计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200400115

基于动态近邻套索算子的金字塔演化策略

张 蔷 黄樟灿 谈 庆 李华峰 湛 航
武汉理工大学理学院 武汉 430070
(q.zhang@whut.edu.cn)

摘 要 优化问题是工程领域常见的问题之一,大多数工程问题的本质是函数优化问题。金字塔演化策略(Pyramid Evolution Strategy,PES)在求解函数优化问题时虽然能够很好地建立种群"开采"与"探索"以及"竞争"与"协作"之间的平衡,但是仍存在收敛速度慢、求解精度低、容易陷入局部最优等问题。针对上述问题,提出了基于动态近邻套索算子的金字塔演化策略(DNLPES)。DNLPES算法根据演化代数自适应控制目标个体群的选择范围参数,同时在目标个体群中通过欧氏距离来度量个体之间的差异性;利用个体之间的差异信息引导个体间的协作,通过持续产生新个体并剔除适应度值较差的个体来完成种群进化;通过充分利用种群个体之间的差异性信息并增强个体之间的协作来进一步提高算法的求解精度。将DNLPES算法与7种算法在9个测试函数上进行对比实验,实验结果表明,DNLPES算法在求解精度上具有一定的竞争力,DNLPES算法相比标准PES算法在求解精度与收敛速度上均具有明显优势。

关键词:函数优化算法;金字塔演化策略;动态近邻套索算子;欧氏距离;智能算法
中图法分类号 TP18

Pyramid Evolution Strategy Based on Dynamic Neighbor Lasso

ZHANG Qiang,HUANG Zhang-can,TAN Qing,LI Hua-feng and ZHAN Hang
College of Science,Wuhan University of Technology,Wuhan 430070,China

Abstract The optimization problem is one of the common problems in the engineering field,the essence of most engineering problems is the function optimization problem. Pyramid evolution strategy(PES) algorithm can effectively set a balance between "exploitation" and "exploration" as well as "competition" and "cooperation" when solving function optimization problems,but there are still some shortcomings,such as slow convergence speed,low accuracy,and easy to fall into a local optimal. In order to solve these shortcomings,this paper proposes a pyramid evolution strategy based on dynamic nearest neighbor lasso(DNLPES). The DNLPES algorithm adaptively controls the selection range parameters of the target individual group based on the evolution. At the same time,the Euclidean distance is used to measure the difference between individuals in the target individual group. The difference information between individuals is used to guide the cooperation between individuals,the population evolution is completed by continuously generating new individuals and eliminating the individuals with poor fitness value. The DNLPES algorithm improves the accuracy of the algorithm by making full use of the difference information between individuals in the population and enhancing the cooperation between individuals. Comparing the DNLPES algorithm and the 7 algorithms on 9 test functions,experimental result shows that the DNLPES algorithm has a certain competitiveness in solving accuracy. Compared with the standard PES algorithm,the DNLPES algorithm has obvious advantages in solving accuracy and convergence speed.

Keywords Function optimization problem,Pyramid evolution strategy(PES),Dynamic neighbor lasso(DNL),Euclidean distance,Intelligent algorithm

1 引言

优化问题是工程领域中最常见的问题之一,传统的优化算法包括牛顿法、分支定界法、外逼近法等[1]。随着变量维数的增加,这些算法的计算量会急剧增加[2],存在很大的局限性,因此探索新的求解方法就显得十分重要。受人类智能、生物群体的社会性或自然现象规律的启发,涌现出了不少群智能算法[3]。目前这些算法被应用于图像处理、模式识别、函数优化等领域中[4]。

群智能算法的本质是从动物或个体的集群行为中挖掘规

到稿日期:2020-04-26 返修日期:2020-09-08 本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金资助项目(61672391)
This work was supported by the National Natural Science Foundation of China(61672391).
通信作者:黄樟灿(huangzc@whut.edu.cn)

律，进而提炼出的优化算法，它具备可扩展性、容错性、自适应性、自治性和并行性的特点[5]。比较主流的有从鸟类觅食行为中提炼的粒子群算法（Particle Swarm Optimization，PSO）[6]，由当前最优个体与历史最优个体对当前个体产生的驱动力进行个体更新；从蚂蚁运动轨迹与信息素传递中提炼的蚁群算法（Ant Colony System，ACO）[7]，通过群体产生的信息素来进行分工协作以进行更新；受群体内个体之间的相互合作与竞争的启发而产生的差分进化（Differential Evolution，DE）[8]算法，该算法通过群体中个体之间的差分来指导优化搜索方向；通过模仿自然界生物遗传进化机制提出的遗传算法（Genetic Algorithm，GA）[9]，个体通过杂交将父代优秀基因传递给子代进行个体更新；通过对磷虾群体觅食行为进行仿真的磷虾群算法（Krill Herd，KH）[10]，在该算法中，磷虾个体通过食物局部吸引，随机扰动控制种群进化；磷虾群演化步长限制因子线性下降算法（Krill Herd with Linear Decreasing step，KHLD）[[11]通过增加磷虾移动步长限制来控制种群演化速度；模仿生物地理区域迁入迁出行为进行仿真的生物地理学优化算法（Biogeography-based Optimization，BBO）[[12]，通过对地理区域迁入迁出的动态变化来引导整体区域目标以达到平衡状态，进而达到优化的目的。针对蜂群觅食行为仿生提出的人工蜂群（Artificial Bee Colony，ABC）[13]算法，通过将食物源对雇佣蜂和引领蜂的招募和放弃行为作为驱动力来进行个体更新寻优。此外，近年来国内研究学者针对函数优化问题也提出了比较优秀的群智能演化算法，比较典型的是 Pan 于 2012 年提出的果蝇优化算法（Fruit Fly Optimization Algorithm，FOA）[14]和 Shi 于 2011 年提出的头脑风暴优化算法（Brain Strom Optimization，BSO）[15]，前者是模仿果蝇在觅食过程中嗅觉与视觉对食物的相互作用而提炼的仿生算法，后者是模仿人类利用创造性思维来解决问题而提出的一种算法。这些算法在一定程度上改善了传统的优化算法对初始解的依赖，并很好地被应用于大空间、非线性、全局寻优的复杂问题中，但它们均将研究中心放在了个体之间的竞争上，主要解决种群内部个体之间“开采”[16]与“探索”的矛盾，而忽略了种群内部个体以及种群之间的“竞争”与“协作”之间的矛盾，这会导致算法出现收敛速度慢、求解精度低、容易陷入局部最优等问题。解决以上算法问题的关键在于平衡种群内部的“开采”与“探索”以及“竞争”与“协作”之间的矛盾。

受人类集群智能行为的社会性合作启发，Tan 于 2018 年提出了一种金字塔演化策略（Pyramid Evolution Strategy，PES）[17]。PES 算法从集群分层协同优化的角度，给出了一个群智能协作演化策略，该算法在函数优化问题、整数优化问题、图像颜色量化问题上均有所应用。PES 算法能够有效建立“开采”与“探索”以及“竞争”与“协作”之间的平衡，但是其个体间的协作不够充分，对优秀个体周边的领域挖掘也就不够充分，导致算法的收敛速度慢、求解精度低。为了解决上述问题，本文提出了一种基于动态近邻套索算子的金字塔演化策略（Pyramid Evolution Strategy based on Dynamic Nearest Neighbor Lasso，DNLPES）。

DNLPES 算法根据演化代数自适应地选择目标个体群，通过欧氏距离来度量目标个体群中个体之间的差异信息，以指导挑选近邻个体对。通过近邻个体对之间的协作来挖掘潜在的优秀个体，这充分发挥了目标个体群对潜在优秀个体的引导作用。在演化初期，选择种群中适应度值靠前的部分优秀个体作为近邻个体群，提高算法的局部开采能力，进而加快寻优速度；在演化中后期，目标个体群选取逐渐偏向种群中低适应度值的个体，进而增加算法的全局探索能力，避免陷入局部最优。

2 标准的 PES 算法

2.1 PES 基本原理

金字塔演化算法的模型类似于人类社群智能行为中的企业管理。顶层由少数优秀的决策者构成，中间两层由部分较为优秀的管理者构成，底层由普通的基层员工构成。决策者、管理者和普通员工依次负责决策、筛选和传递命令、实现命令的工作。层间个体相互竞争，优秀个体逐层向上传输。这种自下而上的人才逐层传输规则与明确的分工规则使得金字塔框架能够很好地保存下来。PES 框架结构如图 1 所示。

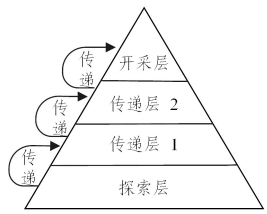


图 1 PES 的框架结构图

Fig. 1 Framework structure of PES

PES 算法中种群的大小从顶层到底层依层增加，搜索邻域半径依层增加，适应度值依层降低。金字塔由下到上的 4 层分别被称为探索层、传递层 1、传递层 2、开采层。标准的 PES 算法按照适应度值对种群进行分层。各层子种群进行明确的“开采”与“探索”工作。标准的 PES 算法具有明确的分工机制和晋升机制，使得算法能有效地平衡“开采”与“探索”、“竞争”与“协作”之间的矛盾。

2.2 标准的 PES 算法

参数设置：设种群规模为 $\boldsymbol{N}$，分层比例为 $\boldsymbol{D}$，传递比例为 $\boldsymbol{T}$，初始搜索半径为 $\boldsymbol{R}_0$，加速步长为 λ，收敛因子为 α，初始可行解集合为 $\boldsymbol{X}_0$，最大迭代次数 G。初始化种群，计算适应度值并分层，即：

$$\boldsymbol{Y}_0 = f(\boldsymbol{X}_0) \tag{1}$$

$$[\widetilde{\boldsymbol{Y}}, \widetilde{\boldsymbol{X}}] = sort(Y_0) \tag{2}$$

$$[\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3, \boldsymbol{X}_4] = D(\widetilde{\boldsymbol{X}}) \tag{3}$$

其中，$f(\cdot)$为适应度函数，Y_0 是适应度值集合，$\widetilde{Y}$ 与 $\widetilde{\boldsymbol{X}}$ 分别表示 $\boldsymbol{X}_0$ 按照适应度值升序排列的适应度值集合与种群集合。$\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3, \boldsymbol{X}_4$ 分别表示将种群 $\widetilde{\boldsymbol{X}}$ 按照分层比例 $\boldsymbol{D}$ 划分后的子种群。

(1)金字塔-层内竞争模型

1)层内探索算子：明确的分工机制使得 4 层具有不同的

搜索半径。新个体的生成规则如下：

$$\boldsymbol{X}_n=\boldsymbol{X}_t+\boldsymbol{R}\times\delta \tag{4}$$

$$\boldsymbol{R}=\boldsymbol{R}_0\times\alpha^{gen} \tag{5}$$

式(4)表示新解探索方式。其中，$\boldsymbol{X}_n$ 与 $\boldsymbol{X}_t$ 分别表示各层新生成的子种群与已存在的子种群，$\boldsymbol{R}_0$ 表示各层个体初始搜索半径，δ 是[−1,1]的随机数，当探索层的 α 取值为 1 时，其他层取 0.99，gen 是当前迭代次数。

2)层内加速算子：对于各层新产生的个体，对其进行加速培养操作，规则如下：

$$\boldsymbol{X}_{ac}=\boldsymbol{X}_n+\lambda(\boldsymbol{X}_n-\boldsymbol{X}_t) \tag{6}$$

其中，$\boldsymbol{X}_{ac}$ 是加速之后产生的新个体集合。

(2)金字塔-层间协作模型

1)层间传递算子：对个体适应度值进行排序，按照单精英轮盘赌策略[17]计算个体被选中的概率，然后根据升层比率 $\boldsymbol{T}$ 进行升层操作。

2)孵化传递算子：本文对传递上来的个体进行加速操作。对应的孵化操作公式如下：

$$\boldsymbol{X}_\beta=\boldsymbol{X}_\alpha+\boldsymbol{R}\times\delta \tag{7}$$

$$\boldsymbol{X}_{au}=\boldsymbol{X}_\beta+\lambda(\boldsymbol{X}_\beta-\boldsymbol{X}_\alpha) \tag{8}$$

式(7)表示对上传个体 $\boldsymbol{X}_\alpha$ 在半径邻域内产生新个体 $\boldsymbol{X}_\beta$ 的过程与式(4)一致，式(8)是针对产生新个体的加速过程。其中，$\boldsymbol{X}_{au}$ 是新个体 $\boldsymbol{X}_\beta$ 加速产生的新解。

3　基于动态近邻套索算子的 PES 算法

标准的 PES 算法中种群个体协作不够充分，因此 DNLPES 算法根据演化代数自适应选择目标个体群，通过欧氏距离来度量个体之间的差异，并根据个体之间的差异信息指导个体之间的协作。动态近邻套索算子主要包含两个子算子：1)动态选择算子；2)近邻套索[18]算子。

(1)动态选择算子：每次演化后从标准 PES 算法得到的种群 $\boldsymbol{S}$ 中动态选择目标个体群：

$$\boldsymbol{P}=f(\boldsymbol{S}) \tag{9}$$

$$[\widetilde{\boldsymbol{P}},\widetilde{\boldsymbol{S}}]=sort(\boldsymbol{P}) \tag{10}$$

其中，$\boldsymbol{P}$ 表示适应度值集合，$\widetilde{\boldsymbol{S}}$ 与 $\widetilde{\boldsymbol{P}}$ 分别表示种群 $\boldsymbol{S}$ 按照适应度值按升序排列的种群、适应度值集合。目标个体群 $\boldsymbol{X}'$ 的选择如下：

$$index1=\begin{cases}0, & gen\leqslant 0.5G\\ (1-c)\dfrac{(gen-0.5G)}{0.5G}, & gen>0.5G\end{cases} \tag{11}$$

$$index2=\begin{cases}c+(1-c)\left(\dfrac{gen}{0.5G}\right), & gen\leqslant 0.5G\\ 1, & gen>0.5G\end{cases} \tag{12}$$

$$ind1=\max(round(N*index1),1) \tag{13}$$

$$ind2=round(N*index2) \tag{14}$$

$$\boldsymbol{X}'=\{\widetilde{\boldsymbol{x}}_{ind1},\widetilde{\boldsymbol{x}}_{ind1+1},\cdots,\widetilde{\boldsymbol{x}}_{ind2}\} \tag{15}$$

其中，c 为选择参数，这里取 1/2，$round(\cdot)$ 为取整函数，目标个体群 $\boldsymbol{X}'$ 表示种群 $\widetilde{\boldsymbol{S}}$ 中下标 $ind1$ 到下标 $ind2$ 构成的种群。动态压力选择算子的运行机制如图 2 所示。

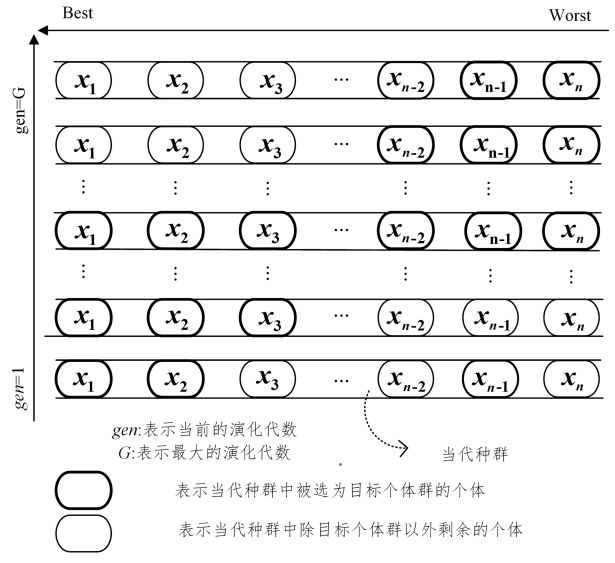


图 2　动态选择算子的运行机制

Fig. 2　Operating mechanism of dynamic selection operator

(2)近邻套索算子：目标个体群 $\boldsymbol{X}'$ 中任意两个个体之间的距离集合 $\boldsymbol{L}$ 定义如下：

$$\boldsymbol{L}=\{l_{uv}=\|\boldsymbol{x}_u-\boldsymbol{x}_v\|_2\,|\,u,v=1,\cdots,m,u\neq v\} \tag{16}$$

其中，m 是目标个体群 $\boldsymbol{X}'$ 的大小，l_{uv} 表示目标个体群 $\boldsymbol{X}'$ 中个体 $\boldsymbol{x}_u$ 与个体 $\boldsymbol{x}_v$ 之间的欧氏距离。

$$\{\boldsymbol{x}_a,\boldsymbol{x}_b\}=\min(\boldsymbol{L}) \tag{17}$$

其中，$\boldsymbol{x}_a$，$\boldsymbol{x}_b$ 表示距离集合 $\boldsymbol{L}$ 中最小距离对应的两个个体。

$$\boldsymbol{y}_k=\begin{cases}\boldsymbol{x}_a+rand(\boldsymbol{x}_a-\boldsymbol{x}_b), & f(\boldsymbol{x}_a)\leqslant f(\boldsymbol{x}_b)\\ \boldsymbol{x}_b+rand(\boldsymbol{x}_b-\boldsymbol{x}_a), & f(\boldsymbol{x}_a)>f(\boldsymbol{x}_b)\end{cases} \tag{18}$$

$$\boldsymbol{Y}=\{\boldsymbol{y}_1,\boldsymbol{y}_2,\cdots,\boldsymbol{y}_{m-1}\} \tag{19}$$

其中，$\boldsymbol{y}_k$ 表示新产生的个体，将 $\boldsymbol{x}_a$ 与 $\boldsymbol{x}_b$ 中适应度值比较好的个体从目标个体群 $\boldsymbol{X}'$ 中删除。

重复上述套索操作，直到 $\boldsymbol{X}'$ 中只有一个个体为止，此时产生新的个体种群 $\boldsymbol{Y}$。动态近邻套索算子的运行机制如图 3 所示。

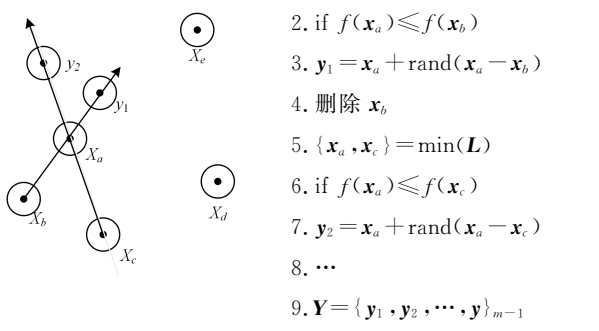


图 3　近邻套索算子运行机制

Fig. 3　Operating mechanism of neighbor lasso operator

DNLPES 算法的伪代码如算法 1 所示。

算法 1　DNLPES

Input：**X**-The Individual Group；N-The maximum Evolution Times

Output：$\mathbf{X}_{best}$-The best individual；F_{best}-The fitness of the best individual

Begin

1. Definition of boundary, layer of algorithms and parameters,
2. Population initialization to $\mathbf{X}$
3. While(gen≤G)(Termination Condition G > gen)
4. Dividing layer based on fitness, the initial population of each layer is $\mathbf{X}_t$
5. Use basic PES layer strategy to all $\mathbf{X}_t$ and get $\mathbf{S}$, / * DNL operator * /
6. Select the target individual group $\mathbf{X}'$ in the set $\mathbf{S}$
7. While(size of $\mathbf{X}'$ > 1)(Termination Condition size==1)
8. Find the minimum distance with $\mathbf{x}_a$ and $\mathbf{x}_b$ on population $\mathbf{X}'$, a≠b
9. If(fitness($\mathbf{x}_a$) < =fitness($\mathbf{x}_b$))
10. $\mathbf{y}_k=\mathbf{x}_a+\text{rand}(\mathbf{x}_a-\mathbf{x}_b)$
11. Eliminate $\mathbf{x}_b$ from population.
12. else
13. $\mathbf{y}_k=\mathbf{x}_b+\text{rand}(\mathbf{x}_b-\mathbf{x}_a)$
14. Eliminate x_a from population.
15. End if
16. save $\mathbf{y}_k$ into $\mathbf{Y}$
17. size=size−1
18. End while
19. Compare the $\mathbf{Y}$ and $\mathbf{S}$, save the better individuals to $\mathbf{X}$.
20. gen=gen+1
21. End while
22. Return the best of $\mathbf{X}$ as $\mathbf{X}_{best}$ and the F_{best} of the $\mathbf{X}_{best}$.

End

4 仿真实验与结果分析

为详细评估 DNLPE 算法在函数优化问题上的表现与性能，本文进行了两项测试实验工作：

(1)实验 1　针对金字塔模型，测试中间传递层的数量分别为 1 层、2 层、3 层时，DNLPES 算法在 9 个测试函数上的表现；

(2)实验 2　对比 DNLPES 算法与 GA 算法、PSO 算法、ABC 算法、BBO 算法、KH 算法、KHLD 算法、标准的 PES 算法在 9 个测试函数上的表现。

关于测试函数如表 1 所列，其中，$f_{01}\sim f_{04}$ 是高维单峰函数，$f_{05}\sim f_{09}$ 是低维多峰少极值函数。

表 1　测试函数

Table 1　Benchmark function

公式	名称	维度	范围	极小值
f_{01}	Sphere Model	20	$[-100,100]^d$	0
f_{02}	Schwefel's Problem 1.2	20	$[-100,100]^d$	0
f_{03}	Schwefel's Problem 2.21	20	$[-100,100]^d$	0
f_{04}	Step Function	20	$[-100,100]^d$	0
f_{05}	Kowalik's Function	4	$[-5,5]^d$	0.003075
f_{06}	Branin's Function	2	$[-5,10]\text{U}[0,15]$	0.398
f_{07}	Shekel's Foxholes function	2	$[-65.536,65.536]^d$	1
f_{08}	Goldstein-Price's Function	2	$[-2,2]^d$	3
f_{09}	Hartman's Function 1	3	$[0,1]^d$	−3.86

4.1 实验参数设置

设置算法收敛因子 $\alpha=0.99$，加速步长因子 $\lambda=2$，通用性参数 $c=1/2$。设置种群样本数 $N=50$，$f_{01}\sim f_{04}$ 的迭代次数为 10000，$f_{05}\sim f_{09}$ 的迭代次数为 1000。

(1)实验 1　具体参数设置如下。对于 DNLPES 算法中传递层分别为 1 层、2 层、3 层的模型，分别称为模型 1(M_1)、模型 2(M_2)、模型 3(M_3)。设置模型 1 分层比率 $\boldsymbol{D}=[0.3,0.3,0.4]$，层间传输比 $\boldsymbol{T}=[0.4,0.2]$，初始搜索半径 $\boldsymbol{R}_0=[0.05,0.1,0.55]*(\boldsymbol{U}-V)$；模型 2 分层比率 $\boldsymbol{D}=[0.1,0.2,0.3,0.4]$，层间传输比 $\boldsymbol{T}=[0.6,0.4,0.2]$，初始搜索半径 $\boldsymbol{R}_0=[0.03,0.05,0.1,0.55]*(\boldsymbol{U}-V)$；模型 3 分层比率 $\boldsymbol{D}=[0.1,0.16,0.2,0.24,0.3]$，层间传输比 $\boldsymbol{T}=[0.8,0.5,0.4,0.25]$，初始搜索半径 $\boldsymbol{R}_0=[0.03,0.05,0.1,0.3,0.55]*(\boldsymbol{U}-V)$，其中，$[V,\boldsymbol{U}]$是个体取值范围边界。

(2)实验 2　DNLPES 算法的实验参数参照模型 2，KH 算法参数设置参考文献[10]，KHLD 算法参数设置参考文献[11]，标准 PES 算法参数设置参考文献[17]，GA 算法、PSO 算法参数设置参考文献[6]，ABC 算法参数设置参考文献[13]，BBO 算法参数设置参考文献[12]。

实验工具为 Matlab(R2018a)。实验环境：Windows 10 操作系统，Intel Core i5-7300HQ，2.60 GHz，16 GB 内存。

4.2 实验结果分析

实验 1　通过 50 次独立重复实验，对比 DNLPES 算法的 3 个模型在 9 个测试函数上在误差收敛结果上的最好表现(Best)、平均表现(Mean)、整体方差(Stdev)表现情况，对比结果如表 2 所列。

表 2　DNLPES 算法子模型的结果对比

Table 2　Comparison of results of sub models of DNLPES calculation method

		f_{01}	f_{02}	f_{03}	f_{04}	f_{05}	f_{06}	f_{07}	f_{08}	f_{09}
M_1	Best	3.63×10^{-85}	3.72×10^{-76}	3.69×10^{-43}	0	3.07×10^{-4}	1.74×10^{-13}	0	7.33×10^{-14}	2.00×10^{-14}
	Mean	9.11×10^{-85}	7.22×10^{-71}	7.56×10^{-43}	0	5.80×10^{-4}	2.77×10^{-12}	2.58×10^{-15}	2.46×10^{-11}	8.94×10^{-12}
	Stdev	2.52×10^{-85}	3.8×10^{-70}	6.28×10^{-43}	0	2.07×10^{-4}	2.91×10^{-12}	2.91×10^{-15}	2.05×10^{-11}	1.33×10^{-11}
M_2	Best	**9.74×10^{-86}**	**1.09×10^{-84}**	**2.28×10^{-43}**	0	3.07×10^{-4}	0	0	2.24×10^{-13}	**4.53×10^{-14}**
	Mean	**2.28×10^{-85}**	**2.51×10^{-84}**	**3.75×10^{-43}**	0	6.44×10^{-4}	2.60×10^{-12}	2.70×10^{-15}	5.35×10^{-11}	**6.94×10^{-12}**
	Stdev	**7.77×10^{-86}**	**9.91×10^{-85}**	**7.11×10^{-44}**	0	2.16×10^{-4}	3.82×10^{-12}	2.23×10^{-15}	5.62×10^{-11}	**6.66×10^{-12}**
M_3	Best	1.03×10^{-85}	1.00×10^{-84}	2.87×10^{-43}	0	**3.07×10^{-4}**	**2.82×10^{-14}**	**2.22×10^{-16}**	**4.52×10^{-13}**	1.75×10^{-13}
	Mean	2.91×10^{-85}	4.02×10^{-84}	**4.41×10^{-43}**	0	**4.75×10^{-4}**	**1.87×10^{-12}**	**2.40×10^{-15}**	**2.04×10^{-11}**	7.42×10^{-12}
	Stdev	6.91×10^{-86}	1.63×10^{-84}	9.15×10^{-43}	0	**1.68×10^{-4}**	**1.68×10^{-12}**	**2.11×10^{-15}**	**2.25×10^{-11}**	6.45×10^{-12}

由表 2 可知，DNLPES 的模型 2 在高维测试函数 f_{01}～f_{03}上具备一定优势，在低维测试函数 f_{05}～f_{08}上与最好的模型 3 的实验结果处于同一个量级且相差较小，在低维测试函数 f_{09}上优于模型 3 的测试结果。综合来看，DNLPES 的模型 2 是整体表现最好、最稳定的子模型。

上述实验结果表明，DNLPES 算法中传递层的数量直接影响算法最终的收敛结果。针对 1 层传递层的模型 1，由于模型仅进行了两次局部传递工作，对优秀个体的邻域开采不够充分，这导致算法在收敛精度上表现相对较差；对于 3 层传递层的模型 3，模型进行了 4 次局部传递工作，多次传递操作使得种群以较快的速度收敛于局部最优解，因为过度侧重开采能力降低了算法的全局探索能力的传递性，所以降低了算法的最终收敛结果。因此 DNLPES 模型 2 是算法开采能力与探索能力平衡下的选择。实验 2 中 DNLPES 采用 DNLPES 模型 2 进行实验。

实验 2　通过蒙特卡洛模拟，在多次独立重复实验的情况下，比较 DNLPES 算法与 GA，PSO，ABC，BBO，KH，KHLD，PES 算法在误差收敛结果上的最好表现、平均表现、整体方差表现情况。其中，GA、PSO、ABC、BBO、PES、KH 与 KHLD 算法的实验配置环境与 DNLPES 算法的相同。DNLPES 算法选取 4 层金字塔结构模型进行测试，本文对算法整体进行独立重复的 50 次蒙特卡洛模拟。测试结果如表 3 所列。特别的是，“Best”“Mean”“Stdev”分别表示蒙特卡罗模拟实验总体的最佳误差值、平均误差值、误差方差。

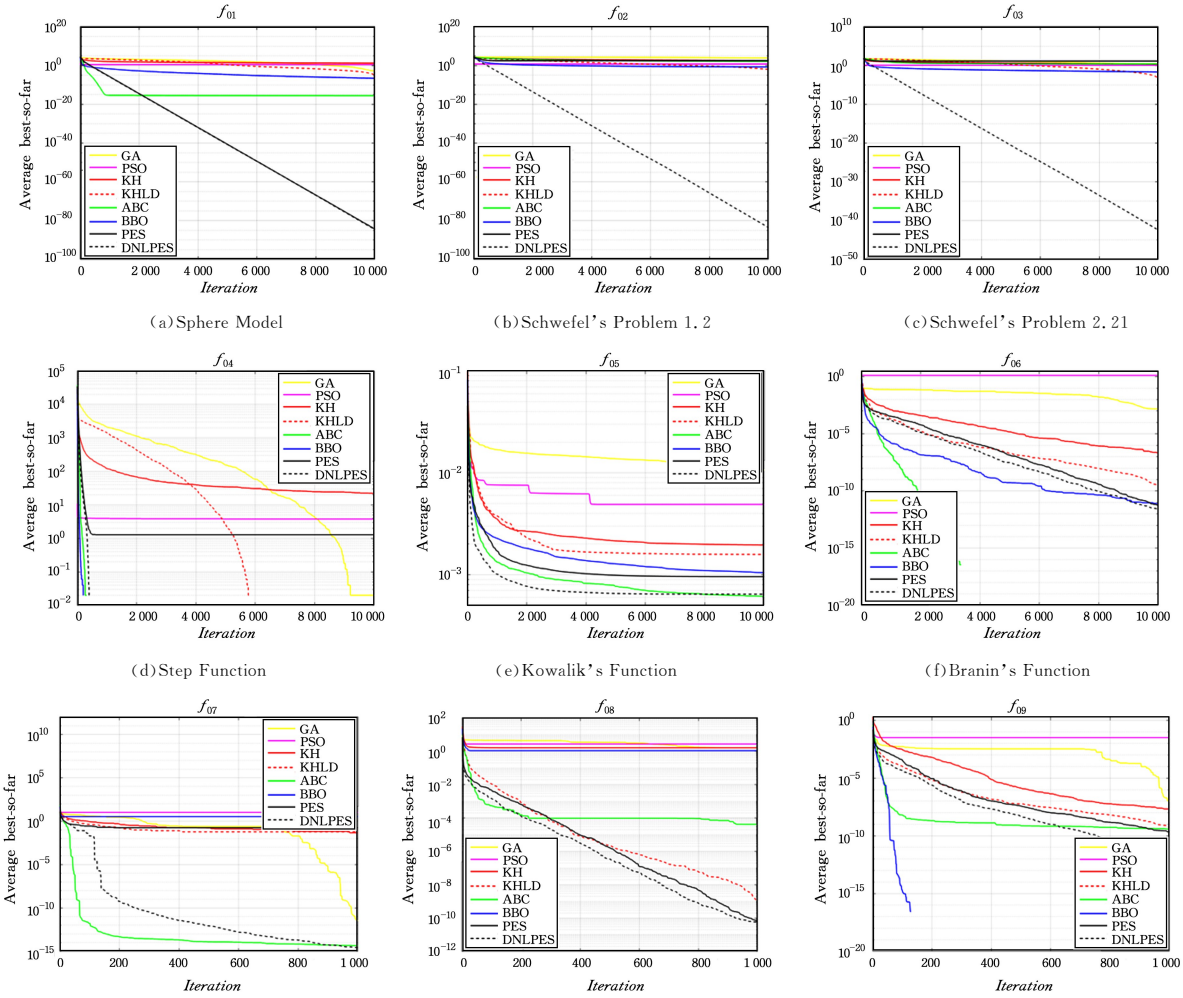
表 3　DNLPES 算法的结果对比

Table 3　Comparison of results of DNLPES calculation method

		GA	PSO	ABC	BBO	KH	KHLD	PES	DNLPES
f_{01}	Best	3.31×10^{-5}	7.06×10^{-2}	2.00×10^{-16}	2.36×10^{-8}	3.51	2.61×10^{-6}	1.59×10^{-85}	$\mathbf{9.74\times10^{-86}}$
	Mean	1.86×10^{-3}	3.13	2.81×10^{-16}	2.56×10^{-7}	1.61×10	9.52×10^{-6}	4.73×10^{-85}	$\mathbf{2.28\times10^{-85}}$
	Stdev	4.45×10^{-3}	1.81	3.40×10^{-17}	2.72×10^{-7}	7.01	4.00×10^{-6}	1.36×10^{-85}	$\mathbf{7.77\times10^{-86}}$
f_{02}	Best	1.89×10^{3}	3.38×10^{-1}	4.24×10	6.41×10^{-2}	5.70×10	1.20×10^{-3}	8.29	$\mathbf{1.09\times10^{-84}}$
	Mean	8.32×10^{3}	5.53	1.43×10^{2}	1.81×10^{-1}	3.25×10^{2}	1.00×10^{-2}	3.28×10^{2}	$\mathbf{2.51\times10^{-84}}$
	Stdev	2.78×10^{3}	2.72	6.47×10	8.85×10^{-2}	1.32×10^{2}	5.86×10^{-3}	3.14×10^{2}	$\mathbf{9.91\times10^{-85}}$
f_{03}	Best	9.03×10^{-1}	1.58×10^{-1}	6.35×10^{-1}	1.17×10^{-2}	1.00	2.27×10^{-4}	2.62×10^{-1}	$\mathbf{2.28\times10^{-43}}$
	Mean	2.93	1.18	2.46	2.16×10^{-2}	2.29	5.52×10^{-4}	1.42×10	$\mathbf{3.75\times10^{-43}}$
	Stdev	1.06	6.25×10^{-1}	5.44×10^{-1}	5.58×10^{-3}	5.09×10^{-1}	1.50×10^{-4}	7.62	$\mathbf{7.11\times10^{-44}}$
f_{04}	Best	0	0	**0**	**0**	0	**0**	0	**0**
	Mean	2.00×10^{-2}	3.78	**0**	**0**	2.18×10	**0**	1.28	**0**
	Stdev	1.41×10^{-1}	2.39	**0**	**0**	9.22	**0**	9.70×10^{-1}	**0**
f_{05}	Best	6.71×10^{-4}	3.14×10^{-4}	$\mathbf{3.41\times10^{-4}}$	3.13×10^{-4}	6.95×10^{-4}	5.25×10^{-4}	3.20×10^{-4}	3.07×10^{-4}
	Mean	1.07×10^{-2}	4.90×10^{-3}	$\mathbf{6.14\times10^{-4}}$	1.05×10^{-3}	1.96×10^{-3}	1.58×10^{-3}	9.59×10^{-4}	6.44×10^{-4}
	Stdev	9.67×10^{-3}	5.43×10^{-3}	$\mathbf{1.23\times10^{-4}}$	2.09×10^{-3}	4.70×10^{-3}	3.88×10^{-3}	1.10×10^{-3}	2.16×10^{-4}
f_{06}	Best	7.77×10^{-10}	1.86×10^{-2}	**0**	0	4.01×10^{-9}	3.72×10^{-12}	8.88×10^{-15}	0
	Mean	1.53×10^{-3}	1.30	**0**	7.78×10^{-12}	2.15×10^{-7}	2.49×10^{-10}	5.06×10^{-12}	2.60×10^{-12}
	Stdev	4.50×10^{-3}	1.06	**0**	3.72×10^{-11}	3.20×10^{-7}	4.61×10^{-10}	5.97×10^{-12}	3.82×10^{-12}
f_{07}	Best	0	1.84×10^{-2}	0	0	9.00×10^{-11}	2.00×10^{-15}	4.44×10^{-16}	**0**
	Mean	3.39×10^{-12}	1.03×10	4.50×10^{-15}	3.36	4.67×10^{-2}	5.96×10^{-2}	1.79×10^{-1}	$\mathbf{2.70\times10^{-15}}$
	Stdev	1.21×10^{-11}	6.33	6.27×10^{-15}	3.45	1.97×10^{-1}	3.11×10^{-1}	5.18×10^{-1}	$\mathbf{2.23\times10^{-15}}$
f_{08}	Best	2.50×10^{-13}	0	0	0	1.57×10^{-9}	9.97×10^{-13}	1.13×10^{-12}	$\mathbf{2.24\times10^{-13}}$
	Mean	1.62	2.70	4.16×10^{-5}	1.08	1.62	1.09×10^{-9}	6.17×10^{-11}	$\mathbf{5.35\times10^{-11}}$
	Stdev	6.49	1.25×10	2.18×10^{-4}	5.34	1.15×10	1.38×10^{-9}	5.59×10^{-11}	$\mathbf{5.62\times10^{-11}}$
f_{09}	Best	0	0	1.60×10^{-14}	**0**	8.61×10^{-11}	1.53×10^{-11}	1.61×10^{-12}	4.53×10^{-14}
	Mean	1.01×10^{-7}	3.18×10^{-2}	3.83×10^{-10}	**0**	2.12×10^{-8}	5.59×10^{-10}	2.20×10^{-10}	6.94×10^{-12}
	Stdev	3.11×10^{-7}	1.53×10^{-1}	5.63×10^{-10}	**0**	2.40×10^{-8}	5.02×10^{-10}	2.78×10^{-10}	6.66×10^{-12}

由表 3 可知，在高维测试函数 f_{01}～f_{03}中，DNLPES 相比其他演化算法表现出了明显的优势；在测试函数 f_4上，DNLPES 与 ABC，BBO，KHLD 算法并列最优，这说明 DNLPES 算法在高维测试环境中相比其他算法具备相同的竞争力；在低维测试函数中，DNLPES 在 f_7和 f_8上表现最优，在 f_5，f_6和 f_9上表现次优，这说明 DNLPES 算法在低维测试环境中也具备一定的竞争力。本文测试进一步对比了 DNLPES 算法与 GA，PSO，ABC，BBO，KH，KHLD 和标准 PES 算法在 8 个函数上的平均误差演化收敛曲线，测试函数 f_{01}～f_{09}平均误差演化收敛效果如图 4 所示。由图 4 可知，相比标准 PES，DNLPES 对于高维函数 f_{01}收敛速度有一定的提升；对于高维函数 f_{02}～f_{03}收敛速度有明显的提升，且 DNLPES 算法在目标中止条件时的收敛效果和整体收敛速度优于其他 7 种算法；对于高维测试函数 f_{04}，DNLPES 算法与 GA，KDLD，ABC，BBO 算法均找到了最优解，但 DNLPES 算法与 ABC，BBO 算法在代数较少时就已经找到了目标最优解，这说明 DNL 算子的加入能提高标准 PES 算法在高维测试函数演化过程中跳出局部最优的能力。如图 4 所示，对于低维测试函数 f_{05}，DNLPES 算法在这 8 种算法中具备最快的收敛速度和次优的最终收敛精度；对于低维测试函数 f_{06}和 f_{09}，DNLPES 算法具备次优的收敛速度和次优的最终收敛精度；对于低维测试函数 f_{07}，DNLPES 算法具备次优的收敛速度和最优的最终收敛精度；对于低维测试函数 f_{08}，DNLPES 算法具备最优的收敛速度和最优的最终收敛精度，这说明 DNL 算子能提高标准 PES 算法在低维测试函数上的收敛速度和跳出局部最优。总的来说，DNLPES 算法相比其他 7 种算法具备较强的竞争力，相比标准 PES 算法，DNLPES 算法在演化前期有较快的收敛速度，在演化的后期，有较强的跳出局部最

优的能力，进而提高算法的求解精度，DNL 算子有效改进了标准 PES 算法种群内部协作不够充分的不足。



(a) Sphere Model (b) Schwefel's Problem 1.2 (c) Schwefel's Problem 2.21

(d) Step Function (e) Kowalik's Function (f) Branin's Function

(g) Shekel's Foxholes Function (h) Goldstein-Price's Function (i) Hartman's Function 1

图 4 平均误差演化迭代图

Fig. 4 Mean error evolution iteration

结束语 本文深入分析了标准 PES 算法演化过程中种群内部协作不够充分的问题，提出了基于动态近邻套索算子的金字塔演化策略，并将其应用于求解复杂函数的优化问题中。对 DNLPES 算法与 GA，PSO，ABC，BBO，KH，KHLD 和 PES 共 7 种算法在 9 个复杂测试函数上进行仿真试验，实验结果表明，DNL 算子能有效改进标准 PES 算法的不足，并且 DNLPES 具备较强的竞争力与鲁棒性。

参 考 文 献

[1] NIRI T D, FAZELI S A S, HEYDARI M. A two-step improved Newton method to solve convex unconstrained optimization problems[J]. Journal of Applied Mathematics and Computing, 2020, 62(1/2): 37-53.

[2] ZANDEVAKILI H, RASHEDI E, MAHANI A. Gravitational search algorithm with both attractive and repulsive forces[J]. Soft Computing, 2019, 23(3): 783-825.

[3] TEODOROVIC D. Swarm intelligence systems for transportation engineering: Principles and applications[J]. Transportation Research Part C: Emerging Technologies, 2008, 16(6): 651-667.

[4] PENG W, LU X C. A hybrid genetic algorithm for function optimization [J]. Journal of Software, 1999, 10(8): 819-823.

[5] RAO A R M, SIVASUBRAMANIAN K. Multi-objective optimal design of fuzzy logic controller using a self configurable swarm intelligence algorithm [J]. Computers & Structures, 2008, 86(23/24): 2141-2154.

[6] POLI R, KENNEDY J, BLACKWELL T. Particle swarm optimization[J]. Swarm Intelligence, 2007, 1(1): 33-57.

[7] ZHU G, KWONG S. Gbest-guided artificial bee colony algorithm for numerical function optimization[J]. Applied Mathematics and Computation, 2010, 217(7): 3166-3173.

[8] LIU J, LAMPINEN J. A fuzzy adaptive differential evolution algorithm[J]. Soft Computing, 2005, 9(6): 448-462.

[9] MAULIK U, BANDYOPADHYAY S. Genetic algorithm-based clustering technique[J]. Pattern Recognition, 2000, 33(9): 1455-1465.

[10] WANG G G, GANDOMI A H, ALAVI A H. Stud krill herd algorithm[J]. Neurocomputing, 2014, 128: 363-370.

[11] LI J, TANG Y, HUA C. An improved krill herd algorithm: Krill herd with linear decreasing step[J]. Applied Mathematics and Computation, 2014, 234: 356-367.

[12] SIMON D. Biogeography-based optimization[J]. IEEE Transactions on Evolutionary Computation，2008，12(6)：702-713.

[13] KARABOGA D，BASTURK B. A powerful and efficient algorithm for numerical function optimization：artificial bee colony (ABC) algorithm[J]. Journal of Global Optimization，2007，39(3)：459-471.

[14] PAN W T. A new fruit fly optimization algorithm：Taking the financial distress model as an example[J]. Knowledge-Based Systems，2012，26：69-74.

[15] SHI Y H. Brain storm optimization algorithm[C]// International Conference in Swarm Intelligence. Berlin Heidelberg：Springer，2011：303-309.

[16] ABRAHAM A，GUO H，LIU H. Swarm intelligence：foundations，perspectives and applications[M]. Berlin Heidelberg：Springer，2006：3-25.

[17] TAN Q. Swarm intelligence evolution strategy based on pyramid structure[D]. Wuhan：Wuhan University of Technology，2018.

[18] TAN Q，HUANG Z C. Krill Herd with Nearest Neighbor Lasso Operator[J]. Computer Engineering and Applications，2019，55(9)：124-129.

ZHANG Qiang，born in 1996，postgraduate. Her main research interests include intelligent computation and so on.

HUANG Zhang-can，born in 1960，Ph.D，professor，Ph. D supervisor. His main research interests include intelligent computation and so on.