

一种基于正向学习和反向学习的改进鸡群算法

张慕雪, 张达敏, 杨菊靖, 朱陈柔玲

(贵州大学 大数据与信息工程学院, 贵州 贵阳 550025)

摘 要: 针对标准的鸡群算法容易陷入局部最优的缺陷, 提出一种基于正向学习和反向学习的改进鸡群算法. 公鸡粒子在每次迭代中向最优粒子正向学习, 使算法迅速进入最有希望的区域寻找食物; 而在算法陷入局部最优解时向最差粒子反向学习以跳出局部最优. 通过对 6 个典型的基准测试函数的仿真表明, 改进算法具有较强的全局搜索能力, 同时寻优精度和收敛速度比原算法也有较大的提高. 尤其是在处理高维函数问题上, 改进算法表现出了较强的优势.

关键词: 鸡群算法; 正向学习; 反向学习

中图分类号: TP18

文献标识码: A

文章编号: 1000-7180(2018)06-0022-06

An Improved Chicken Algorithm Based on Positive Learning and Reverse Learning

ZHANG Mu-xue, ZHANG Da-min, YANG Ju-qing, ZHUCHEN Rou-ling

(College of Big Data and Information Engineering, Guizhou University, Guiyang 550025, China)

Abstract: A Chicken Swarm Optimization with Positive Learning and Reverse Learning (PRCSO) was proposed to solve the defect that traditional Chicken Swarm Optimization (CSO) is easy to fall into local optimum. The cock crow always learns the optimal particles in each iteration, so that the algorithm can quickly enter the optimal region to find food; and when the algorithm falls into the local optimal solution, Learn to jump out of the local optimal. The results on six typical standard test functions show that the improved CSO not only improves the global search ability, but also the search efficiency, search accuracy and convergence rate are better than the traditional CSO. Especially in dealing with high-dimensional function problems, the improved HJCSO shows a strong advantage.

Key words: chicken swarm optimization; positive learning; reverse Learning

1 引言

鸡群算法 (Chicken Swarm Optimization, CSO) 是 2014 年由 Xianbing Meng 等提出的模拟鸡群中的等级制度和觅食行为的一种优化算法^[1-3]. 但标准的鸡群算法由于小鸡粒子容易陷入局部最优解, 仍然存在容易停滞早熟、收敛速度慢等问题^[4]. 文献[1]在小鸡粒子位置更新中加入随机惯性权重和固定学习因子的基础上, 引入模拟退火思想, 使算法在陷入停滞状态时跳出局部最优解, 继续搜索更好的解. 文献[5]在小鸡粒子位置更新中加入指

数递减惯性权重和固定学习因子, 使得小鸡粒子拥有自身惯性权重和向所在子群中公鸡粒子学习的能力. 文献[6]通过种间与种内竞争, 确定子群的规模及等级关系, 采用全局最优引导策略和动态惯性策略, 使个体具有向最优个体学习的能力. 文献[7]在小鸡位置更新过程中加入自我学习系数, 并向所在种群公鸡学习, 利用个体认知多样性, 使个体以一定的概率发生变异, 从而逃离局部最优. 本文提出一种基于正向学习和反向学习的改进鸡群算法 (An Improved Chicken Algorithm Based on Positive Learning and Reverse Learning, PRC-

SO).

2 鸡群算法

鸡群算法模拟自然界中鸡群的等级制度和觅食行为,将搜索的粒子比作鸡群^[8].算法做如下理想化规定:

(1)将整个鸡群分成若干子群,每个子群都由一只公鸡,若干母鸡和一群小鸡组成.

(2)根据每只鸡自身的适应度值,将若干的鸡分成公鸡、母鸡和小鸡.拥有最好适应度值的鸡作为每个子群的公鸡,每个子群中的公鸡是该子群的头目;具有最差适应度值的鸡被指定为子群中的小鸡;剩余个体为母鸡.子群中小鸡和母鸡的母子关系是随机建立的.

(3)鸡群中的等级制度、支配关系和母子关系一旦建立就保持不变,直到进化数代之后才更新.

(4)鸡群中母鸡和小鸡跟随他们子群中的公鸡寻找食物,同时可以阻止其他子群中的鸡抢夺食物.每只小鸡跟随他们的鸡妈妈寻找食物,且小鸡可以随机偷食其他个体发现的食物.在每个子群中公鸡具有良好的竞争优势,它们优先于其他个体找到食物^[9].

公鸡作为子群中具有最好适应度值的粒子,拥有更广泛的区域来接近食物.其对应的位置更新公式如下:

$$x_{i,j}^{t+1} = x_{i,j}^t * (1 + \text{Randn}(0, \sigma^2)) \quad (1)$$

$$\sigma^2 = \begin{cases} 1, & \text{if } f_i \leq f_k \\ \exp\left(\frac{(f_k - f_i)}{|f_i| + \epsilon}\right), & \text{otherwise} \end{cases}$$

$$k \in [1, N], K \neq i \quad (2)$$

式中:Randn(0, σ^2) 为均值为 0, 标准差为 σ^2 的一个高斯分布; ϵ 为一个很小的常熟; k 为所有公鸡中除去 i 后的任一个体.

对于母鸡粒子,它们跟随子群中的公鸡寻找食物.此外,它们还可以随机的偷食其他鸡找到的更好的食物.其位置更新公式如下:

$$x_{i,j}^{t+1} = x_{i,j}^t + S_1 * \text{Rand} * (x_{r_1,j}^t - x_{i,j}^t) + S_2 * \text{Rand} * (x_{r_2,j}^t - x_{i,j}^t) \quad (3)$$

$$S_1 = \exp\left(\frac{f_i - f_{r_1}}{\text{abs}(f_i) + \epsilon}\right) \quad (4)$$

$$S_2 = \exp((f_{r_2} - f_i)) \quad (5)$$

式中:Rand为[0,1]之间均匀分布的随机数; r_1 是第 i 只母鸡所在子群中的公鸡; r_2 是整个鸡群中公鸡和母鸡中随机选取的任意个体,且 $r_1 \neq r_2$.

小鸡围绕着它们的鸡妈妈寻找食物.其位置更新公式如下:

$$x_{i,j}^{t+1} = x_{i,j}^t + \text{FL} * (x_{m,j}^t - x_{i,j}^t) \quad (6)$$

式中: $x_{m,j}^t$ 代表第 i 只小鸡对应鸡妈妈的位置;FL为[0,2]之间服从均匀分布的随机数,表示小鸡跟随鸡妈妈寻找食物的跟随系数,考虑到个体差异,每只小鸡的FL随机在[0,2]之间选择^[1,4-10].

鸡群算法伪代码如下:

初始化:鸡群大小 N , 公鸡粒子个数 N_R , 母鸡粒子个数 N_H , 小鸡粒子个数 N_C , 鸡妈妈粒子个数 N_M , 进化代数 G , 最大迭代次数 M

Begin

1. 计算 N 只鸡的适应度值, $t = 0$;
2. WHILE($t < \text{Max_Generation}$)
3. IF($t \% G == 0$)
4. 对鸡的适应度值进行排序,建立等级制度;
5. 根据排序的适应度值将每只鸡分到不同的群体,并确定小鸡和母鸡之间的母子关系;
6. END IF
7. FOR($i = 1 : N$)
8. IF($i == \text{rooster}$)
9. 根据公式(1)更新公鸡粒子的适应度值;
10. END IF
11. IF($i == \text{hen}$)
12. 根据公式(3)更新母鸡粒子的适应度值;
13. END IF
14. IF($i == \text{chick}$)
15. 根据公式(6)更新小鸡粒子的适应度值;
16. END IF
17. 更新鸡群的粒子当前最好位置和鸡群全局最好位置;
18. END FOR
19. END WHILE
- END

3 改进的鸡群算法

与其他优化算法一样,CSO 希望粒子既能开拓和探索解空间,又能在算法停滞早熟时及时跳出局部最优.根据 CSO 对 10 个标准测试集函数的实验分析发现,在公鸡粒子陷入局部最优解后,由于苛刻的等级制度使得母鸡和小鸡也会随即陷入局部最优而无法跳出^[7].在 CSO 中,公鸡粒子在对解空间进行开拓之后,母鸡和小鸡粒子跟随公鸡对解空间进行探索.

3.1 粒子的正向学习

通过对标准 CSO 的分析和吸取其他优化算法

的经验^[11], 公鸡粒子除寻找食物外还可以向整个鸡群中最优秀的粒子学习, 实现扬长的过程. 改进后公鸡粒子位置更新公式如下:

$$x_{i,j}^{t+1} = x_{i,j}^t * (1 + \text{Randn}(0, \sigma^2)) + Q * (x_{\text{best}}^t - x_{i,j}^t)$$

(7)

式中, x_{best}^t 是第 t 次迭代整个鸡群中的最优位置; Q 是公鸡向最优粒子正向学习的学习系数. 公鸡粒子向群体中的最优粒子正向学习, 能使整个鸡群进入更有希望的区域寻找食物, 避免粒子在整个解空间中盲目搜索, 减少算法的迭代时间.

3.2 粒子的反向学习

虽然上述的正向学习能使算法进入更有效的区域寻找食物, 但当遇到高维度的多峰函数时, 算法仍然容易陷入局部最优且难以跳出. 所以, 公鸡粒子既要有正向学习的能力, 也要有反向学习的功能. 在迭代的过程中对每一次迭代的优化解进行检测判断, 当最优解多次出现不变时, 认为算法出现早熟停滞现象, 此时公鸡粒子应反向学习, 使算法跳出局部最优解. 具体方法为: 当发现最优解多次 (例如 10 次) 出现不变或无明显变化时, 对公鸡粒子进行反向学习操作, 使其发挥避短功能. 此时公鸡粒子的位置更新公式如下:

$$x_{i,j}^{t+1} = x_{i,j}^t * (1 + \text{Randn}(0, \sigma^2)) + K * (x_{\text{worst}}^t - x_{i,j}^t)$$

(8)

式中, x_{worst}^t 是第 t 次迭代整个鸡群中的最差位置; K 是公鸡向最差粒子反向学习的学习系数. 公鸡粒子向群体中的最差粒子反向学习, 能使整个鸡群以一定的概率跳出当前的局部最优区域.

改进后的算法步骤如下:

(1) 初始化鸡群大小 N , 公鸡粒子个数 N_R , 母

鸡粒子个数 N_H , 小鸡粒子个数 N_C , 鸡妈妈粒子个数 N_M , 进化代数 G , 最大迭代次数 M 等相关参数;

(2) 计算每个鸡群粒子的适应度值, $t = 0$;

(3) 判断是否满足结束条件, 满足转步骤(9), 不满足转步骤(4);

(4) 判断是否满足更新鸡群等级制度条件, 满足转步骤(5), 不满足转步骤(6);

(5) 对鸡群粒子的适应度值进行排序, 并建立等级制度, 根据排序的适应度值将每个粒子分到不同的群体, 并随机确定小鸡和母鸡之间的母子关系.

(6) 根据公式(2)~(7)分别对公鸡、母鸡和小鸡粒子进行适应度值更新, 并更新鸡群粒子的当前最优解和最差解;

(7) 判断粒子是否早熟停滞, 满足转步骤(8), 不满足转步骤(4);

(8) 根据公式(8)对公鸡粒子进行反向学习训练, 转步骤(4);

(9) 找到最优解, 输出优化结果.

4 实验仿真

为了测试改进算法 PRCSO 的有效性和客观性, 本文进行了 CSO、PRCSO、PSO、ICSO^[5]、IC-SO^[7] 和 RLPSO^[11] 在 6 个典型的基准测试函数中的优化性能比较. 本文所用的 6 个典型基准测试函数均来自全局优化测试函数库. 函数形式、搜索区间、理论极小值如表 1 所示. 其中, Sphere 和 Rosenbrock 为连续单峰函数, 通常用于衡量算法的收敛速度, 函数 Ackley、Rastrigin、Griewank 和 Schaffor 是复杂的非线性多峰函数, 存在大量的局部极值, 通常用于衡量算法的全局搜索性能.

表 1 算法测试函数

函数名称	函数形式	搜索区间	最小值
$f_1(x)$ Sphere	$f_1(x) = \sum_{i=1}^n x_i^2$	$[-100, 100]$	0
$f_2(x)$ Rosenbrock	$f_2(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i)^2 + (x_i - 1)^2]$	$[-30, 30]$	0
$f_3(x)$ Ackley	$ff_3(x) = 20 + e - 20 \exp(-\frac{1}{5\sqrt{n}} \sum_{i=1}^n x_i^2) - \exp(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i))$	$[-32, 32]$	0
$f_4(x)$ Rastrigin	$f_4(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$	$[-100, 100]$	0
$f_5(x)$ Griewank	$f_5(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]$	0
$f_6(x)$ Schaffor	$f_6(x) = 0.5 + \frac{\sin^2(\sqrt{\sum_{i=1}^n x_i^2} - 0.5)}{[1.0 + 0.001(\sum_{i=1}^n x_i^2)]^2}$	$[-10, 10]$	0

固定种群规模为 100, 维数为 30, 最大迭代次数为 200, 对比标准 CSO、PSO 和改进算法 PRCSO 的

寻优精度、稳定性和迭代次数. 算法具体相关参数如表 2 所示. 为使结果更客观有效,将三种算法分别对选用的 6 个函数进行独立实验 50 次,测试结果如表 3 所示.

表 2 算法相关参数设置

算法	参数
CSO	$N=100, G=10, FL \in \text{rand}(0, 4, 1), D=30$
	$N_R=0.2 \cdot N, N_H=0.6 \cdot N, N_C=N-N_R-N_H$
	$N_M=0.1 \cdot N_H$
PSO	$c_1=c_2=1.5, \omega=0.8, N=100$
	$V_{\max}=5, V_{\min}=-5, D=30$
PRCSO	$N=100, G=10, FL \in \text{rand}(0, 4, 1), D=30$
	$N_R=0.2 \cdot N, N_H=0.6 \cdot N, N_C=N-N_R-N_H$
	$N_M=0.1 \cdot N_H, Q=0.5, K=0.5$

从表 3 可以看出,无论是对连续单峰函数,还是

对复杂的非线性多峰函数,PRCSO 的优化结果都比 CSO 和 PSO 的优化结果要好,尤其是在方差方面,PRCSO 远远优于 CSO 和 PSO,这说明改进算法的改进是有效的,且算法的稳定性得到了很大的提高.

设置最大迭代次数为 200,种群规模为 100,比较 PRC SO 和 CSO 在维度分别为 10 和 100 下的寻优精度,结果如表 3 所示. 当维数为 10 时,标准的 CSO 对函数 $f_4(x)$ 和 $f_5(x)$ 的优化已经能得到最优解,但在维数为 100 时,标准的 CSO 陷入局部最优解且无法跳出. 对函数 $f_1(x)$ 到 $f_6(x)$ 而言,无论是低维度下还是高维度下,PRCSO 的优化结果均远远好于 CSO,尤其在高维度下,PRCSO 表现出了更优的寻优能力.

表 3 CSO,PSO 和 PRC SO 的性能测试对比

测试函数	算法名称	最差解	最优解	平均解	方差	10 维	100 维
$f_1(x)$	PSO	1.377 6	0.352 9	0.971 4	0.131 2	—	—
	CSO	4.052 1e-06	1.461 3e-08	1.106 3e-07	2.043 1e-12	6.389 9e-22	833.753 5
	PRCSO	9.318 2e-11	1.663 0e-12	1.930 2e-11	7.556 0e-22	1.650 4e-37	1.203 6
$f_2(x)$	PSO	272.987 7	74.803 3	197.680 0	3.487 6e+03	—	—
	CSO	6.591 0e-06	1.416 7e-08	2.942 7e-07	4.791 3e-12	6.183 6e-14	3.465 2
	PRCSO	8.351 4e-08	3.277 1e-09	1.521 0e-08	2.463 8e-21	6.215 4e-19	2.361 e-04
$f_3(x)$	PSO	2.563 3	0.377 6	1.337 8	0.525 8	—	—
	CSO	4.383 9e-04	2.673 0e-05	1.513 8e-04	2.529 0e-08	4.858 3e-13	6.558 6
	PRCSO	7.201 9e-07	1.975 2e-07	5.007 4e-07	2.919 1e-14	2.490 1e-15	0.005 2
$f_4(x)$	PSO	195.889 1	110.083 2	155.503 2	861.336 1	—	—
	CSO	3.166 4e-04	4.387 9e-07	8.627 0e-05	1.190 0e-08	0	423.192 4
	PRCSO	6.163 9e-08	1.480 7e-09	1.144 4e-08	3.584 6e-16	0	0.412 5
$f_5(x)$	PSO	0.102 0	0.026 1	0.060 1	7.168 6e-004	—	—
	CSO	1.323 0e-05	6.972 9e-09	2.073 1e-06	1.723 9e-11	0	5.333 7
	PRCSO	9.978 0e-11	1.258 0e-12	2.457 3e-11	9.663 1e-22	0	0.003 4
$f_6(x)$	PSO	0.440 5	0.279 8	0.355 0	0.003 4	—	—
	CSO	1.568 8e-06	5.839 5e-10	2.788 5e-07	3.179 4e-13	3.330 7e-16	0.421 2
	PRCSO	7.753 9e-11	1.733 6e-12	3.084 9e-11	7.258 3e-22	0	6.1727e-05

设置目标精度为 10^{-5} ,在固定测试精度的情况下,比较 PRC SO 和 CSO 初次达到目标精度时的迭代次数,如图 1 所示. 就 6 个函数而言,在达到目标精度时 PRC SO 的迭代次数都少于 CSO 的迭代次数. 在 PRC SO 收敛时,CSO 并还没有收敛,相比较标准的 CSO,PRCSO 能更快速地收敛到全局最优. CSO 在收敛后容易陷入局部最优无法跳出,而

PRCSO 则会继续优化,所以在达到目标精度时,PRCSO 的迭代次数少于 CSO 的迭代次数. 通过上述实验,改进算法的高效性得到了验证.

为比较本文算法与其他改进算法的优劣,将本文算法与参考文献中的几种算法进行比较,引用文献[5]、文献[7]和文献[11]的几组数据,如表 4 所示(“—”表示参考文献未给出).

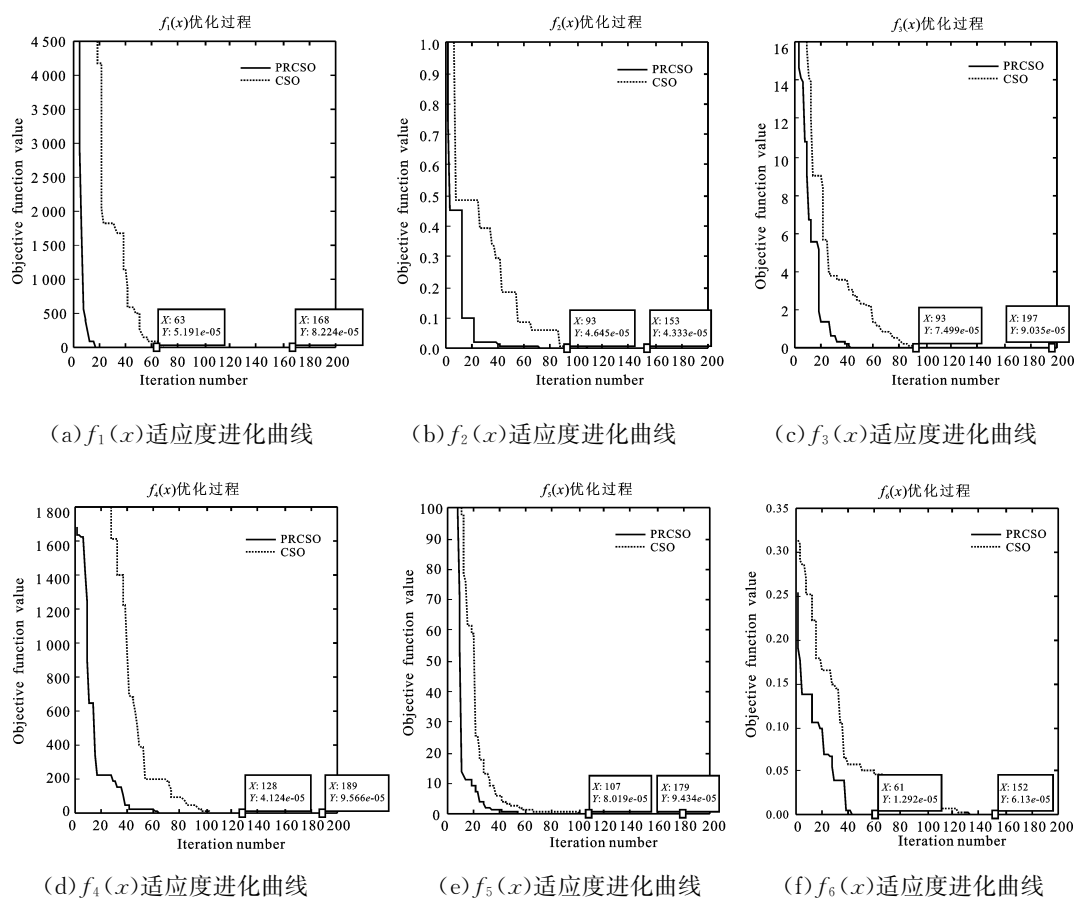


图 1 算法适应度进化曲线

首先,PRCSO 比 CSO、PSO 具有更优的性能,这在前面的实验中已经进行了相关阐述.从表 4-4 可知,ICSO₁的优化解比 PRCSO 的要好,这是因为 ICSO₁的实验结果是在维数为 10,最大迭代次数为 1000 的情况下得出的.由表 3 可知,当维数为 10 时,PRCSO 对函数 $f_4(x)$ 和 $f_5(x)$ 在最大迭代次数为 200 时就能达到最优解为 0 的结果,而对函数 $f_2(x)$ 的寻优结果也远远优于 ICSO₁,综上可知,PRCSO 比 ICSO₁具有更优的寻有能力.对比本文算法和文献[7]算法,ICSO₂是在维数为 20,最大迭代次数为 1 000 的情况下得出的结果.设置维数为 20,最大

迭代次数为 1 000,PRCSO 的寻优结果分别为 $6.2541\text{e-}133$ 、 $2.0175\text{e-}10$ 、0、0、0.除函数 $f_1(x)$ 外,PRCSO 对其余 4 个函数的寻优精度均优于 ICSO₂.对比本文算法和文献[11]算法,PRCSO 在对函数 $f_2(x)$ 、 $f_4(x)$ 和 $f_5(x)$ 进行优化时,PRCSO 表现出了比 RLPSO 更明显的优势,但在对函数 $f_1(x)$ 和 $f_3(x)$ 的优化上,RLPSO 具有更优的优化能力.当增加维数到 100 时,PRCSO 在 6 个函数上的寻有能力均表现出了优于 RLPSO,这表明,在处理高维函数优化问题上,PRCSO 具有更优的优势.

表 4 与参考文献中算法性能的对比

函数名称	平均最优值					
	PRCSO	CSO	PSO	ICSO1 ^[5]	ICSO2 ^[7]	RLPSO ^[11]
$f_1(x)$	$1.930\ 2\text{e-}11$	$1.106\ 3\text{e-}07$	0.971 4	$6.148\ 0\text{e-}132$	$5.265\ 1\text{e-}148$	$1.230\ 0\text{e-}13$
$f_2(x)$	$1.521\ 0\text{e-}08$	$2.9427\text{e-}07$	197.680 0	6.829 1	$1.189\ 5\text{e-}00$	$2.030\ 0\text{e+}02$
$f_3(x)$	$5.007\ 4\text{e-}07$	$1.513\ 8\text{e-}04$	1.337 8	0	$4.440\ 8\text{e-}16$	$4.990\ 0\text{e-}08$
$f_4(x)$	$1.144\ 4\text{e-}08$	$8.627\ 0\text{e-}05$	155.503 2	0	0	$1.560\ 0\text{e+}00$
$f_5(x)$	$2.457\ 3\text{e-}11$	$2.073\ 1\text{e-}06$	0.060 1	0	0	$1.630\ 0\text{e-}02$
$f_6(x)$	$3.084\ 9\text{e-}11$	$2.788\ 5\text{e-}07$	0.355 0	—	—	—

5 结束语

标准的CSO在寻优能力上优于其他的优化算法,但因其小鸡粒子容易陷入局部最优,使算法易出现早熟停滞现象^[12].虽然在处理低维度函数问题时,能较快地收敛到全局最优,但在处理高维多峰函数时,CSO常常陷入局部最优无法跳出.本文模拟人类扬长避短的优点,让公鸡粒子向优秀个体学习的同时,在自身停滞不前的情况下向较差粒子反向学习以跳出局部最优.通过实验验证,改进算法在寻优能力,稳定性,高效性等方面均得到了提高.

参考文献:

- [1] 李振璧,王康,姜媛媛.基于模拟退火的改进鸡群优化算法[J].微电子学与计算机,2017,34(2):30-33.
- [2] 李俊,汪冲,李波,等.基于扰动的精英反向学习粒子群优化算法[J].计算机应用研究,2016,33(9):2584-2587.
- [3] 张建春,康凤举,梁洪涛,等.基于鸡群优化的粒子滤波算法研究[J].系统仿真学报,2017,29(2):295-300.
- [4] Chen S, Yan R. Parameter estimation for chaotic systems based on improved boundary chicken swarm optimization[C]// International Symposium on Optoelectronic Technology and Application 2016. Beijing, China, International Society for Optics and Photonics, 2016:101571K.
- [5] 孔飞,吴定会.一种改进的鸡群算法[J].江南大学学报(自然科学版),2015,14(6):681-688.
- [6] 黄霞,叶春明,郑军.混合改进搜索策略的鸡群优化算法[J/OL]. (2017-03-04). [2017-07-05]. <http://kns.cnki.net/kcms/detail/11.2127.TP.20170304.1727.004.html>.
- [7] 肖亮,刘思彤.基于认知多样性变异的鸡群算法协同优化异步实现[J].计算机科学,2017,44(6A):99-104.
- [8] Chen Y L, He P L, Zhang Y H. Combining penalty function with modified chicken swarm optimization for constrained optimization[C]// First International Conference on Information Sciences, Machinery, Materials and Energy. Pairs: Atlantis Press, 2015: 1899-1907.
- [9] 崔东文.鸡群优化算法-投影寻踪洪旱灾害评估模型[J].水利水电科技进展,2016,36(2):16-23.
- [10] 徐仪勋,李旺,李东东,等.基于改进鸡群算法的非侵入式家电负荷分解[J].电力系统保护与控制,2016,44(13):27-32.
- [11] 夏学文,刘经南等.具备反向学习和局部学习能力的粒子群算法[J].计算机学报,2015,38(7):1397-1407.
- [12] Hedar J. Test functions for unconstrained global optimization[EB/OL]. [2017-07-05]. http://www-optim.amp.i.kyoto-u.ac.jp/member/student/hedar_files/Test_GO_files/Page364.htm.

作者简介:

张慕雪 女,(1992-),硕士研究生.研究方向为智能优化算法、数据挖掘. E-mail: zhangmuxue@foxmail.com.

张达敏 男,(1967-),博士,教授.研究方向为优化计算、网络拥塞控制.

杨菊靖 女,(1993-),硕士研究生.研究方向为数据挖掘.

朱陈柔玲 女,(1993-),硕士研究生.研究方向为数据挖掘.

(上接第21页)

- [9] Khan R, Amjad M. Automatic test case generation for unit software testing using genetic algorithm and mutation analysis[C]// IEEE up Section Conference on Electrical Computer and Electronics. Jaipur, Rajasthan India, IEEE, 2015:1-5.
- [10] Khan R, Amjad M, Srivastava A K. Generation of automatic test cases with mutation analysis and hybrid genetic algorithm[C]// International Conference on Computational Intelligence & Communication Technology. Ghaziabad, India, IEEE, 2017:1-4.

- [11] 丁蕊,董红斌,张岩,等.基于关键点路径的快速测试用例自动生成方法[J].软件学报,2016,27(4):814-827.

作者简介:

李益 女,(1993-),硕士研究生.研究方向为软件测试. E-mail: 249401266@qq.com.

吴军华 女,(1965-),博士,副教授,硕士生导师.研究方法为软件工程.