doi:10.3969/j.issn.1001-4616.2019.03.009

基于精英反向学习的逐维改进蜻蜓算法

何　庆[1,2],黄闽茗[1,2],王　旭[1,2]

(1.贵州大学大数据与信息工程学院,贵州 贵阳 550025)
(2.贵州大学贵州省公共大数据重点实验室,贵州 贵阳 550025)

[**摘要**]　针对蜻蜓算法(DA)寻优精度不高、收敛速度慢及后期搜索活力不足等问题,提出了基于精英反向学习的逐维改进蜻蜓算法(EDDA). 首先,利用精英反向学习策略初始化种群,以增强种群多样性,提高搜索效率;其次,利用逐维更新策略对蜻蜓个体进行更新,减少维间干扰,有效提高了算法的寻优能力;最后,充分利用当前解的信息双向搜索,提升了解的搜索活力. 通过 9 个测试函数的实验结果表明,该算法相比较于标准蜻蜓算法,寻优精度更高、收敛速度更快及后期搜索活力更强,与其他改进算法相比也具有一定的竞争优势.

[**关键词**]　蜻蜓算法,精英反向学习,函数优化,维间干扰
[**中图分类号**]TP301　[**文献标志码**]A　[**文章编号**]1001-4616(2019)03-0065-08

Elite Opposition Learning-Based Dimension by Dimension Improved Dragonfly Algorithm

He Qing[1,2],Huang Minming[1,2],Wang Xu[1,2]

(1.College of Big Data and Information Engineering,Guizhou University,Guiyang 550025,China)
(2.Guizhou Provincial Key Laboratory of Public Big Data,Guizhou University,Guiyang 550025,China)

Abstract:Dragonfly algorithm(DA) is a widely applied algorithm,however,it still has some shortcomings such as low convergence precision,slow convergence speed and weak search vitality. Elite opposition learning-based dimension by dimension improved dragonfly algorithm(EDDA) was proposed. Firstly,the elite opposition-based learning strategy was used to initialize the population,its result enhanced population diversity and improved the search efficiency. Secondly,the individual of dragonfly was updated with the strategy of dimension-by-dimension to reduce interdimensional interference and effectively improved the search ability of the algorithm. Finally,the information of the current solution was fully utilized to bilateral search to enhance the search vitality of the algorithm. The experimental results of nine test functions show that compared with the standard dragonfly algorithm,the proposed algorithm has higher precision,faster convergence speed and stronger search vitality. Compared with other improved algorithms,it has certain competitive advantage.

Key words:dragonfly algorithm,elite opposition-based learning,function optimization,interdimensional interference

蜻蜓算法(Dragonfly Algorithm)是由 Seyedali Mirjalili[1] 在 2015 年提出的一种新型的群智能优化算法. 其思想源于对蜻蜓的社会行为的模拟,具有参数少、易于实现等优点,已经在函数优化[1]、0-1 背包问题[2]以及机器学习的参数优化[3]问题上取得了较好的效果. 尽管蜻蜓算法已经表现出了一定的优势,但仍然存在着求解精度低、收敛速度慢等不足,因此,国内外学者陆续对算法做了进一步的研究和改进.

赵齐辉[4]等人研究了一种差分进化的蜻蜓算法,引入差分进化策略,对蜻蜓个体进行变异、交叉、选择,通过实验结果证明了算法的性能. 吴伟民[5]等人提出了一种基于个体信息交流增强的蜻蜓算法,引入贪婪、平衡和组合 3 种策略,有效解决了蜻蜓个体之间信息交流不充分的问题,实验证明了改进算法具有一定的竞争能力. Sree Ranjini[6] 等人提出了一种具有记忆算子的混合蜻蜓算法,与 PSO 算法相结合增强

收稿日期:2019-07-05.
基金项目:贵州省科技计划项目重大专项(黔科合重大专项字[2018]3002)、贵州省公共大数据重点实验室开放课题(2017BDKFJJ004、2017BDKFJJ034)、贵州省教育厅青年科技人才成长项目(黔科合 KY 字[2016]124)、贵州省科技计划项目重大专项(黔科合重大专项字[2016]3022).
通讯联系人:王旭,博士,副教授,研究方向:机器学习应用、进化计算、量子通讯. E-mail:xuwang@ gzu.edu.cn

了局部开发能力. 实验证明了算法具有一定的优势. 韩鹏[7]等人提出了一种引入混合变异算子和基于拥挤距离的外部档案动态维护策略,在一定程度上较好地改善了算法的性能.

上述研究成果通过改善 DA 算法的收敛速度和收敛精度丰富了 DA 算法的相关改进工作,但都采用整体更新评价策略,即先更新解的所有维度信息,再根据目标函数对所有维度进行整体的评价,这样的评价更新方式使得维间相互干扰,一定程度上将恶化解的收敛效率和收敛速度,影响解的后期收敛活力. 为进一步解决标准 DA 算法在寻优过程中收敛速度慢、求解精度低及后期搜索活力不足等问题,本文提出了一种基于精英反向学习的逐维改进蜻蜓算法(elite opposition learning-based dimension by dimension improved dragonfly algorithm,EDDA). 首先,在算法的初始化阶段,利用精英反向学习策略初始化种群,同时搜索解及其动态反向解,精英保留最优解,以扩大搜索空间,提高搜索性能;然后,在个体更新阶段,利用逐维更新的策略更新蜻蜓个体,利用贪心策略保留评价结果,有效提高算法的寻优能力;最后,利用蜻蜓个体的当前最优解的信息引导解的进一步收敛,提升解的搜索活力. 本文对选取的 9 个测试函数进行实验,结果表明,相比较于标准 DA 算法,本文改进的 EDDA 算法具有更好的全局搜索能力和局部开发能力,具有更好的后期搜索活力,性能优于 DA 算法;相比于其他改进算法,EDDA 算法具有更高的收敛精度,具有一定的竞争力.

1 相关工作

1.1 蜻蜓算法(DA)

蜻蜓算法基本原理是通过对蜻蜓个体之间的社会行为活动进行模拟,和大多数群智能算法相同,蜻蜓个体的行为遵循“求生”的原则,把蜻蜓食物的位置映射为函数优化过程中的解,通过寻找食物源和躲避天敌来进行位置移动. 具体意义和数学表达方式如下[1]:

定义 1 分离度指相邻个体之间避让碰撞的行为:

$$S_i=-\sum_{j=1}^{N}X-X_j. \tag{1}$$

定义 2 对齐度指个体趋于保持相同速度的行为:

$$A_i=\frac{\sum_{j=1}^{N}V_j}{N}. \tag{2}$$

定义 3 内聚度指个体趋于种群中心的行为:

$$C_i=\frac{\sum_{j=1}^{N}V_j}{N}-X. \tag{3}$$

定义 4 食物吸引指蜻蜓个体受食物源吸引的行为:

$$F_i=X^+-X. \tag{4}$$

定义 5 天敌排斥指蜻蜓个体远离天敌的行为:

$$E_i=X^-+X. \tag{5}$$

式中,X 表示当前个体位置,X_j 表示第 j 个相邻个体的位置,N 表示相邻个体的数量. V_j 表示第 j 个相邻个体的速度,X^+表示食物源位置,X^-表示天敌位置. 下一代蜻蜓个体步长更新公式为:

$$\Delta X_{t+1}=(sS_i+aA_i+cC_i+fF_i+eE_i)+w\Delta X_t, \tag{6}$$

式中,s 表示分离权重,a 表示结队权重,c 表示聚集权重,f 表示食物影响因子,e 表示天敌影响因子,w 表示惯性权重,S_i、A_i、C_i、F_i、E_i 分别表示第 i 个蜻蜓个体的分离度、对齐度、内聚度、食物吸引力以及天敌排斥力,t 表示当前迭代次数. 下一代蜻蜓个体位置更新公式为:

$$X_{t+1}=X_t+\Delta X_{t+1}. \tag{7}$$

判断蜻蜓个体之间是否相邻,可以以蜻蜓个体为圆心,画一个半径为 r 的圆,所有在圆内的个体都认为是相邻的,每只蜻蜓的搜索半径随迭代次数的变化更新,半径更新公式如下:

$$r=\frac{a-b}{4}+2(a-b)\frac{t}{t_{\max}}, \tag{8}$$

式中,r 为搜索半径,t 为当前迭代次数,$t_{\max}$为最大迭代次数,a、b 为搜索范围的上下限.

当蜻蜓个体不存在相邻个体时,执行莱维飞行[8],位置更新公式为:

$$X_{t+1}=X_t+\mathrm{Levy}(d)\times X_t, \tag{9}$$

式中,t 表示当前迭代次数,d 表示当前位置的维度.

莱维飞行随机搜索路径公式如下:

$$\mathrm{Levy}(\beta)\sim\frac{\Phi\times u}{|v|^{1/\beta}}(s,\lambda), \tag{10}$$

式中,u 和 v 是服从标准正态分布的随机变量,β 表示莱维飞行的控制因子,通常为 1.5. Φ 的计算公式如下:

$$\Phi=\left[\frac{\Gamma(1+\beta)\times\sin(\pi\times(\beta/2))}{\Gamma((1+\beta)/2)\times\beta\times2^{(\beta-1)/2}}\right]^{\frac{1}{\beta}}. \tag{11}$$

1.2　反向学习

反向学习(opposition-based learning,OBL)在 2005 年由 Tizhoosh[9]提出,是智能计算领域中的一种新策略. 精英反向学习算法的基本思想为:在寻优过程中,同时搜索当前的可行解与动态反向学习的解,并选择较优的可行解保留到下一代[10].

定义 6　反向点[11],设 $X=(x_1,x_2,\cdots,x_D)$ 为 D 维空间中的一点,$x_j\in[a_j,b_j]$,其对应的反向点为:

$$\begin{cases}X'=(x'_1,x'_2,\cdots,x'_D),\\ x'_j=a_j+b_j-x_j.\end{cases} \tag{12}$$

定义 7　动态反向点[12],设 $X_i=(x_{1,i},x_{2,i},\cdots,x_{D,i})$ 为当前可行解,则对应的动态反向点为:

$$\begin{cases}X'_i=(x'_{1,i},x'_{2,i},\cdots,x'_{D,i}),\\ X'_{i,j}=k(a_j+b_j)-x_{i,j},\end{cases} \tag{13}$$

式中,$x_{i,j}\in[a_j,b_j]$,k 为(0,1)之间的随机数,服从均匀分布.

2　EDDA 算法

2.1　基于精英反向学习的种群初始化

在群智能算法中,初始化策略的选择决定了初代种群在解空间的分布情况,而 DA 算法对于初始化策略的构造方法比较敏感[13],因此不同的解空间分布情况将影响算法的搜索能力和收敛效率. 在标准 DA 算法中,对于种群初始化采用了随机生成的方式,虽然这种方法简单易实现,但随机生成的点具有一定的盲目性,使得有的较好的点不能被覆盖,将会减慢算法的求解性能. 为了改善这个问题,本文采用基于精英反向学习的策略对种群进行初始化,在初始化阶段,同时搜索当前解与其动态反向学习的解,将较优的解作为初始解,去除盲目性,在解空间内并行搜索;扩大了种群的多样性,使算法的搜索效率得到提升. 初始化步骤如下:

(1)在搜索空间随机生成 N 个蜻蜓个体初始位置 $x_{i,j}$,其中,$i=1,2,\cdots,N$;$j=1,2,\cdots,D$,N 为种群规模,D 为每个种群的维度;

(2)生成动态反向种群 $x'_{i,j}$,其中 $x'_{i,j}=k(a_j+b_j)-x_{i,j}$,$x_{i,j}\in[a_j,b_j]$;

(3)比较初始化种群与其动态反向学习之后的种群,选择 N 个适应度较好的蜻蜓个体组成初始化种群.

2.2　基于贪婪保留的逐维更新策略

在标准 DA 算法中,采用的是全维度更新再评价策略,即更新所有维度信息之后,再根据目标函数值评价更新后的解,这种方法在一定程度上会掩盖进化维度的信息,浪费了解的评价次数,恶化解的收敛效率[14];同时,这样的更新方式造成了维度与维度之间的相互干扰,从而影响算法的收敛速度和寻优精度.

改进的基于贪婪保留的逐维更新策略的蜻蜓算法,能够考虑每一维度的信息更新. 该策略的思想是:某一维度的值经过更新后与其他维度的值组成新的解;再根据目标函数适应度评价该新解;如果能够改善当前解的质量,则保留该维度对于解的更新结果;反之,则放弃对于当前维度值的更新,保留未更新之前的

维度信息,并进行下一维度的更新,采用这种贪婪保留的方式,直到各维度更新完毕.

利用基于贪婪保留的逐维更新策略,使得解的进化维度得到关注,淘汰退化维度对于解的影响,节约了随机更新所浪费的评价次数;能够有效抑制各维度之间相互干扰,提高了算法的收敛效率,也提升了算法的局部开发能力. 伪代码如下:

```
For j=1 to N
    For i=1 to D
        Y=X(:,i);
        ΔX(j,i)=(a*A(j,i)+c*C(j,i)+s*S(j,i)+f*F(j,i)+e*E(j,i))+w*ΔX(j,i);
        Y(j,i)=X(j,i)+ΔX(j,i)
        If fobj(Y)<fobj(X(:,i))
            接受更新:X(j,i)=Y;
        End
    End
End
```

2.3　基于当前解信息的双向搜索

标准 DA 算法中,当蜻蜓个体至少具有一个相邻个体时,通过式(6)更新步长因子,从而按照式(7)更新下一代蜻蜓个体的位置,这样的机制没有保留历史较优的个体作为下一次迭代的参考值,并且只依赖于 5 种行为和步长来更新下一次迭代的位置,使得算法需要耗费更多的迭代次数来寻找最优解. 因此,为了进一步增强个体间的信息传递,提高算法的寻优能力,可在式(7)中加入历史最优解和当前解的信息,通过差分结果来进一步引导解的收敛,当前解的信息得到充分利用,增强了解的寻优能力;同时,r 作为方向控制因子,可取(-1,1)之间均匀分布的随机数,能够控制搜索方向,达到双向搜索,提高解的收敛效率和寻优能力,因此本文改进的算法把式(7)改进成如下公式:

$$X_{t+1,i}=X_{t,i}+\Delta X_{t+1,i}+r(x_{\text{Food}}-x_{t,i}), \tag{14}$$

式中,r 是(-1,1)之间的随机数,服从均匀分布,x_{Food}表示当前食物源的位置,即历史最优解,$x_{t,i}$为当前解的位置,t 为当前迭代次数. 改进算法的流程如下:

步骤 1　定义目标函数 $F(x)$,设定种群规模 N、最大迭代次数 t、各权重参数及搜索空间维度 D 等;

步骤 2　基于精英反向学习策略初始化种群位置,并初始化步长因子;

步骤 3　计算目标函数值,并更新食物源和天敌的位置,更新与位置移动相关的 5 种行为因子,并计算权重;

步骤 4　更新搜索半径,并判断是否有相邻个体,若存在相邻个体,执行式(14)更新位置,若不存在相邻个体,则通过莱维飞行更新位置,即式(9);

步骤 5　对更新后的蜻蜓个体位置进行逐维更新,考虑每一维度的更新信息,即将当前更新维度的解与其他维度的解组合,能够改善当前解的质量则保留该维度的更新,反之,放弃该维度的更新,通过贪婪保留的方式,更新各维度;

步骤 6　判断是否满足结束条件,若满足,则结束,输出算法的最优解;否则返回步骤 3,直到满足算法结束条件时结束.

改进算法流程图如图 1 所示.

3　实验与结果分析

3.1　测试函数和实验参数设置

本文中,实验的运行环境为 64 位 Windows 7 操作系统,处理器类型为 Intel Core i5-6500,仿真软件为 MATLAB R2016a. 为了验证 EDDA 算法的性能,采用了 9 个不同的标准测试函数进行验证,如表 1 所示,其中 $F_1 \sim F_4$ 为单峰函数,$F_5 \sim F_9$ 是多峰函数. Sphere 函数(F_1)为连续的单峰凸函数,用于评价算法的有效性和收敛精度[15];Rosenbrock 函数(F_5)为典型病态非凸单模函数,极难找到全局最优解,用于评价算法的执行效率[16];Ackley 函数(F_7)为具有深度局部最小的多峰函数,随着维度的增加,局部极小值的个数

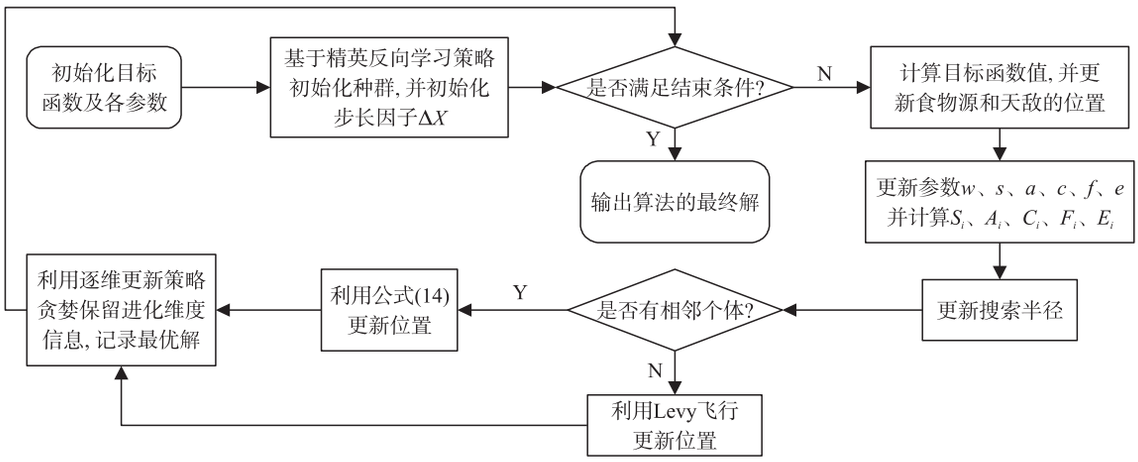

图 1　EDDA 算法流程图

Fig. 1　The schematic diagram of EDDA algorithm

呈指数性增加,用于评价算法的全局搜索能力[17].

本文从两个主要方面来对算法进行测试:(a)针对改进的 EDDA 算法和标准 DA 算法在解的寻优能力方面的比较,(b)针对固定维度函数进行测试,并与其他改进 DA 算法进行比较.

表 1　测试函数

Table 1　Test functions

表达式	维度	范围	理论最优值
$F_1=\sum_{j=1}^{D}(x_j^2)$	10	[-100,100]	0
$F_2=\sum_{j=1}^{D}\lvert x_j\rvert+\prod_{j=1}^{D}\lvert x_j\rvert$	10	[-10,10]	0
$F_3=\sum_{j=1}^{D}\left(\sum_{i-1}^{j}x_j\right)^2$	10	[-100,100]	0
$F_4=\max_j\{\lvert x_j\rvert,1\leqslant j\leqslant D\}$	10	[-100,100]	0
$F_5=\sum_{j=1}^{D-1}[100(x_j^2-x_{j+1})^2+(x_j-1)^2]$	10	[-30,30]	0
$F_6=\sum_{j=i}^{D}(x_j^2-10\cos(2\pi x_j)+10)$	10	[-5.12,5.12]	0
$F_7=-20\exp\left(-0.2\sqrt{\sqrt{\frac{1}{n}\sum_{j=1}^{n}x_j^2}}\right)-\exp\left(\frac{1}{n}\sum_{j=1}^{n}\cos(2\pi x_j)\right)+20+e$	10	[-32,32]	0
$F_8=\frac{1}{4\,000}\sum_{j=i}^{D}x_j^2-\prod_{j=1}^{D}\cos\left(\frac{x_j}{\sqrt{j}}\right)+1$	10	[-600,600]	0
$F_9=\frac{\pi}{D}\left\{10\sin(\pi y_1)+\sum_{j=1}^{D-1}(y_j-1)^2[1+\sin^2(\pi y_{j+1})]+(y_D-1)\right\}+\sum_{j=1}^{D}u(x_j,10,100,4)$ 式中 $y_j=1+\frac{x_j+1}{4}$, $u(x_j,a,k,m)=\begin{cases}k(x_j-a)^m, & x_j>a,\\ 0, & -a<x_j<a,\\ k(-x_j-a)^m, & x_j<-a.\end{cases}$	10	[-50,50]	0

3.2　与标准 DA 算法的比较

3.2.1　寻优能力比较

为了验证 EDDA 算法改进的有效性,本文设定种群规模 $N=30$,最大迭代次数为 500,莱维飞行的步长缩放因子分别为 0.1、1.3[18],测试函数为 $F_1\sim F_9$. 表 2 展示了 EDDA 算法和标准 DA 算法独立运行 30 次实验后的测试结果,分别从平均适应度及其标准差两个方面进行对比,最好结果由粗体凸显显示.

由表 2 可以看出,EDDA 算法相比标准 DA 算法,总体上改善了解的质量. 对于单模测试函数而言,解的质量有了数量级的提高;对于具有众多局部极值点的多模函数来说,EDDA 算法也表现优秀,特别地,对于函数 F_6,DA 算法未能收敛,但 EDDA 算法取得了全局最优值. 由此可得,相比于标准 DA 算法,本文改进的 EDDA 算法具有较高的寻优精度、较强的全局搜索能力及更好的鲁棒性.

表 2　DA 与 EDDA 的寻优精度结果

Table 2　The optimization accuracy values of DA and EDDA

函数	DA		EDDA	
	Mean	Std	Mean	Std
F_1	6.13	11.67	**3.75e−63**	**8.13e−63**
F_2	1.82	0.88	**7.23e−18**	**1.38e−18**
F_3	159.64	259.94	**4.28e−08**	**4.49e−08**
F_4	3.03	2.29	**1.01e−05**	**2.69e−05**
F_5	269.57	456.25	**3.16**	**2.16**
F_6	28.46	11.08	**0.00**	**0.00**
F_7	2.31	1.07	**5.15e−15**	**1.42e−15**
F_8	0.47	0.42	**7.41e−03**	**9.25e−03**
F_9	1.20	0.69	**4.42e−29**	**1.29e−28**

为了进一步验证 EDDA 算法优于标准 DA 算法,图 2(a)~(i)以算法的迭代次数为横轴,函数的适应度值为纵轴,图形化展示了两种算法平均适应度函数的收敛曲线. 从图 2 可以看出,不管是单峰函数还是多峰函数,EDDA 算法的收敛过程明显具有较好的收敛曲线,在收敛前期,改进算法相比于标准的 DA 算法具有更好的收敛能力;在收敛后期,标准 DA 算法收敛曲线平缓,而改进算法收敛曲线陡峭,具有更好的搜索活力,从而验证了表 2 的结果. EDDA 算法基于贪婪保留的逐维更新策略能够及时地跳出局部最优值,从而向全局最优解靠拢,并充分利用历史最优解的信息引导收敛,能够有效提高解的后期收敛效率,从而提高解的寻优能力和后期搜索活力.

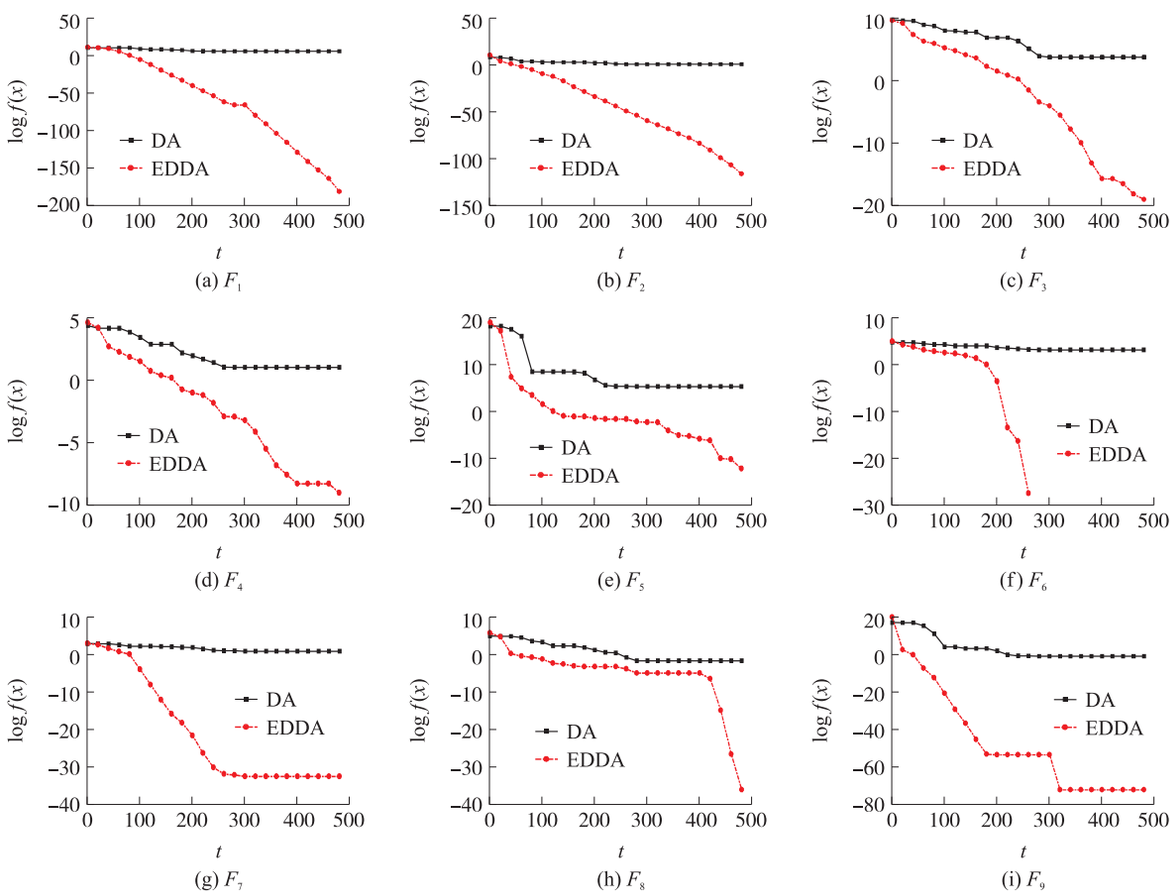

图 2　EDDA 与 DA 寻优收敛曲线图

Fig. 2　The optimization convergence curves of EDDA and DA

3.2.2　维度变化比较

为了观察与分析 EDDA 算法随维度变化的影响,本文通过表 3 展示在维度为 50、100 维时,DA 算法与 EDDA 算法在以相同迭代次数为结束条件时的寻优能力的结果比较. 表中数据为 DA 算法与 EDDA 算法独立运行 30 次之后获得的平均适应度以及标准差,最好结果由粗体凸显显示. 设定种群规模 $N=30$,最大迭代次数为 1 000,莱维飞行的步长缩放因子分别为 0.1、1.3[18],其中,"—"表示最大迭代次数内无法收敛到指定适应度阈值 $1.0\times e^4$.

表 3　不同维度的寻优精度结果

Table 3　The optimization accuracy values with different dimensions

函数	$D=50$				函数	$D=100$			
	DA		EDDA			DA		EDDA	
	Mean	Std	Mean	Std		Mean	Std	Mean	Std
F_1	—	—	**2.93e−19**	**1.46e−19**	F_1	—	—	**6.10e−84**	**3.28e−85**
F_2	29.61	3.53	**4.21e−57**	**8.41e−57**	F_2	79.88	31.32	**1.07e−55**	**3.12e−55**
F_3	—	—	**488.28**	**547.19**	F_4	58.35	6.78	**27.12**	**9.73**
F_4	33.54	4.87	**14.82**	**4.49**	F_5	—	—	**113.9**	**27.28**
F_5	—	—	**42.01**	**0.85**	F_6	767.56	89.91	**1.98e−10**	**3.96e−10**
F_6	333.43	58.23	**0.00**	**0.00**	F_7	12.67	2.93	**6.76e−14**	**9.89e−14**
F_7	10.13	1.87	**2.00e−14**	**5.31e−15**	F_8	172.57	62.47	**0.00**	**0.00**
F_8	37.34	9.65	**2.99e−12**	**1.61e−12**	F_9	—	—	**4.71e−33**	**2.35e−33**
F_9	—	—	**6.64e−21**	**1.33e−20**					

由表 3 可以看出,DA 算法维度从 50 增加到 100 时,函数的寻优能力明显降低,F_1、F_3、F_5、F_9 函数已经无法成功收敛到指定阈值. 反之,本文改进的 EDDA 算法,维度增加时,依然能够收敛于指定精度,体现出算法在求解精度上具有较强的稳定性. 虽然 EDDA 算法在函数 $F_3\sim F_5$ 上并没有成功收敛,逐维评价的策略在搜索前期也将浪费一定的函数迭代次数用于评价维度的进化信息,但是基于贪婪的逐维更新策略,能够加强局部开发能力,有利于获得更好的解. 特别地,维度为 50 时,函数 F_6 成功收敛到理论最优值;维度 100 时,函数 F_8 成功收敛到理论最优值. 由此可见,EDDA 算法采用的精英反向初始化和贪婪保留的逐维更新策略,更好地避免了各维度间的干扰,在高维度函数求解中表现出了更好的稳定性,随着维度的增加,相比 DA 算法仍然具有更好的寻优能力.

3.3　与其他改进算法比较

为了进一步证明改进算法的性能,本文分别与文献[16]中的 DGSDA 算法和文献[19]中的 PSOGSA 算法进行比较,其中,EDDA 算法的种群规模为 30,最大迭代次数为 500,莱维飞行的步长缩放因子分别为 0.1、1.3[18],测试函数为 $F_1\sim F_9$,每个算法独立运行 30 次,其他的算法参数设置由参考文献确定,结果如表 4 所示. 表 4 分别列举了算法的平均适应度值以及标准差,最好的结果用粗体凸显显示.

表 4　EDDA 与其他改进算法的寻优精度结果

Table 4　The optimization accuracy values of EDDA and other improved algorithms

函数	PSOGSA[19]		DGSDA[16]		EDDA	
	Mean	Std	Mean	Std	Mean	Std
F_1	1.72e−20	6.23e−20	1.00e−20	3.82e−20	**3.75e−35**	**1.13e−35**
F_2	1.51	3.58	6.52e−07	1.21e−06	**7.23e−18**	**5.38e−18**
F_3	3.51	6.57	1.63	3.09	**4.28e−08**	**4.49e−08**
F_4	1.89	2.86	0.12	0.24	**1.01e−05**	**2.69e−05**
F_5	464.70	1 215.60	109.90	379.28	**3.16**	**2.16**
F_6	36.11	16.54	11.84	6.51	**0.00**	**0.00**
F_7	0.49	0.79	1.99	4.22	**5.15e−15**	**1.42e−15**
F_8	0.32	0.42	0.19	0.10	**7.41e−03**	**9.25e−03**
F_9	0.53	1.09	0.40	0.70	**4.42e−29**	**1.29e−28**

由表 4 可以看出,针对不同函数,各个算法表现的性能也各不相同. 对于函数 F_1,本文改进的算法与 PSOGSA 算法和 DGSDA 算法相差 5 个数量级或以上;对于函数 F_6,本文改进的算法取得了全局最优解,而其他函数并未收敛;针对函数 $F_3\sim F_5$,其他改进算法性能不理想,但 EDDA 算法能够成功收敛. 表 4 中的

数据显示 EDDA 算法总体上优于其他改进算法,具有较强的竞争优势.

综上所述,对于本文中选取的 9 个测试函数,本文提出的基于精英反向学习的逐维改进蜻蜓算法,采用精英反向学习策略、逐维更新策略及双向搜索的方式,有效提高了 DA 算法的收敛速度、收敛精度以及搜索活力.

4 结语

本文针对 DA 算法的不足,提出了一种基于精英反向学习的逐维改进蜻蜓算法. 该算法从种群的初始化、基于各维度更新的评价策略以及蜻蜓个体位置更新上进行了改进. 通过 9 个测试函数的实验结果证明,在解决函数优化问题时,相比于标准 DA 算法,本文改进的 EDDA 算法表现出的性能更优,相比于其他算法,改进算法具有较强的竞争力. 不足之处为逐维更新的策略将消耗算法的评价次数,并且增加算法的复杂度,以牺牲复杂度来换取算法的精度,本文后续工作将主要分析解决这个问题. 此外,本文的算法还需要在实际优化问题中进行分析验证,例如,应用于机器学习中的参数优化等.

[参考文献]

[1] MIRJALILI S. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems[J]. Neural computing & applications, 2016, 27(4): 1053-1073.

[2] ABDEL B M, LUO Q F, MIAO H, et al. Solving 0-1 knapsack problems by binary dragonfly algorithm[C]//International Conference on Intelligent Computing. Liverpool: Springer, 2017.

[3] THARWAT A, GABEL T, HASSANIEN A E. Parameter optimization of support vector machine using dragonfly algorithm[C]// Proceedings of the International Conference on Advanced Intelligent Systems and Informatics. Cairo: Springer, 2017.

[4] 赵齐辉,杜兆宏,刘升,等. 差分进化的蜻蜓算法[J]. 微电子学与计算,2018,35(7):101-105.

[5] 吴伟民,吴汪洋,林志毅,等. 基于增强个体信息交流的蜻蜓算法[J]. 计算机工程与应用,2017,53(4):10-15.

[6] SREE R K S, MURUGAN S. Memory based hybrid dragonfly algorithm for numerical optimization problems[J]. Expert systems with applications, 2017, 83(1): 63-78.

[7] 韩鹏,陈锋. 一种改进的多目标蜻蜓优化算法[J]. 信息技术与网络安全,2017,36(30):27-31.

[8] VISWANATHAN G M, AFANASYEV V, BULDYREV S V, et al. Levy flight search patterns of wandering albatrosses[J]. Nature, 1996, 361(6581): 413-415.

[9] TIZHOOSH H R. Opposition-based learning: a new scheme for machine intelligence[C]//Computational Intelligence for Modelling. Vienna: Computer society, 2005.

[10] WEI W H, ZHOU J L, FANG C, et al. Constrained differential evolution using generalized opposition-based learning[J]. Acta electronica sinica, 2016, 20(11): 4413-4437.

[11] ZHANG S, LUO Q F, ZHOU Y Q. Hybrid grey wolf optimizer using elite opposition-based learning strategy and simplex method[J]. International journal of computational intelligence and applications, 2017, 16(2): 1-12.

[12] AHANDANI M A, ALAVI R H. Opposition-based learning in shuffled frog leaping: an application for parameter identify cation[J]. Information sciences, 2015, 291(291): 19-42.

[13] 赵挺,孟子航,沈海斌. 基于反向学习与 Levy 飞行的改进蜂群算法[J]. 传感器与微系统,2017,36(1):111-115.

[14] 王李进,尹义龙,钟一文. 逐维改进的布谷鸟搜索算法[J]. 软件学报,2013,24(11):2687-2698.

[15] 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): 281-296.

[16] 马骏,项铁铭. 一种基于佳点集原理与引力搜索的新型蜻蜓算法[J]. 软件导论,2018,12(1):85-89.

[17] 范帅军.布谷鸟搜索算法的应用研究与改进[D]. 成都:西南交通大学. 2016.

[18] CAI Z F, YANG X D. Cuckoo search algorithm with deep search[C]//Proceedings of the 3rd IEEE International Conference on Computer and Communications(ICCC). Chengdu: IEEE Publications, 2018.

[19] MIRJALILI S, HASHIMM S Z M. A new hybrid PSOGSA algorithm for function optimization[C]//2010 International Conference on Computer and Information Application(ICCIA). Tianjin: IEEE Publications, 2010.

[责任编辑:丁 蓉]