的策略可描述如下式:

$$Dim_{i,j}=\begin{cases}1, & \text{if } fit((x_{i,1}^t,x_{i,2}^t,\cdots,nb_{i,j}^t,\cdots,x_{i,D}^t))>fit(\boldsymbol{X}_i^t)\\ 0, & \text{else}\end{cases}\tag{5}$$

其中,$nb_{i,j}^t$ 为 $Neighbor_i^t$ 在第 j 维($1\leqslant j\leqslant D$)的取值;$(x_{i,1}^t,x_{i,2}^t,\cdots,nb_{i,j}^t,\cdots,x_{i,D}^t)$ 为用 $nb_{i,j}^t$ 替换 $x_{i,j}^t$ 后得到的新粒子;$Dim_{i,j}$ 为粒子 i 在第 j 维上的学习标志:1 表示学习,0 表示不学习. 当种群初始化和个体重新选择了邻域个体后,个体的变量维学习标志都置 1,因为此时无法确定该个体在哪些变量维上进行学习更有效.

3.2 探测-收缩机制

3.2.1 探测机制

在复杂多维函数的优化过程中,gBest 的每维变量一般很难同时处于全局最优解附近,这时就需要 gBest 有目的地探测该维空间的其它区域,以便能跳出当前的局部最优.

为方便探测行为的操作,每维变量空间被等分为多个互不相交的子区间,即:

$$\bigcup_{j=1}^{M}s_{i,j}=\boldsymbol{S}^i,s_{i,j}\bigcap_{j,k=1}^{M}s_{i,k}=\varnothing(j\neq k)\tag{6}$$

其中,$\boldsymbol{S}^i$ 为第 i 维($1\leqslant i\leqslant D$)变量的搜索空间;$s_{i,j}$ 为第 i 维的第 j 个($1\leqslant j\leqslant M$)搜索子区间;M 为第 i 维变量空间划分为的子区间个数. 对搜索空间进行划分后,就可以确定 gBest 每维变量所在的子区间,同时,gBest 将以子区间为单位进行探测. 为了使 gBest 的探测更有目的性,本文将利用粒子群的 pBest_i 和 gBest 的统计信息来确定各子区间的优势度,结合当前 gBest 所处的子区间来选择合适的目标子区间进行探测. 具体为:通过每隔 Cycle 代对种群中 pBest_i 进行周期采样和统计,获取每维变量在各个子区间上的优势度,gBest 将利用子区间的优势度来指导其探测过程.

定义 1 子区间优势度. 若 $pbest_{i,j}$ 为第 i 个粒子的历史最优解在第 j 维变量上的取值,$s_{j,k}$ 为第 j 维变量第 k 个子区间,则当前种群第 j 维变量第 k 个子区间的优势度 $F_{j,k}$($1\leqslant j\leqslant D,1\leqslant k<M$)为:

$$F_{j,k}=\begin{cases}F_{j,k}+1, & \text{if } pbest_{i,j}\in s_{j,k}\\ F_{j,k}, & \text{else}\end{cases}\tag{7}$$

根据 $F_{j,k}$ 值的大小,可将每维变量的子区间分为三类:优势子区间(superior subregion)、劣势子区间(inferior subregion)和平凡子区间(ordinary subregion).

定义 2 优势子区间. 第 j 维变量优势子区间为:

$Sup_j=\{s_{j,k}\mid\max\{F_{j,k}\}\},(1\leqslant j\leqslant D,1\leqslant k<M)$

定义 3 劣势子区间. 第 j 维变量劣势子区间为:

$Inf_j=\{s_{j,k}\mid\min\{F_{j,k}\}\},(1\leqslant j\leqslant D,1\leqslant k<M)$

定义 4 平凡子区间. 第 j 维变量平凡子区间为:

$Ord_j=\boldsymbol{S}^j-(Sup_j+Inf_j),(1\leqslant j\leqslant D,1\leqslant k<M)$

gBest 根据第 j 维变量取值所属的子区间类别进行相应的探测. 相应的探测过程如下:

(1)若 $gbest_j\in Sup_j$,说明此时 $gbest_j$ 对种群第 j 维变量的影响很大,大部分粒子的 $pbest_{i,j}$ 取值都在 $gbest_j$ 附近,而在 Inf_j 内的取值则很少(甚至没有),若在 Inf_j 中存在第 j 维变量的全局最优解,则粒子群在后续飞行的过程中发现该维全局最优解的概率非常低或者需要更多次迭代. 因此,gBest 应该首先选择对 Inf_j 进行探测,以跳出当前的局部最优.

(2)若 $gbest_j\in Inf_j$,说明此时 $gbest_j$ 与当前种群中大部分粒子的 $pbest_{i,j}$ 不在同一子区间. 出现这种现象有三种可能性:第一种是大部分 $pbest_{i,j}$ 陷入了局部最优,而当前的 $gbest_j$ 处于第 j 维的全局最优解附近,这种情况下,不必进行探测;第二种是大部分的 $pbest_{i,j}$ 均集中在全局最优解附近,而 $gbest_j$ 则远离该区域,这时 gBest 可以选择在 Sup_j 内进行探测;第三种情况是当前粒子种群在该维的取值都不在全局最优区间,此时也不必进行探测. 综合考虑以上几种情况,为了提高算法的收敛速度,此时选择向 Sup_j 进行探测.

(3)若 $gbest_j\in Ord_j$,gBest 选择在其它任一子区间

内进行探测. 此时的探测过程和一般的变异操作类似,目的是为了增强种群的多样性.

$gbest_j$在进行上述 3 种探测操作时均采用的贪心策略,即在相应子区间随机选择值 x,若将 x 替换 $gbest_j$后能改善 gBest 的适应值,则保留 x,并替换原 gBest,否则将抛弃 x. 为防止 gBest 对同一子区间进行连续多次探测,本文采用了禁忌探测机制,即每次探测完某个子区间后,为该子区间置"已探测"标志,下次再进行探测时则不再将该子区间纳为探测对象,而是选择剩余子区间中符合要求的子区间. 当所有子区间都已被置为"已探测",则清除所有子区间的"已探测"标志. 这样,就保证了 gBest 在对重点子区间进行探测的同时,也满足了 gBest 对整个搜索空间的遍历性,增强了其搜索能力. 综上所述,gBest 在第 j 维的探测过程可描述为算法 1.

算法 1　$Detecting(\text{gBest}, F_{j,k}, \text{taboo\_table}(j,k), s_{j,k})$

/ * gBest:当前种群历史最优解;$F_{j,k}$:第 j 维变量的第 k 个子区间的优势度;taboo_table(j,k):第 j 维变量的第 k 个子区间是否探测标志,0 为未探测,1 为已探测;$s_{j,k}$为第 j 维变量的第 k 个搜索子区间的下界 * /

Begin

1:for $j = 1$ to D do

2:　对 $F_{j,k}(1 \leqslant k \leqslant M)$进行升序排列;假定排序结果为 $F_{i,p_m} \leqslant F_{i,p_n}, 1 \leqslant m < n \leqslant M$;

3:　if $gbest_j \in Sup_j$ then

4:　　for $k = 1$ to M do

5:　　if taboo_table$(j,p_k) = 0$ then goto 7;end if

6:　　end for

7:　　随机生成 $rand \in s_{j,k}$;

8:　　if $fit((gbest_1, \cdots, gbest_{j-1}, rand, gbest_{j+1}, \cdots, gbest_D)) > fit(\text{gbest})$ then//$fit(*)$表示 * 的适应值

9:　　$gbest_j = rand$;

10:　　end if

11:　else if $gbest_j \in Inf_j$ then

12:　　for $k = M$ to 1 do

13:　　if taboo_table$(j,p_k) = 0$ then goto 15;end if

14:　　end for

15:　　随机生成 $rand \in s_{j,k}$;

16:　　if $fit((gbest_1, \cdots, gbest_{j-1}, rand, gbest_{j+1}, \cdots, gbest_D)) > fit(\text{gbest})$ then $gbest_j = rand$;end if

17:　else // $gbest_j \in Ord_j$

18:　　随机选择 k,保证 $gbest_j \notin s_{j,k}$ && $taboo\_table(j,k) \neq 0$;

19:　　随机生成 $rand \in s_{j,k}$

20:　　if $fit((gbest_1, \cdots, rand, \cdots, gbest_D)) > fit(\text{gbest})$ then $gbest_j = rand$;end if

21:　end if

22:　taboo_table$(j,p_k) = 1$;

End Begin

3.2.2　收缩机制

当算法在第 j 维进行多次探测后,确定全局最优解存在于某个或某些子区间中,就有必要将第 j 维的搜索限定在一定的有效区域内(本文称此过程为空间收缩),以加快算法收敛速度,提高求解精度. 本文将以 gBest 在第 j 维上进行多次探测后,其适应值连续停滞代数作为执行空间收缩的指标. 这里,首先有如下定义:

$$gbstay_j^{t+1} = \begin{cases} 0, & \text{if gBest 对第 } j \text{ 维执行探测后适应值更优} \\ gbstay_j^t + 1, & \text{else} \end{cases} \tag{8}$$

式中,$gbstay_j^t$ 为 gBest 在第 j 维上进行 t 次探测后其适应值连续停滞的代数. 当 $gbstay_j^t$ 超过设定阈值 $MaxStag_{pop}$ 时,则对第 j 维执行空间收缩过程. 对第 i 维变量空间的收缩算法可描述为算法 2.

算法 2　$S^i = Shrinking(F_{i,k}, gbest_i, \boldsymbol{S}^i)$

Begin

1:　for $j = 1$ to M do

2:　　if $gbest_i \in s_{i,j}$ then $\boldsymbol{S}^{i*} = s_{i,j}$ break;end if

3:　end for

4:　$k = j - 1$;

5:　while $k \geq 1$ && $F_{i,k} > 0$

6:　　$\boldsymbol{S}^{i*} = \boldsymbol{S}^{i*} \cup s_{i,k}$;$k + +$;

7:　end

8:　$k = j + 1$;

9:　while $k \leqslant M$ && $F_{i,k} > 0$

10:　　$\boldsymbol{S}^{i*} = \boldsymbol{S}^{i*} \cup s_{i,k}$;$k - -$;

11:　end

12:　$\boldsymbol{S}^i = \boldsymbol{S}^{i*}$

13:　按照式(6)对 $\boldsymbol{S}^i$ 划分为 M 个子区域;

End

3.3　MDPSO 算法

综合 3.1 ~ 3.2 中提出的策略,MDPSO 算法可描述为算法 3.

算法 3　$\text{MDPSO}(N, D, \boldsymbol{S}^i, Max\_FEs, Cycle, MaxStag_{ind}, MaxStag_{pop}, M)$

Begin

1:随机初始化规模为 N、维数为 D 的粒子种群并评价每个粒子;

2:按式(6)将每维搜索空间等分为 M 个子区间;初始化 $Dim_{j,i} = 1$,$(1 \leqslant j \leqslant N, 1 \leqslant i \leqslant D)$

3:评价次数 $fes = N$;进化代数 $t = 0$;

4:　while $fes < Max\_FEs$ do

5:　　$t = t + 1$;

6:　　根据 式(3)选择相应的拓扑结构;

7:　　更新种群 P 中每个粒子的速度和位置;

8:　　评价粒子并更新 pBest 和 gBest;

9:　　更新粒子个体的连续停滞代数 $stay_j (1 \leqslant j \leqslant N)$

10:　　$fes = fes + N$;

11:　　if $stay_j \geqslant MaxStag_{ind} (1 \leqslant j \leqslant N)$ then

12:　　根据式(4)、(5)为粒子 j 选择 $Neighbor_j^t$ 和 $Dim_{j,i} = 1 (1 \leqslant i$

```
        ≤D);
13:     end
14:     if t mod Cycle = =0 then
15:       根据(7)统计子区间优势度 F_{i,k}(1≤i≤D,1≤k≤M);
16:       for i =1 to D do
17:       DetectingOpt(gBest,F_{i,k},taboo_table(i,k),s_{i,k});//1≤k≤M
18:         根据式(8)更新 gbstay_j^t;
19:         if gbstay_j^t ≥ MaxStag_pop then ShrinkingOpt(F_{i,k},gbest_i,S^i);
            end if
20:       end for
21:     end if
22:   end while
End
```

4 仿真实验及结果分析

MDPSO 算法的参数设置:种群规模 $N=20$,粒子最大连续停滞代数 $MaxStag_{ind}=5$,每维变量分割为的子区间数 $M=8$;统计周期 $Cycle$ 为 3;种群停滞阈值 $MaxStag_{pop}$将随着进化代数线性递增,其取值范围为[20,100]. 实验环境为:Intel i3 CPU 2.93GHz. RAM 4.00GB, Windows 7 操作系统,MATLAB 2009a.

4.1 测试函数

本文选取了 22 个 Benchmark 函数进行了对比测试,各函数简要说明如下:

- f_1:Sphere,$\boldsymbol{x}\in[-100,100]^D$,Accept_error:0.001
- f_2:Schwefel's $P2.22$,$\boldsymbol{x}\in[-10,10]^D$,Accept_error:0.001
- f_3:Rosenbrock,$\boldsymbol{x}\in[-30,30]^D$,Accept_error:100
- f_4:Step,$\boldsymbol{x}\in[-100,100]$,Accept_error:0
- f_5:Sum of different power,$\boldsymbol{x}\in[-1,1]^D$,Accept_error:0.001
- f_6:Ackley,$\boldsymbol{x}\in[-32,32]^D$,Accept_error:0.001
- f_7:Alpine,$\boldsymbol{x}\in[-10,10]$,Accept_error:0.001
- f_8:Schwefel's P1.2,$\boldsymbol{x}\in[-500,500]^D$,Accept_error:1000
- f_9:Rastrigin,$\boldsymbol{x}\in[-5.12,5.12]^D$,Accept_error:100
- f_{10}:Rastrigin_noncont,$\boldsymbol{x}\in[-5.12,5.12]^D$,Accept_error:100
- f_{11}:Griewank,$\boldsymbol{x}\in[-600,600]^D$,Accept_error:0.01
- f_{12}:Weierstrass,$\boldsymbol{x}\in[-0.5,0.5]^D$,Accept_error:0.01
- f_{13}:Generalized Penalized,$\boldsymbol{x}\in[-50,50]^D$,Accept_error:0.001
- f_{14}:Shifted Sphere,$\boldsymbol{x}\in[-100,100]^D$,Accept_error:0.001
- f_{15}:Shifted Schwefel $P1.2$,$\boldsymbol{x}\in[-100,100]^D$,Accept_error:1000
- f_{16}:Shifted Schwefel P1.2 with Noise,$\boldsymbol{x}\in[-100,100]^D$,Accept_error:1000
- f_{17}:Shifted Rosenbrock,$\boldsymbol{x}\in[-100,100]^D$,Accept_error:1000
- f_{18}:Shifted Rotated Griewank,$\boldsymbol{x}\in[-600,600]^D$,Accept_error:1000
- f_{19}:Shifted Rotated Ackley,$\boldsymbol{x}\in[-32,32]^D$,Accept_error:1000
- f_{20}:Shifted Rastrigin,$\boldsymbol{x}\in[-5,5]^D$,Accept_error:1000
- f_{21}:Shifted Rotated Rastrigin,$\boldsymbol{x}\in[-5,5]^D$,Accept_error:1000
- f_{22}:Expanded Extended Griewank's plus Rosenbrock,$\boldsymbol{x}\in[-3,1]^D$,Accept_error:1000

其中,$f_1\sim f_5$为单峰函数,主要用来检验算法的求解精度;$f_6\sim f_{13}$为多峰函数,主要用来检验算法的全局搜索能力.$f_{14}\sim f_{22}$为 shifted、rotated 以及复合函数. 本文实验中,变量维数 $D=30$,最大评价次数 $maxFEs=1000*D$. 每个函数独立测试 30 次,取其统计结果(均值 *Mean*、标准方差 *Std. Dev*)进行比较,加粗数据表示对比算法在相应函数上得到的最优结果.

4.2 不同策略的效果

为了检验本文提出的不同策略对算法不同性能方面的贡献度,我们选取了部分函数进行了测试(f_1,f_4为单峰函数,f_6,f_8,f_9,f_{12}为多峰函数),实验结果见表 1. PSO-1 和 PSO-2 分别表示从 MDPSO 算法中移除了多尺度选择性学习机制和探测-收缩机制后得到的算法.

表 1 不同策略的实验结果对比

		MDPSO	PSO-1	PSO-2
f_1	Mean	**1.07E-20**	3.93e-20	4.24e+00
	Std. Dev	**5.67E-20**	2.09e-19	3.78e+00
f_4	Mean	**0.00e+00**	**0.00e+00**	2.37e+01
	Std. Dev	**0.00e+00**	**0.00e+00**	1.33e+01
f_6	Mean	**2.55e-12**	4.40e-11	2.79e+00
	Std. Dev	**3.50e-12**	2.07e-10	7.67e-01
f_8	Mean	3.52e+00	**1.29e+00**	4.84e+03
	Std. Dev	1.10e+01	**5.08e+00**	7.50e+02
f_9	Mean	**2.36e-14**	1.93e-07	4.43e+01
	Std. Dev	**9.16e-14**	9.48e-07	9.83e+00
f_{12}	Mean	**4.95e-04**	8.67e-04	4.18e+00
	Std. Dev	**7.39e-04**	1.50e-03	9.77e-01

从表 1 中可以看出,本文提出的策略有效改善了算法的性能,MDPSO 在测试的 6 个函数中除 f_8 外,都取得到最优的结果. 从表中也看出:移除了探测-收缩机制后,算法的性能急剧下降,这说明合理利用种群的历史信息来指导粒子有目的地进行探测是有效、可行的;移

除了多尺度选择性学习机制后,算法对大部分测试函数的求解精度都有所下降,从对 f_9 优化的结果可以看出,该策略提高了算法的学习效率,大大提升了求解精度. 实验结果表明:本文提出的策略都对原 PSO 算法在不同性能指标上进行了优化与提升.

4.3 求解精度

为了进一步说明本文算法的效果,这里选取了近年来的较为优秀的一些 PSO 算法进行实验对比,相关算法及其参数设置见表 2. 其中,DMS-PSO 算法的种群规模设置为 10 个子种群,每个子种群规模为 3,其它算法的种群规模均为 20. 其中,Merits 表示该算法在测试函数上表现最优的次数. 加粗数值表示对比算法在相应函数上得到的最优结果. 具体结果如表 3 所示.

表 2　相关 PSO 算法的参数设置

Algorithm	Year	Parameters Settings
FDR[8]	2003	ω:[0.4,0.9],$c_1=c_2=1$,$c_3=2$
FIPS[9]	2004	$\chi=0.729$,$\sum c_i=4.1$
CLPSO[10]	2006	ω:[0.4,0.9],$c=1.4945$
DMS-PSO[17]	2008	$\chi=0.729$,$c_1=c_2=1.49445$,$R=10$,$L=100$,$L\_FEs=200$
GDPSO[11]	2009	$W1=0.729$,$W2=2.187$
DCPSO[12]	2012	ω:[0.3,0.95],$c_1=c_2=2$
PSORDS[13]	2013	$\chi=0.7298$,$c_1=c_2=2.05$
PSODDS[13]	2013	$\chi=0.7298$,$c_1=c_2=2.05$

表 3　9 种算法在 22 个测试函数上的实验结果对比

		FDR	FIPS	CLPSO	DMS-PSO	GDPSO	DCPSO	PSORDS	PSODDS	MDPSO
f_1	Mean	1.61e−17	2.94e-01	1.35e-03	1.87e-19	4.41e-12	1.55e-02	2.40e-08	4.19e-16	**1.07E-20**
	Std. Dev	3.64e-17	1.10e-01	8.89e-04	1.64e-19	8.85e-12	1.51e-02	2.81e-08	3.52e-16	**5.67E-20**
f_2	Mean	3.33e-01	1.35e-01	2.57e-03	2.74e-05	7.99e-08	9.85e-01	1.33e+00	2.33e+00	**1.44e-08**
	Std. Dev	1.83e+00	3.30e-02	6.90e-04	1.23e-05	1.42e-07	2.94e+00	3.46e+00	6.26e+00	**7.90e-08**
f_3	Mean	4.39e+01	1.30e+02	1.26e+02	**3.63e+01**	5.82e+01	8.80e+01	3.19e+03	3.14e+03	1.29e+02
	Std. Dev	4.70e+01	7.24e+01	5.00e+01	**2.40e+01**	6.22e+01	6.20e+01	1.64e+04	1.64e+04	1.87e+02
f_4	Mean	2.33e+00	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	7.00e-01	1.62e+01	9.67e-01	**0.00e+00**
	Std. Dev	2.47e+00	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	9.15e-01	2.58e+01	8.09e-01	**0.00e+00**
f_5	Mean	**2.68e-48**	4.54e-17	1.23e-21	4.96e-29	1.72e-38	1.85e-16	1.45e-32	4.24e-10	**1.57e-39**
	Std. Dev	**1.47e-47**	1.85e-16	2.39e-21	6.51e-29	3.43e-38	2.93e-16	6.73e-32	4.52e-10	**8.58e-39**
f_6	Mean	3.48e-01	1.73e-01	9.06e-03	7.46e-07	5.20e-07	6.45e-01	1.02e+00	2.40e-01	**2.55e-12**
	Std. Dev	5.51e-01	5.16e-02	2.94e-03	7.11e-07	7.61e-07	7.17e-01	8.80e-01	4.98e-01	**3.50e-12**
f_7	Mean	3.33e-05	8.28e-02	1.57e-02	1.78e-02	4.03e-02	1.53e-01	7.62e-04	3.67e-01	**8.64e-08**
	Std. Dev	1.80e-04	4.28e-02	6.05e-03	6.91e-02	7.02e-02	3.27e-01	2.26e-03	1.41e+00	**4.08e-07**
f_8	Mean	4.29e+03	1.62e+03	1.42e+02	4.60e+03	5.53e+03	4.51e+03	4.30e+03	4.05e+03	**3.52e+00**
	Std. Dev	6.40e+02	7.70e+02	1.26e+02	7.81e+02	5.70e+02	7.73e+02	5.36e+02	6.75e+02	**1.10e+01**
f_9	Mean	5.06e+01	1.32e+02	1.79e+01	2.51e+01	2.63e+01	5.34e+01	6.49e+01	5.47e+01	**2.36e-14**
	Std. Dev	1.84e+01	1.28e+01	3.84e+00	7.71e+00	4.75e+00	2.15e+01	1.54e+01	1.28e+01	**9.16e-14**
f_{10}	Mean	4.18e+01	1.19e+02	1.91e+01	3.00e+01	3.55e+01	7.75e+01	5.96e+01	5.92e+01	**3.88e-11**
	Std. Dev	1.55e+01	1.65e+01	2.58e+00	4.78e+01	1.07e+01	3.19e+01	2.60e+01	2.27e+01	**1.36e-10**
f_{11}	Mean	2.09e-02	5.84e-01	1.56e-02	**7.48e-16**	2.34e-03	6.10e-02	2.00e-02	1.22e-02	3.08e-02
	Std. Dev	2.56e-02	1.49e-01	8.96e-03	**6.84e-16**	7.94e-03	4.83e-02	2.45e-02	1.21e-02	2.92e-02
f_{12}	Mean	1.03e+00	9.23e-01	4.84e-02	4.90e-02	**1.24e-05**	4.24e-01	3.97e+00	3.13e+00	4.95e-04
	Std. Dev	1.03e+00	1.11e-01	1.09e-02	1.02e-02	**2.30e-05**	1.48e-01	2.26e+00	1.95e+00	7.39e-04
f_{13}	Mean	5.18e-02	2.26e-02	5.40e-05	1.23e-09	2.80e-01	1.28e-01	1.73e-01	4.49e-02	**1.19e-20**
	Std. Dev	6.53e-02	2.06e-02	3.85e-05	6.05e-09	3.15e-01	2.47e-01	3.28e-01	1.11e-01	**6.43e-20**

续表

		FDR	FIPS	CLPSO	DMS-PSO	GDPSO	DCPSO	PSORDS	PSODDS	MDPSO
f_{14}	Mean	4.65e-08	4.35e-01	5.65e-04	**5.68e-14**	3.05e+01	2.01e+02	2.22e+03	6.16e+02	2.05e-13
	Std. Dev	2.23e-13	2.09e-01	2.24e-04	**6.42e-30**	1.06e+02	1.10e+03	2.72e+03	5.41e+02	6.26e-14
f_{15}	Mean	2.38e+03	3.82e+03	9.33e+03	9.42e+02	1.33e+04	6.68e+03	**3.88e+02**	7.98e+02	2.24e+03
	Std. Dev	2.20e+03	9.31e+02	1.71e+03	5.59e+02	3.87e+03	2.63e+03	**5.26e+02**	1.77e+03	1.20e+03
f_{16}	Mean	1.05e+04	9.99e+03	2.14e+04	1.23e+04	3.05e+04	1.54e+04	**6.89e+03**	2.36e+04	1.52e+04
	Std. Dev	7.11e+03	2.89e+03	4.08e+03	2.74e+03	6.78e+03	5.02e+03	**4.45e+03**	6.92e+03	4.38e+03
f_{17}	Mean	1.79e+08	1.51e+03	5.59e+02	**2.46e+02**	1.07e+04	3.26e+03	1.70e+08	1.29e+07	1.12e+03
	Std. Dev	2.89e+08	1.20e+03	3.30e+02	**1.45e+02**	2.24e+04	4.33e+03	2.72e+08	2.97e+07	2.69e+03
f_{18}	Mean	1.54e+02	1.24e+01	2.98e+00	**6.47e-03**	6.96e+00	3.31e+01	6.83e+01	2.26e+01	9.98e-02
	Std. Dev	1.33e+02	3.51e+00	1.37e+00	**5.86e-03**	7.60e+00	8.50e+01	5.56e+01	2.33e+01	2.27e-01
f_{19}	Mean	2.10e+01	2.11e+01	2.10e+01	**2.05e+01**	2.10e+01	2.10e+01	2.10e+01	2.07e+01	2.09e+01
	Std. Dev	8.19e-02	5.59e-02	5.14e-02	**9.16e-02**	6.61e-02	7.24e-02	8.42e-02	2.13e-01	9.58e-02
f_{20}	Mean	6.46e+01	9.67e+01	7.75e+00	4.84e+01	4.42e+01	6.61e+01	9.69e+01	9.94e+01	**4.24e-13**
	Std. Dev	1.95e+01	1.44e+01	2.04e+00	1.26e+01	1.26e+01	2.08e+01	2.91e+01	2.19e+01	**3.06e-13**
f_{21}	Mean	1.05e+02	1.99e+02	1.81e+02	7.53e+01	**4.87e+01**	1.84e+02	1.10e+02	1.68e+02	1.80e+02
	Std. Dev	2.89e+01	1.08e+01	1.97e+01	1.58e+01	**1.52e+01**	3.25e+01	2.97e+01	4.01e+01	5.65e+01
f_{22}	Mean	3.91e+00	1.57e+01	7.52e+00	4.46e+00	3.89e+00	8.55e+00	4.29e+00	4.33e+00	**1.23e+00**
	Std. Dev	1.09e+00	1.35e+00	9.89e-01	9.06e-01	1.19e+00	2.12e+00	1.23e+00	1.20e+00	**2.87e-01**
Merits		1	1	1	7	3	0	2	0	**11**

从表3可以看出,本文提出的MDPSO在11个函数上都取得了最优表现,尤其是在f_6、f_9、f_{10}、f_{13}和f_{20}等几个多峰函数上取得了非常明显的优势,这说明本文提出的几种策略可以有效地帮助算法跳出局部最优;DMS-PSO则在7个函数上取得了最优表现,这也说明利用小规模子种群间的协作是提升群智能算法的有效途径之一. 需要指出的是,尽管CLPSO只在1个函数上取得最优表现,但其在所有多峰函数上均取得了前3名的表现,在多峰函数优化上表现出良好的性能.

4.4 *t*-检验

为了验证数据的置信度,本文还进一步对上述算法的优化结果进行了*t*-检验,置信水平$\alpha=0.05$,检验结果如表4所示. 表中,MDPSO显著优于对比算法的用黑体标出,显著劣于对比算法的用方框标出,Better、Same和Worse分别表示MDPSO比相应的对比算法在*t*-检验结果中具有显著更优、相同和显著更劣的函数个数. 尽管MDPSO和FIPS、CLPSO、DMS-PSO、GDPSO等在各自的30次独立测试中,均在函数f_4上取得了全局最优解(表中用-表示),但MDPSO能以最少的评价次数和时间耗费取得全局最优解(结果见表5). *t*-检验的结果表明,MDPSO算法的综合性能最优,该结论也验证了表3的实验结果.

4.5 收敛速度

表5给出了对比算法在达到设定精度(Accept_error)时所需的评价次数及耗时,其中"-"表示该算法在相应函数上未能在最大评价次数内达到所需精度. 表中数据均为独立运行30次的平均值,其中平均评价次数Fes和耗时Time均为算法达到设定精度时的取值,Ratio表示在30次独立运行中算法能在最大评价次数内获得精度误差要求的解的比例.

从表5可看出,MDPSO在大部分函数上均能以最少的评价次数和耗时取得设定精度的解. 尽管本文提出的MDPSO算法中引入了多种不同的策略,在算法描述上较为复杂,但这三种策略本身耗时其实很少. 一方面是因为这些策略并非每轮迭代都执行,而是种群进化到一定时期是才执行一次;另一方面,每个策略本身的运算也很简单,耗时较少,因此,加入这些策略后的算法在性能提升的同时,运算量增加很少. 不同算法在测试函数上的收敛曲线也说明了MDPSO具有较快的速度,由于篇幅有限,本文省略了算法收敛曲线对比图.

表 4　均值的 t-检验结果($\alpha=0.05$)

		FDR	FIPS	CLPSO	DMS-PSO	GDPSO	DCPSO	PSORDS	PSODDS
f_1	t	**2.420**	**14.644**	**8.344**	**5.548**	**2.729**	**5.622**	**4.662**	**6.517**
	p	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
f_2	t	**1.000**	**22.425**	**20.345**	**12.168**	**2.210**	**1.838**	**2.115**	**2.041**
	p	**0.043**	**0.000**	**0.000**	**0.000**	**0.037**	**0.000**	**0.000**	**0.000**
f_3	t	-2.432	0.003	-0.099	−2.708	−1.984	−1.153	1.023	1.006
	p	0.019	0.126	0.020	0.005	0.058	0.079	0.047	0.047
f_4	t	**7.397**	–	–	–	–	**4.188**	**3.437**	**6.547**
	p	**0.000**	–	–	–	–	**0.000**	**0.000**	**0.000**
f_5	t	−1.000	**1.343**	**2.814**	**4.175**	**2.421**	**3.466**	**1.179**	**5.144**
	p	0.043	**0.035**	**0.000**	**0.000**	**0.000**	**0.000**	**0.031**	**0.000**
f_6	t	**3.457**	**18.326**	**16.851**	**5.739**	**3.738**	**4.928**	**6.362**	**2.640**
	p	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
f_7	t	**1.009**	**10.582**	**14.157**	**1.410**	**3.149**	**2.570**	**1.846**	**1.433**
	p	**0.043**	**0.000**	**0.000**	**0.037**	**0.000**	**0.000**	**0.000**	**0.004**
f_8	t	**36.746**	**11.462**	**5.997**	**32.216**	**53.099**	**31.890**	**43.912**	**32.806**
	p	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
f_9	t	**15.011**	**56.572**	**25.525**	**17.815**	**30.274**	**13.611**	**23.175**	**23.416**
	p	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
f_{10}	t	**14.789**	**39.762**	**40.590**	**34.405**	**18.277**	**13.310**	**12.572**	**14.267**
	p	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
f_{11}	t	−1.394	**19.945**	−2.724	−5.772	−5.146	**2.939**	−1.547	−3.212
	p	0.653	**0.000**	0.001	0.000	0.000	**0.003**	0.825	0.012
f_{12}	t	**5.501**	**45.786**	**24.025**	**25.982**	**−3.576**	**15.735**	**9.609**	**8.780**
	p	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
f_{13}	t	**4.354**	**5.993**	**7.704**	**1.117**	**5.038**	**2.848**	**2.897**	**2.214**
	p	**0.000**	**0.000**	**0.000**	**0.030**	**0.000**	**0.000**	**0.000**	**0.000**
f_{14}	t	**5.918**	**11.407**	**13.848**	−3.159	**1.572**	**1.000**	**4.459**	**6.226**
	p	**0.000**	**0.000**	**0.000**	0.007	**0.005**	**0.043**	**0.000**	**0.000**
f_{15}	t	−0.519	**5.214**	17.380	−4.163	**14.761**	8.224	−6.117	−3.204
	p	0.037	**0.005**	0.670	0.000	**0.000**	0.050	0.000	0.592
f_{16}	t	−2.985	−5.924	6.916	−3.168	**11.416**	**0.569**	−7.787	**6.358**
	p	0.009	0.774	0.161	0.468	**0.013**	**0.024**	0.524	**0.000**
f_{17}	t	**3.396**	**0.465**	−1.525	−2.213	**2.286**	**2.169**	**3.421**	**2.380**
	p	**0.000**	**0.059**	0.001	0.000	**0.001**	**0.003**	**0.000**	**0.000**
f_{18}	t	**6.353**	**19.272**	**11.715**	−6.209	**4.980**	**2.132**	**6.726**	**5.309**
	p	**0.000**	**0.000**	**0.000**	0.000	**0.000**	**0.000**	**0.000**	**0.000**
f_{19}	t	5.535	**9.109**	**8.400**	−10.968	**8.119**	5.380	5.450	−4.315
	p	0.150	**0.013**	**0.016**	0.440	**0.033**	0.142	0.166	0.002

续表

		FDR	FIPS	CLPSO	DMS-PSO	GDPSO	DCPSO	PSORDS	PSODDS
f_{20}	t	**18.177**	**36.640**	**20.807**	**20.982**	**19.227**	**17.468**	**18.215**	**24.884**
	p	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
f_{21}	t	−7.192	**4.667**	**1.935**	−12.489	−16.213	1.975	−6.516	0.210
	p	0.223	**0.000**	**0.006**	0.001	0.000	0.845	0.247	0.338
f_{22}	t	**12.747**	**57.234**	**33.453**	**18.484**	**11.665**	**18.656**	**13.091**	**13.496**
	p	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
Better		15	19	16	12	17	18	17	18
Same		3	3	3	3	2	4	4	2
Worse		4	0	3	7	3	0	1	2

表 5 设定误差精度下的成功率及收敛速度对比

		FDR	FIPS	CLPSO	DMS-PSO	GDPSO	DCPSO	PSORDS	PSODDS	DRSPSO
f_1	Ratio	**100**	0	46.67	**100**	**100**	0	**100**	**100**	**100**
	Fes	18007	–	29942	3277	20153	–	18193	11447	**7713**
	Time	14.33	–	14.77	5.92	10.10	–	12.40	7.68	**3.61**
f_2	Ratio	96.67	0	0	**100**	**100**	0	80.00	83.33	**100**
	Fes	19520	–	–	22014	20733	–	24960	19587	9456
	time	11.85	–	–	10.58	10.60	–	12.05	6.99	4.47
f_3	*Ratio*	96.67	43.33	0	**100**	80.00	63.33	76.67	90.00	86.67
	Fes	17753	29393	–	**7173**	16253	27569	18313	10500	**9076**
	time	10.69	14.24	–	**3.47**	8.29	12.13	8.81	5.08	**4.17**
f_4	Ratio	6.67	**100**	**100**	**100**	**100**	53.33	3.33	26.67	**100**
	Fes	–	26613	19218	13595	19820	26928	–	22167	**4516**
	time	–	13.83	9.68	6.58	10.11	11.10	–	3.07	**2.01**
f_5	Ratio	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	Fes	346	1873	2192	638	**473**	2128	1486	1153	956
	time	0.21	1.10	1.20	0.32	**0.24**	0.96	0.67	0.51	0.45
f_6	Ratio	70.00	0	0	**100**	**100**	0	36.67	80.00	**100**
	Fes	20033	–	–	16830	20907	–	25240	13973	**9376**
	time	12.07	–	–	8.11	10.83	–	10.96	6.76	**4.33**
f_7	Ratio	**100**	0	0	30.00	10.00	0	86.67	26.67	**100**
	Fes	19060	–	–	–	–	–	23187	–	**9480**
	time	11.70	–	–	–	–	–	11.04	–	**4.21**
f_8	Ratio	0	20.00	**100**	0	0	0	0	0	**100**
	Fes	–	28756	12953	–	–	–	–	–	**5193**
	time	–	13.46	7.37	–	–	–	–	–	**2.30**
f_9	Ratio	96.67	0	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	Fes	8866	–	13488	3114	3293	21513	13580	4060	**390**
	time	5.36	–	7.28	1.52	1.65	9.25	2.49	1.90	**0.17**
f_{10}	Ratio	**100**	16.67	**100**	**100**	**100**	66.67	90.00	90.00	**100**
	Fes	7400	29720	9469	3027	3786	26783	4760	3260	**316**
	time	4.47	15.57	4.98	1.48	1.91	11.09	2.24	1.53	**0.14**

续表

		FDR	FIPS	CLPSO	DMS-PSO	GDPSO	DCPSO	PSORDS	PSODDS	DRSPSO
	Ratio	43.33	0	26.67	**100**	93.33	13.33	46.67	60.00	33.33
f_{11}	Fes	22250	–	29154	**3517**	17880	–	25407	23340	23557
	time	10.66	–	15.24	**1.70**	6.26	–	7.92	4.94	9.35
	Ratio	0	0	0	0	**100**	0	0	6.67	**100**
f_{12}	Fes	–	–	–	–	21967	–	–	–	**11113**
	time	–	–	–	–	22.96	–	–	–	**10.60**
	Ratio	56.67	0	**100**	**100**	13.33	0	56.67	76.67	**100**
f_{13}	Fes	20033	–	24913	21054	27398	–	27760	15693	**9713**
	time	9.75	–	13.33	10.96	8.23	–	13.43	8.01	**4.65**
	Ratio	6.67	0	86.67	**100**	20.00	0	0	3.33	**100**
f_{14}	Fes	29606	–	29165	1984	27064	–	–	18920	**8103**
	time	7.27	–	4.867	0.36	2.87	–	–	55.09	**0.94**
	Ratio	53.33	0	0	60.00	0	0	**90.00**	83.33	23.33
f_{15}	Fes	25360	–	–	27965	–	–	**15947**	9273	25601
	time	5.45	–	–	5.09	–	–	3.29	**2.10**	2.87
	Ratio	0	0	0	0	0	0	0	0	0
f_{16}	Fes	–	–	–	–	–	–	–	–	–
	time	–	–	–	–	–	–	–	–	–
	Ratio	30.00	36.67	80.00	**100**	33.33	56.67	3.33	46.67	86.67
f_{17}	Fes	29426	28386	24418	10155	21368	27604	28742	17204	**4424**
	time	7.31	4.76	4.31	1.714	1.60	3.11	2.22	1.40	**0.54**
	Ratio	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
f_{18}	Fes	**180**	533	2797	785	246	1756	1220	1200	280
	time	0.05	0.10	0.59	0.16	0.05	0.23	0.15	0.16	**0.038**
	Ratio	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
f_{19}	Fes	40	40	**22**	51	40	82	1020	1020	40
	time	0.01	0.006	**0.004**	0.008	0.009	0.01	0.10	0.12	0.006
	Ratio	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	93
f_{20}	Fes	40	40	**21**	49	40	82	1020	1020	40
	time	0.01	0.006	**0.004**	0.007	0.006	0.01	0.10	0.10	0.007
	Ratio	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
f_{21}	Fes	40	73	60	64	**53**	82	1020	1020	76
	time	0.08	0.02	0.02	0.03	**0.02**	0.02	0.10	0.11	0.04
	Ratio	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
f_{22}	Fes	**40**	**40**	229	83	53	82	1020	1020	150
	time	**0.01**	**0.01**	0.08	0.02	0.01	0.02	0.20	0.20	0.03

5　结束语

本文提出了多尺度选择性学习和探测-收缩机制来提高 PSO 的性能. 在多尺度选择性学习机制中,当个体进化出现停滞现象时,该个体根据其自身与邻居个体间适应值的相对优劣关系对拓扑结构、邻居个体、目标变量维等多个尺度上进行变化与调整,实现多尺度上的自适应学习,从而帮助个体更有效地进行学习;探测-收缩策略则首先利用粒子个体的历史最优解的统计信息指导种群历史最优解进行探测,提高算法的全局搜索能力和跳出局部最优的能力,当种群历史最优解长期在某局部区域搜索时,则执行空间收缩策略,将整个种群的搜索过程限定在更小的空间内,以提高算法的求解精度和收敛速度. 通过和其它 8 个优秀的 PSO 算法在 22 个测试函数上

的实验对比表明,本文提出的策略能提高了算法的综合性能,可以有效地帮助算法逃离局部最优,同时也具有较快的收敛速度.由于本文提出的改进策略具有较好的通用性,下一步将利用该思想对其它群智能算法,如蚁群算法、蜂群算法等开展相关的研究.

参考文献

[1] Kennedy J, Eberhart R C. Particle swarm optimization[A]. Proceedings of IEEE International Conference on Neural Networks[C]. Piscataway: IEEE Press, 1995. 1942 – 1948.

[2] Shi Y, Eberhart R C, Fuzzy adaptive particle swarm optimization[A]. Proceedings of IEEE Congress on Evolutionary Computation[C]. Seoul, Korea: IEEE Press, 2001. 1011 – 106.

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

[4] Ioan Cristian Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection[J]. Information Processing Letters, 2003, 85(6): 317 – 325.

[5] M Jiang, et al. Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm[J]. Information Processing Letters, 2007, 102(1): 8 – 16.

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

[7] Suganthan P N, Particle swarm optimizer with neighborhood operator[A]. Proceedings of IEEE Congress on Evolutionary Computation[C]. Washington, D C, USA: IEEE Press, 1999. 1958 – 1962.

[8] T Peram, K Veeramachaneni, C K Mohan. Fitness-distance-ratio based particle swarm optimization[A]. Proceedings of Swarm Intelligence Symp. [C]. Indianapolis, Indiana, USA: IEEE Press, 2003. 174 – 181.

[9] Rui Mendes, James Kennedy, José Neves. The fully informed particle swarm-simpler, maybe better[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 204 – 210.

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

[11] 倪庆剑,等. 一种基于可变多簇结构的动态概率粒子群优化算法[J]. 软件学报, 2009, 20(2): 339 – 349.
NI Qing-Jian, ZHANG Zhi-Zheng, WANG Zhen-Zhen, et al. Dynamic probabilistic particle swarm optimization based on varying multi-cluster structure[J]. Journal of Software, 2009, 20(2): 339 – 349. (in Chinese)

[12] 汤可宗,柳炳祥,杨静宇,等. 双中心粒子群优化算法[J]. 计算机研究与发展, 2012, 49(5): 1086 – 1094.
Tang Kezong, et al. Double center particle swarm optimization algorithm[J]. Journal of Computer Research and Development, 2012, 49(5): 1086-1094. (in Chinese)

[13] Xin Jin, Yongquan Liang, Dongping Tian, et al. Particle swarm optimization using dimension selection methods[J]. Applied Mathematics and Computation, 2013, 219(10): 5185 – 5197.

[14] 喻飞,李元香,魏波 等,透镜成像反学习策略在粒子群算法中的应用[J]. 电子学报, 2014, 42(2): 230 – 235.
YU Fei, LI Yuan-xiang, WEI Bo et al. The application of a novel OBL based on lens imaging principle in PSO[J]. Acta Electronica Sinica, 2014, 42(2): 230 – 235. (in Chinese)

[15] Xuewen Xia, et al. A quantum genetic algorithm based on cellular automata model, International Journal of Modelling[J]. Identification and Control, 2013, 18(3): 243 – 250.

[16] Xuewen Xia, Jingnan Liu, Zhongbo Hu. An improved particle swarm optimizer based on tabu detecting and local learning strategy in a shrunk search space[J]. Applied Soft Computing, 2014, 23(10): 76 – 90.

[17] Zhao S Z, Liang J J, Suganthan P N. Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization[A]. Proceedings of IEEE Congress on Evolutionary Computation[C]. Hong Kong, China: IEEE Press, 2008. 3845 – 3852.

作者简介

夏学文 男,1974 年出生,博士,华东交通大学副教授,研究方向:计算智能及其应用
E_mail: laughkid@163.com; xwxia@whu.edu.cn

桂 凌 女,1977 年出生,本科,华东交通大学交通运输与经济研究所实验师,研究方向:计算机应用.

戴志锋 男,1968 年出生,博士,湖北经济学院副教授,研究方向:智能算法.

谢承旺 男,1974 年出生,博士,华东交通大学副教授,研究方向:多目标优化算法.

魏 波 男,1983 年出生,博士,华东交通大学讲师,研究方向:智能算法.