doi: 10.12068/j.issn.1005-3026.2020.01.007

基于自适应调整权重和搜索策略的鲸鱼优化算法

孔 芝, 杨青峰, 赵 杰, 熊浚钧
(东北大学秦皇岛分校 控制工程学院, 河北 秦皇岛 066004)

摘 要: 针对鲸鱼优化算法(WOA)收敛速度慢、收敛精度低、易陷入局部最优的问题,提出一种基于自适应调整权重和搜索策略的鲸鱼优化算法(AWOA). 设计一种随着鲸鱼种群变化情况而自适应调整权重的方法,提高了算法的收敛速度; 设计一种自适应调整搜索策略,提高了算法跳出局部最优的能力. 利用23个标准测试函数,分别针对高维和低维问题进行测试,仿真结果表明,AWOA在收敛精度和收敛速度方面总体上明显优于其他多种改进的鲸鱼优化算法.

关 键 词: 鲸鱼优化算法; 自适应调整权重; 自适应调整搜索策略; 函数优化; 全局优化
中图分类号: TP 301.6 **文献标志码**: A **文章编号**: 1005-3026(2020)01-0035-09

Adaptive Adjustment of Weights and Search Strategies-Based Whale Optimization Algorithm

KONG Zhi, *YANG Qing-feng*, *ZHAO Jie*, *XIONG Jun-jun*
(School of Control Engineering, Northeastern University at Qinhuangdao, Qinhuangdao 066004, China. Corresponding author: KONG Zhi, E-mail: kongzhi2004916@163.com)

Abstract: The whale optimization algorithm (WOA) has slow convergence speed and low convergence accuracy and tends to fall into local optimum. In order to solve these problems, a whale optimization algorithm (AWOA) based on adaptive adjustment of weight and search strategy was proposed. An adaptive adjustment of weight with the current distribution of whale population was designed to improve the convergence speed of the algorithm, and an adaptive adjustment of search strategy was designed to improve the ability of the algorithm to jump out of local optimum. Using 23 standard test functions, the algorithm was tested for high-dimensional and low-dimensional problems, respectively. The simulation results showed that the AWOA is generally superior to other improved whale optimization algorithms in terms of convergence accuracy and convergence speed.

Key words: whale optimization algorithm; adaptive adjustment of weight; adaptive adjustment of search strategy; function optimization; global optimization

优化问题一直是计算机科学、人工智能和工程实践等多个领域研究的热点问题. 研究学者受到自然界动物群体行为方式的启发,提出多种智能优化算法,如粒子群优化(particle swarm optimization,PSO)算法[1],蚁群优化(ant colony optimization,ACO)算法[2],差分进化(differential evolution, DE)算法[3],萤火虫算法(firefly algorithm, FA)[4],蝙蝠算法(bat algorithm, BA)[5],灰狼优化(grey wolf optimization,GWO)算法[6]等,为复杂函数优化求解问题提供了强有力的工具,充分证明了智能优化算法强大的优化性能.

鲸鱼优化算法(whale optimization algorithm, WOA)[7]是澳大利亚学者于2016年提出的一种启发式算法,通过模拟海洋中鲸鱼群觅食行为来实现对目标函数优化求解. WOA具有操作简单、需要设置的参数少、寻优性能强等优点,但也存在收敛精度低、容易陷入局部最优的缺陷. 针对这

收稿日期: 2019-07-16
基金项目: 国家自然科学基金资助项目(61402088); 河北省自然科学基金资助项目(F2017501041); 中央高校基本科研业务费专项资金资助项目(N172304030).
作者简介: 孔 芝(1979-),女,辽宁北镇人,东北大学秦皇岛分校副教授.

些不足,国内外很多学者对其进行改进,如郭振洲等[8]提出一种 WOAWC 算法,选取柯西逆累积分布函数对鲸鱼个体进行变异,利用自适应权重提高算法搜索能力;龙文等[9]提出一种 IWOA,通过对立学习策略初始化种群,利用非线性收敛因子增强算法的探索能力,最后利用最优个体变异操作,使得算法在解决大规模优化问题中表现出较强的寻优性能;王坚浩等[10]提出一种 CWOA,引入混沌搜素策略,使得算法的收敛精度和鲁棒性有了很大提升;张永等[11]提出一种 MWOA,通过分段 Logistic 混沌映射的方法进行种群初始化,利用非线性自适应权重平衡局部和全局寻优能力;黄清宝等[12]提出一种 CPWOA,根据余弦曲线变化规律来控制参数,同时加入同步余弦惯性权值和最优鲸鱼个体变异的思想. 目前 WOA 已成功应用于训练神经网络中的多层感知器[13]、电力系统[14]和径向配电网电容器的最佳选址[15]等多个领域. 但是 WOA 在处理更加复杂的优化问题时,无法根据当前计算结果调整参数,得到的解具有随机性. 同时,WOA 在迭代后期,群体中其他鲸鱼均向种群中最优鲸鱼个体靠拢,这就导致种群的多样性缺失,使算法陷入局部最优解.

针对 WOA 所存在的不足,本文提出了基于自适应调整权重和搜索策略的鲸鱼优化算法(adaptive adjustment of weights and search strategies-based whale optimization algorithm, AWOA),利用动态惯性权重,在算法迭代初期,因种群个体比较分散,可以赋予较大的权重值,加快算法的全局搜索能力;在迭代后期,算法可以根据当前种群中个体的分布情况,自适应地改变权值的大小,使其在最优解周围精细搜索、加快收敛速度. 同时利用自适应调整搜索策略,使得算法在迭代前期能以较大的概率在全局范围内随机产生一组解,增加种群的多样性,使其具有更强的全局搜索能力. 利用 13 个基准测试函数、5 个大规模测试函数和 5 个固定维测试函数对该算法进行仿真,结果表明,本文提出的 AWOA 在求解高维度目标函数和较高精度的优化问题时,表现出较强的寻优能力.

1 鲸鱼优化算法

座头鲸是一类群居动物,由于它们只能捕食成群的小型鱼虾,因此进化出一种独特的觅食方式,即泡泡网觅食方式. 根据座头鲸这种独特的狩猎行为,Seyedali 等[7]于 2016 年模拟其群体行为方式,提出了鲸鱼优化算法,该算法主要分为两部分,一部分为泡泡网觅食,另一部分为随机搜索.

1.1 泡泡网觅食

座头鲸觅食行为示意图如图 1 所示,从图中可以看出,鲸鱼通过包围猎物和螺旋更新位置实现局部寻优的目的. WOA 中,搜索空间中的每只鲸鱼的位置代表一个解. 假设当前种群中最接近目标函数值的个体为最优鲸鱼的位置,通过全局最优解的位置信息,群体中其他鲸鱼个体均向最优鲸鱼位置包围来更新其自身位置. 鲸鱼包围猎物行为如式(1)所示:

$$\boldsymbol{X}(t+1)=\boldsymbol{X}^{*}(t)-A\cdot\boldsymbol{D}. \tag{1}$$

式中:$\boldsymbol{D}=|C\cdot\boldsymbol{X}^{*}(t)-\boldsymbol{X}(t)|$;$t$ 为当前迭代次数;$\boldsymbol{X}^{*}$ 表示全局最优鲸鱼位置向量;$\boldsymbol{X}$ 表示当前鲸鱼位置向量;A,C 为矩阵系数,$A=2a\cdot r_1-a$,$C=2\cdot r_2$,r_1,r_2 为[0,1]之间的随机数;$a=2-2t/t_{\max}$,$t_{\max}$表示最大迭代次数.

在螺旋更新位置时,首先计算当前位置与最优鲸鱼个体之间的距离,然后通过螺旋运动方式游向最优个体,这个阶段的数学模型表示如下:

$$\boldsymbol{X}(t+1)=\boldsymbol{X}^{*}(t)+\boldsymbol{D}'_{p}\cdot\mathrm{e}^{bl}\cdot\cos(2\pi l). \tag{2}$$

式中:$\boldsymbol{D}'_{p}=|\boldsymbol{X}^{*}(t)-\boldsymbol{X}(t)|$;$b$ 为常数,用于限定螺旋的形状;l 是[-1,1]之间的随机数.

鲸鱼通过螺旋形游向猎物的同时又以收缩包围的方式靠近猎物,此行为即为泡泡网觅食行为. 在 WOA 中,当$|A|<1$ 时,鲸鱼在包围圈内寻优,进行局部最优搜索,以概率 0.5 实行包围猎物行为和螺旋更新行为,其数学模型如下:

$$\boldsymbol{X}(t+1)=\begin{cases}\boldsymbol{X}^{*}(t)-A\cdot\boldsymbol{D}, & p<0.5;\\ \boldsymbol{X}^{*}(t)+\boldsymbol{D}'_{p}\cdot\mathrm{e}^{bl}\cdot\cos(2\pi l), & p\geqslant 0.5.\end{cases} \tag{3}$$

式中,p 为[0,1]之间的随机数.

图 1 鲸鱼的泡泡网觅食行为

Fig. 1 Bubble-net attack behavior of whales

1.2 随机搜索

$|A|\geqslant 1$ 时,鲸鱼在收缩包围圈外寻优,进行随机搜索. 算法在当前鲸鱼种群中随机选取一个鲸鱼个体作为全局最优解,种群中其他鲸鱼向其聚拢. 通过这种方式更新种群的位置,增强了鲸鱼

种群的多样性,算法的全局搜索能力得到增强,其数学模型公式如下：

$$\boldsymbol{X}(t+1)=\boldsymbol{X}_{\text{rand}}-A\cdot\boldsymbol{D}'. \tag{4}$$

式中,$\boldsymbol{D}'=|C\cdot\boldsymbol{X}_{\text{rand}}-\boldsymbol{X}|$,$\boldsymbol{X}_{\text{rand}}$表示随机选取的鲸鱼位置向量.

2　鲸鱼优化算法的改进

2.1　自适应调整权重

惯性权重作为鲸鱼优化算法中一个重要的参数,对目标函数的优化求解起到了很大的作用. 合适的权重值对算法寻优能力的提升有很大帮助. 由于 WOA 在优化求解的过程中,线性的惯性权重调整策略若选择不合适,将影响算法的收敛速度. 因此,本文提出了一种根据当前鲸鱼种群分布情况来自适应改变权值的大小, 公式如下：

$$w=d_1\cdot(\boldsymbol{P}_{i\text{worst}}-\boldsymbol{P}_{i\text{best}})+d_2\cdot(x_i^{\text{upper}}-x_i^{\text{lower}})/n_{\text{g}}. \tag{5}$$

式中:n_{g} 表示当前种群的迭代次数;$x_{\text{i}}^{\text{upper}}$ 和 $x_{\text{i}}^{\text{lower}}$ 分别为变量 x_i 的上界和下界;$\boldsymbol{P}_{i\text{worst}}$和 $\boldsymbol{P}_{i\text{best}}$分别为当前鲸鱼种群中最差鲸鱼的位置向量和最优鲸鱼的位置向量;d_1 和 d_2 是两个常数. 因此,当前鲸鱼个体自适应调整权重收缩包围更新位置和自适应调整权重螺旋更新位置公式如下：

$$\boldsymbol{X}(t+1)=w\cdot\boldsymbol{X}^*(t)-A\cdot\boldsymbol{D}, \tag{6}$$

$$\boldsymbol{X}(t+1)=w\cdot\boldsymbol{X}^*(t)+\boldsymbol{D}'_p\cdot\mathrm{e}^{bl}\cdot\cos(2\pi l). \tag{7}$$

在引入自适应调整权重搜索策略后, 算法可以根据当前鲸鱼种群的分布情况自适应地改变权值的大小. 在算法迭代初期,若鲸鱼种群陷入局部最优解, 并且最优解和最差解差别不大时, $d_2\cdot(x_i^{\text{upper}}-x_i^{\text{lower}})/n_{\text{g}}$ 的值并不受种群分布情况的影响,这部分的设计依然可以得到较大的权重值 w,避免了算法在迭代初期就陷入小范围搜索的缺陷;随着鲸鱼种群迭代次数的增加,$d_2\cdot(x_i^{\text{upper}}-x_i^{\text{lower}})/n_{\text{g}}$ 的值会逐渐变小,其对权重 w 的影响减小,若此时算法并未得到最优解,$d_1\cdot(\boldsymbol{P}_{i\text{worst}}-\boldsymbol{P}_{i\text{best}})$的设计就可以对权重值 w 起到主导作用,可以使算法以较大的步长寻优. 这样设计自适应调整权重 w 的好处在于,其值由两部分决定,前半部分对种群迭代次数过大时起主要调节作用,后半部分对种群陷入局部最优时起主要调节作用. 权重 w 前后两部分会根据当前种群位置的变化情况而发生变化, 不拘于某种固定的形式, 具有很强的自适应性.

2.2　自适应调整搜索策略

为防止算法陷入局部最优,随机搜索阶段,个体根据概率阈值 Q 来选取随机搜索的更新方式,概率阈值定义为

$$Q=\frac{|\bar{f}-f_{\min}|}{|f_{\max}-f_{\min}|}. \tag{8}$$

式中:$\bar{f}$ 表示当前鲸鱼种群的平均适应度值; $f_{\min}$ 为当前鲸鱼种群中最好的适应度值;$f_{\max}$ 为当前鲸鱼种群中最差的适应度值. 对于每个鲸鱼个体,以一个[0,1]之间的随机数 q 与计算出的概率阈值 Q 进行数值比较. 若 $q<Q$,随机选取的鲸鱼个体 $\boldsymbol{X}_{\text{rand}}$根据式(9)更新其位置,其他鲸鱼个体位置不变; 否则,其他鲸鱼个体根据式(4)更新其位置. 这样设计使得算法在迭代前期能以较大的概率在全局范围内随机产生一组解,避免鲸鱼因聚集在一起而导致种群多样性的缺失, 增强了算法的全局搜索能力.

$$\boldsymbol{X}_{\text{rand}}=\boldsymbol{X}_{j\min}+r\cdot(\boldsymbol{X}_{j\max}-\boldsymbol{X}_{j\min}). \tag{9}$$

式中:r 为[0,1]之间的随机数;$\boldsymbol{X}_{\min}$,$\boldsymbol{X}_{\max}$分别为变量 $\boldsymbol{X}_{\text{rand}}$取值的最小值和最大值.

2.3　AWOA 流程图及步骤

综上所述,本文提出的 AWOA 执行过程中的流程图如图 2 所示,具体说明如下：

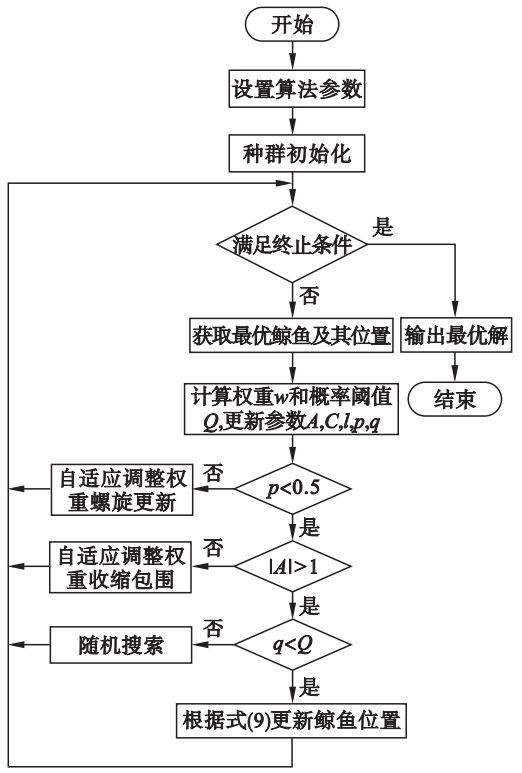

图 2　AWOA 流程图

Fig. 2　Flow chart of AWOA

步骤 1　参数初始化. 设置种群规模为 N,最大迭代次数 $t_{\max}$; 设置参数 d_1,d_2;

步骤 2　种群初始化. 随机生成初始解,计算个体适应度$\{F(X_i),i=1,2,\cdots,N\}$,并记录全局最优解;

步骤 3　根据式(5)和式(8)计算权重 w 和概率阈值 Q; 更新参数 A,C,l,p 和 q;

步骤 4　如果 $p \geqslant 0.5$,根据式(7)更新当前鲸鱼位置;

步骤 5　如果 $p < 0.5$ 并且 $|A| \leqslant 1$,根据式(6)更新当前鲸鱼位置;

步骤 6　如果 $p < 0.5$,$|A| > 1$ 并且 $q < Q$,根据式(9)更新当前鲸鱼位置;否则根据式(4)更新当前鲸鱼位置;

步骤 7　判断算法是否满足终止条件,若是,则输出最优解;否则返回步骤 3.

3　仿真对比与分析

3.1　测试函数与性能指标

选取的测试函数为文献[7]中的 13 个标准测试函数($F_1(x) \sim F_{13}(x)$);另外,为验证 AWOA 的可行性,本文选取 5 个典型的复杂高维函数($n=200,500,1\,000$ 和 $1\,500$ 维)和 5 个固定维多峰函数($n=2,4$ 维)进行对比仿真. 测试函数如表 1 和表 2 所示.

表 1　高维测试函数

Table 1　High dimensional test functions

函数名	测试函数表达式	搜索范围	最优值	收敛精度
Sphere	$f_1(x)=\sum_{i=1}^{n} x_i^2$	$[-100,100]$	0	1×10^{-8}
Rosenbrock	$f_2(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	$[-30,30]$	0	1×10^{0}
Step	$f_3(x)=\sum_{i=1}^{n}(x_i+0.5)^2$	$[-100,100]$	0	1×10^{-8}
Penalized1	$f_4(x)=\frac{\pi}{n}\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4)$, $y_i=1+\frac{x_i+1}{4}$, $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a<x_i<a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$	$[-50,50]$	0	1×10^{-2}
Generalized Penalized	$f_5(x)=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+\sin^2(3\pi x_i+1)]+(x_n-1)^2[1+\sin^2(2\pi x_n)]\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	$[-50,50]$	0	1×10^{-2}

表 2　固定维测试函数

Table 2　Fixed dimension test functions

函数名	测试函数表达式	维数	搜索区间	最小值
Shekel's Foxholes	$f_6(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	2	$[-65,65]$	1
Kowalik	$f_7(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	4	$[-5,5]$	0.000 30
Shekel's Family	$f_8(x)=-\sum_{i=1}^{7}[(\boldsymbol{X}-a_i)(\boldsymbol{X}-a_i)^{\mathrm{T}}+c_i]^{-1}$	4	$[0,10]$	−10.402 8
Shekel's Family4	$f_9(x)=-\sum_{i=1}^{10}[(\boldsymbol{X}-a_i)(\boldsymbol{X}-a_i)^{\mathrm{T}}+c_i]^{-1}$	4	$[0,10]$	−10.536 3
Easom	$f_{10}(x)=-\cos(x_1)\cos(x_2)\exp(-(x_1-\pi)^2-(x_2-\pi)^2)$	2	$[-100,100]$	−1

选取平均值、标准差和成功率 3 个性能指标(运行 30 次)来分析算法的优化性能.

3.2　仿真结果分析

仿真在 Inter(R) Core(TM) CUP: i7 − 4510U, 4GB 内存,2.00 GHz 主频的计算机上运行,程序采用 MATLAB 2015a 编程实现.

通过比较 AWOA 与标准 WOA[7], WOAWC[8], IWOA[9], CWOA[10], CPWOA[12] 的仿真结果来验证 AWOA 的寻优能力. 因 CPWOA 算法在文献[12]中已经和其他算法(象群优化(EHO)算法[16]、灰狼优化(GWO)算法[6]、正余弦算法[17](SCA)等)做了仿真对比,且 CPWOA 在收敛速度与精度上优于这些算法. 因此,这里不再将 AWOA 与其他优化算法作

比较.

经过反复多次仿真,对 AWOA 算法的最优参数设置如下:常数 $b=1$,$d_1=1\times10^{-4}$,$d_2=1\times10^{-4}$. 标准 WOA, IWOA, WOAWC, CWOA 和 CPWOA 中的其他参数均按照其对应参考文献设置. 表 3 给出了 6 种算法对 13 个标准测试函数($F_1(x)\sim F_{13}(x)$,$n=30$ 维)的仿真结果,其中标准 WOA 的结果来源于文献[7],IWOA 的结果来源于文献[9]. 表 4 为 6 种算法对 5 个高维测试函数($n=200,500,1\,000,1\,500$ 维)的仿真数据. 表 5 为 6 种算法对 5 个固定维测试函数($n=2,4$ 维)的仿真数据比较. 仿真中最好的结果以黑色加粗字体表示.

表 3　6 种算法对 13 个低维测试函数的寻优性能比较

Table 3　Comparison of optimal performance of 6 algorithms for 13 low-dimensional test functions

函数	标准 WOA[7]		IWOA[9]		WOAWC	
	平均值	标准差	平均值	标准差	平均值	标准差
$F_1(x)$	1.41E−030	4.91E−030	6.54E−125	6.80E−125	**0**	**0**
$F_2(x)$	1.06E−021	2.39E−021	2.15E−073	3.64E−073	3.95E−180	**0**
$F_3(x)$	5.39E−007	2.93E−006	1.56E−023	1.81E−023	**0**	**0**
$F_4(x)$	0.072 581	0.397 47	7.06E−007	2.18E−006	1.70E−184	**0**
$F_5(x)$	27.865 58	0.763 626	27.279 50	0.215 438	28.585 1	0.095 7
$F_6(x)$	3.116 266	0.532 429	**0**	**0**	0.105 2	0.035 7
$F_7(x)$	0.001 425	0.001 149	2.42E−004	4.41E−004	1.68E−004	2.05E−004
$F_8(x)$	−5 080.76	695.796 8	−1.14E+004	112.636 1	−1.00E+004	2.34E+003
$F_9(x)$	**0**	**0**	**0**	**0**	**0**	**0**
$F_{10}(x)$	7.404 3	9.897 572	3.02E−015	1.95E−015	**8.88E−016**	**0**
$F_{11}(x)$	0.000 289	0.001 586	**0**	**0**	**0**	**0**
$F_{12}(x)$	0.339 676	0.214 864	0.087 646	0.011 997	0.006 7	0.002 8
$F_{13}(x)$	1.889 015	0.266 088	0.466 442	0.248 928	0.128 3	0.061 2

函数	CWOA		CPWOA		AWOA	
	平均值	标准差	平均值	标准差	平均值	标准差
$F_1(x)$	**0**	**0**	3.03E−024	1.47E−023	**0**	**0**
$F_2(x)$	**0**	**0**	1.88E−024	8.22E−024	**0**	**0**
$F_3(x)$	**0**	**0**	5.47E+004	1.25E+004	**0**	**0**
$F_4(x)$	**0**	**0**	33.357 4	25.403 8	**0**	**0**
$F_5(x)$	28.157 7	0.396 1	22.493 9	11.244 8	**0.001 0**	**0.002 4**
$F_6(x)$	0.909 3	0.396 6	0.064 5	0.032 6	1.32E−005	4.05E−005
$F_7(x)$	0.476 4	0.291 5	0.012 3	0.014 7	**1.22E−004**	**1.07E−004**
$F_8(x)$	−1.09E+004	1.85E+003	**−1.25E+004**	743.069 6	**−1.25E+004**	**0.184 7**
$F_9(x)$	**0**	**0**	3.940 1	15.287 6	**0**	**0**
$F_{10}(x)$	**8.88E−016**	**0**	9.10E−014	7.07E−014	**8.88E−016**	**0**
$F_{11}(x)$	**0**	**0**	0.018 2	0.061 9	**0**	**0**
$F_{12}(x)$	0.097 3	0.103 2	0.007 2	0.006 5	**5.63E−007**	**8.51E−007**
$F_{13}(x)$	1.368 6	0.631 5	0.171 3	0.100 1	**3.02E−006**	**4.53E−006**

注:种群规模 $N=30$, 最大迭代次数 $t_{\max}=500$.

由表 3 中的仿真数据可知, AWOA 和 CPWOA 在测试函数 $F_1(x)$, $F_2(x)$, $F_3(x)$, $F_4(x)$, $F_9(x)$ 和 $F_{11}(x)$ 上均可以收敛到理论最优值 0, 表现出较好的寻优能力. AWOA 除在测试函数 $F_6(x)$ 上的搜索结果差于 IWOA 外,在其他测试函数上均优于标准 WOA, IWAO, WOAWC, CWOA 和 CPWOA,表明 AWOA 具有更强的寻优性能和稳定性.

由表 4 中的仿真数据可知,针对高维函数的优化问题,AWOA对目标函数的求解精度和寻优

表 4 6 种算法对 5 个高维测试函数的寻优性能比较

Table 4 Comparison of optimal performance of 6 algorithms for 5 high-dimensional test functions

函数	算法	200 维			500 维		
		平均值	标准差	成功率/%	平均值	标准差	成功率/%
$f_1(x)$	WOA	9.28E-068	5.07E-067	**100**	3.11E-067	1.32E-066	**100**
	IWOA	4.09E-106	1.78E-105	**100**	4.60E-104	2.00E-103	**100**
	WOAWC	1.55E-283	**0**	**100**	6.10E-280	**0**	**100**
	CWOA	**0**	**0**	**100**	**0**	**0**	**100**
	CPWOA	2.95E-022	8.40E-022	**100**	2.01E-020	1.02E-019	**100**
	AWOA	**0**	**0**	**100**	**0**	**0**	**100**
$f_2(x)$	WOA	197.659 0	0.132 5	0	495.849 3	0.354 3	0
	IWOA	197.693 9	0.230 0	0	495.446 0	0.724 5	0
	WOAWC	197.076 5	0.022 0	0	494.166 5	**0.061 1**	0
	CWOA	197.666 2	0.209 5	0	495.959 0	0.568 4	0
	CPWOA	109.483 1	95.811 5	10	309.244 6	221.398 2	0
	AWOA	**0.004 9**	**0.008 2**	**100**	**6.543 6**	12.373 5	**50**
$f_3(x)$	WOA	10.847 8	3.059 8	0	28.580 1	7.293 5	0
	IWOA	19.172 6	6.444 9	0	47.136 6	14.213 8	0
	WOAWC	4.604 5	0.724 0	0	12.310 6	**2.195 3**	0
	CWOA	15.805 8	4.622 0	0	43.870 2	15.923 3	0
	CPWOA	3.123 5	1.163 9	0	9.093 9	2.761 4	0
	AWOA	**1.98E-005**	**3.17E-005**	**3.33**	**2.225 8**	4.867 8	0
$f_4(x)$	WOA	0.093 3	0.050 1	0	0.122 8	0.054 6	0
	IWOA	0.180 1	0.090 8	0	0.184 9	0.064 0	0
	WOAWC	0.024 9	0.008 6	6.67	0.031 3	0.007 5	0
	CWOA	0.138 4	0.051 5	0	0.143 2	0.083 6	0
	CPWOA	0.021 4	0.025 8	20.00	0.022 0	0.017 3	10.00
	AWOA	**1.18E-007**	**1.77E-007**	**100**	**0.001 3**	**0.002 7**	**96.67**
$f_5(x)$	WOA	4.827 5	1.715 8	0	12.307 4	3.838 5	0
	IWOA	8.505 7	3.472 7	0	22.355 7	10.740 8	0
	WOAWC	1.090 3	0.418 6	0	2.918 9	1.116 1	0
	CWOA	11.460 1	3.770 3	0	29.612 2	7.921 9	0
	CPWOA	1.182 5	0.445 3	0	3.021 0	1.014 1	0
	AWOA	**9.78E-006**	**1.39E-005**	**100**	**0.272 6**	**0.820 3**	**46.67**
函数	算法	1 000 维			1 500 维		
		平均值	标准差	成功率/%	平均值	标准差	成功率/%
$f_1(x)$	WOA	2.48E-069	9.14E-069	**100**	2.25E-069	8.59E-069	**100**
	IWOA	8.29E-111	4.24E-110	**100**	4.17E-106	1.44E-105	**100**
	WOAWC	1.57E-291	**0**	**100**	4.06E-288	**0**	**100**
	CWOA	**0**	**0**	**100**	**0**	**0**	**100**
	CPWOA	3.92E-020	1.07E-019	**100**	2.60E-020	1.07E-019	**100**
	AWOA	**0**	**0**	**100**	1.75E-019	1.57E-018	**100**
$f_2(x)$	WOA	994.032 1	0.863 4	0	1.49E+03	1.416 3	0
	IWOA	992.522 4	1.245 2	0	1.48E+03	1.409 1	0
	WOAWC	989.290 4	**0.047 1**	0	1.48E+03	**0.176 3**	0
	CWOA	993.186 2	1.224 5	0	1.49E+03	1.984 5	0
	CPWOA	**399.738 9**	441.694 2	0	770.527 0	695.366 6	0
	AWOA	455.284 4	433.964 5	**6.67**	**542.269 2**	581.482 7	**3.33**

续表

函数	算法	1 000 维			1 500 维		
		平均值	标准差	成功率/%	平均值	标准差	成功率/%
$f_3(x)$	WOA	68.885 9	19.067 5	0	105.956 2	21.198 5	0
	IWOA	93.475 3	25.104 6	0	147.096 2	51.800 6	0
	WOAWC	24.978 4	**1.993 3**	0	38.438 4	**7.442 7**	0
	CWOA	75.233 2	31.722 0	0	128.203 0	43.187 7	0
	CPWOA	20.552 1	5.058 2	0	30.752 8	9.256 0	0
	AWOA	**9.566 2**	17.689 7	0	**22.330 6**	42.048 3	0
$f_4(x)$	WOA	0.104 7	0.053 5	0	0.167 5	0.066 8	0
	IWOA	0.151 8	0.056 2	0	0.150 8	0.064 2	0
	WOAWC	**0.030 7**	**0.007 1**	0	0.028 3	**0.008 4**	3.33
	CWOA	0.196 0	0.097 3	0	0.141 0	0.082 0	0
	CPWOA	0.045 0	0.046 9	10.00	0.042 8	0.083 6	3.33
	AWOA	0.048 6	0.150 6	**73.33**	**0.011 6**	0.026 2	**80.00**
$f_5(x)$	WOA	25.654 6	7.034 9	0	34.400 3	9.643 8	0
	IWOA	41.590 4	16.916 1	0	68.875 5	31.093 5	0
	WOAWC	5.946 4	2.059 5	0	7.417 9	3.636 9	0
	CWOA	61.813 6	11.404 6	0	99.381 0	22.620 5	0
	CPWOA	5.682 0	**1.995 6**	0	10.125 8	3.748 4	0
	AWOA	**1.661 5**	4.363 4	**20.00**	**1.111 0**	**2.861 6**	**6.67**

注：种群规模 $N=30$，最大迭代次数 $t_{\max}=500$.

成功率上总体要优于其他 5 种算法的仿真结果. 对于测试函数 $f_1(x)$，6 种算法在 $n=200$，$n=500$，$n=1\,000$ 和 $n=1\,500$ 维时的收敛成功率均可达到 100%，但 CWOA 均可以收敛到理论最优值 0，表现出更强的寻优能力和稳定性. 然而 AWOA 对于测试函数 $f_4(x)$ 在 $n=200$，$n=500$，$n=1\,000$ 和 $n=1\,500$ 维下的寻优成功率分别为 100%，96.67%，73.33%，80.00%，而其他 5 种算法最好的收敛成功率为 20.00%. 这说明 AWOA 在求解大规模优化问题时显示出较强寻优能力和鲁棒性.

由表 5 的仿真数据可知，AWOA 对于 5 个固定维测试函数的仿真结果在平均值上均优于其他 5 种算法. 对于测试函数 $f_{10}(x)$，除 IWOA 外其他 5 种算法均能收敛到理论最优值 −1，CWOA 的标准差更小，具有更强的稳定性. 但是，对于测试函数 $f_8(x)$ 和 $f_9(x)$，AWOA 也能收敛到理论最优值，其求解结果明显优于其他 5 种算法.

为了便于观察 AWOA 的寻优性能，图 3 给出了 6 种算法在不同维度下测试函数的仿真曲线. 由图 3 可以看出，相较于其他 5 种算法，本文提出的 AWOA 在收敛精度和收敛速度方面更具有优势.

表 5　6 种算法对 5 个固定维测试函数的寻优性能比较

Table 5　Comparison of the optimal performance of 6 algorithms for 5 fixed-dimensional test functions

函数	指标	WOA	IWOA	WOAWC	CWOA	CPWOA	AWOA
$f_6(x)$	平均值	1.328 7	**0.998 0**	**0.998 0**	2.707 3	**0.998 0**	**0.998 0**
	标准差	0.752 1	1.66E−006	9.15E−010	2.378 7	**1.26E−011**	1.62E−010
$f_7(x)$	平均值	6.05E−004	6.49E−004	3.71E−004	4.47E−004	6.08E−004	**3.39E−004**
	标准差	3.56E−004	3.20E−004	5.82E−005	1.37E−004	2.74E−004	**3.54E−005**
$f_8(x)$	平均值	−9.200 6	−8.998 5	−10.375 4	−10.132 0	−9.703 0	**−10.402 8**
	标准差	2.458 0	2.659 1	0.100 6	1.026 1	2.140 2	**9.57E−005**
$f_9(x)$	平均值	−9.634 7	−8.944 1	−10.515 8	−10.492 8	−9.995 3	**−10.536 3**
	标准差	2.049 8	3.042 3	0.029 0	0.039 0	2.058 7	**3.43E−004**
$f_{10}(x)$	平均值	**−1.000 0**	−0.833 2	**−1.000 0**	**−1.000 0**	**−1.000 0**	**−1.000 0**
	标准差	1.59E−005	0.378 9	1.01E−005	**2.04E−008**	3.66E−008	1.20E−006

注：种群规模 $N=50$，最大迭代次数 $t_{\max}=1\,000$.

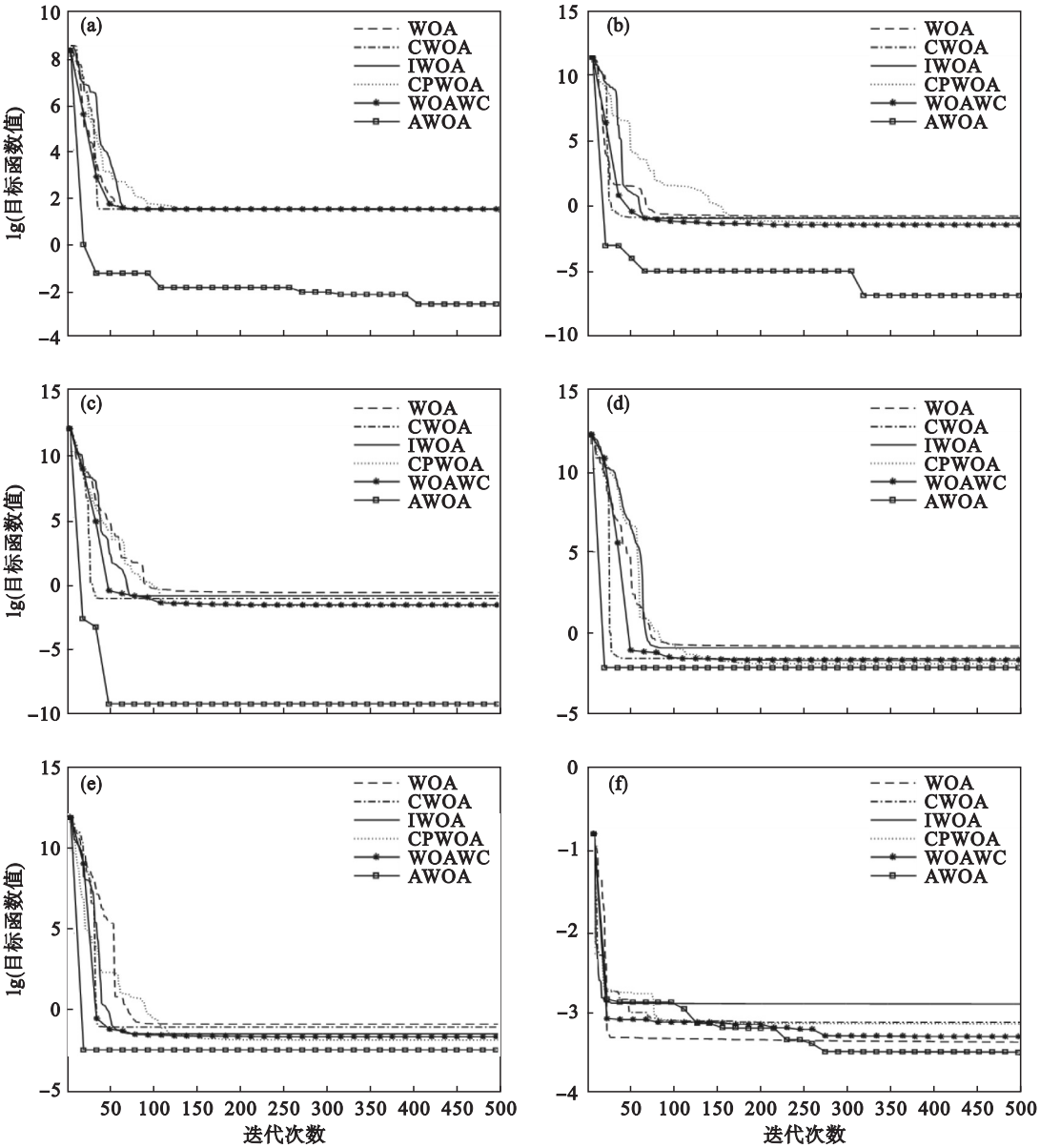

图 3 6 种算法对 3 个基准函数的收敛图

Fig. 3 Convergence graph of 6 algorithms for 3 benchmark functions

(a)—$F_5(x)$(30 维)收敛图;(b)—$f_4(x)$(200 维)收敛图;(c)—$f_4(x)$(500 维)收敛图;(d)—$f_4(x)$(1 000 维)收敛图;(e)—$f_4(x)$(1 500 维)收敛图;(f)—$f_7(x)$(4 维)收敛图.

4 结 语

本文介绍了 WOA 的基本原理,针对其收敛精度低和容易陷入局部最优的缺陷,提出了一种基于自适应调整权重和搜索策略的 AWOA. AWOA 可以根据当前鲸鱼种群的分布情况自适应地改变权值的大小,并且选择不同的搜素策略来更新位置,以平衡算法全局搜索和局部搜索能力.

通过对 WOA 的 13 个标准测试函数、5 个高维测试函数和 5 个固定维测试函数的数值仿真,结果表明,本文提出的 AWOA 在收敛精度和收敛速度方面总体上要优于标准 WOA, WOAWC, IWOA, CWOA 和 CPWOA. 对于求解有约束的实际工程应用问题是下一步研究的主要内容.

参考文献:

[1] Kennedy J, Eberhart R. Particle swarm optimization[C]// Proceeding of the IEEE International Conference on Neural Networks. Perth, 1995: 1942 – 1948.

[2] Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents[J]. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 1996, 26(1): 29 – 41.

[3] Biswas S, Kundu S, Das S. Inducing niching behavior in differential evolution through local information sharing[J]. *IEEE Transactions on Evolutionary Computation*, 2015, 19

(2):246 – 263.
[4] Yang X S. Firefly algorithm, stochastic test functions and design optimization[J]. *International Journal of Bio-Inspired Computation*,2010,2(2):78 – 84.
[5] Yang X S. A new metaheuristic bat-inspired algorithm[C]// Nature Inspired Cooperative Strategies for Optimization. Berlin:Springer,2010:65 – 74.
[6] Mirjalili S,Mirjalili S M,Lewis A. Grey wolf optimizer[J]. *Advances in Engineering Software*,2014,69(7):46 – 61.
[7] Seyedali M, Andrew L. The whale optimization algorithm [J]. *Advances in Engineering Software*,2016,95:51 – 67.
[8] 郭振洲,王平,马云峰,等. 基于自适应权重和柯西变异的鲸鱼优化算法[J]. 微电子学与计算机,2017,34(9):20 – 25.
(Guo Zhen-zhou, Wang Ping, Ma Yun-feng, et al. Whale optimization algorithm based on adaptive weight and Cauchy mutation[J]. *Microelectronics & Computer*, 2017, 34(9): 20 – 25.)
[9] 龙文,蔡绍洪,焦建军,等. 求解大规模优化问题的改进鲸鱼优化算法[J]. 系统工程理论与实践,2017,37(11):2983 – 2994.
(Long Wen, Cai Shao-hong, Jiao Jian-jun, et al. Improved whale optimization algorithm for large scale optimization problem[J]. *System Engineering—Theory & Practice*, 2017, 37(11):2983 – 2994.)
[10] 王坚浩,张亮,史超,等. 基于混沌搜索策略的鲸鱼优化算法[J]. 控制与决策,2019,34(9):1893 – 1900.
(Wang Jian-hao, Zhang Liang, Shi Chao, et al. Whale optimization algorithm based on chaotic search strategy[J]. *Control and Decision*,2019,34(9):1893 – 1900.)
[11] 张永,陈锋. 一种改进的鲸鱼优化算法[J]. 计算工程,2018,44(3):208 – 213.
(Zhang Yong, Chen Feng. A modified whale optimization algorithm[J]. *Computer Engineering*, 2018, 44(3): 208 – 213.)
[12] 黄清宝,李俊兴,宋春宁,等. 基于余弦控制因子和多项式变异的鲸鱼优化算法[EB/OL]. (2018 – 11 – 13)[2018 – 12 – 03]. https://doi.org/10.13195/j.kzyjc.2018.0463.
(Huang Qing-bao, Li Jun-xing, Song Chun-ning, et al. Whale optimization algorithm based on cosine control factor and polynomial mutation[EB/OL]. (2018 – 11 – 13)[2018 – 12 – 03]. https://doi.org/10.13195/j.kzyjc.2018.0463.)
[13] Jangir P, Trivedi I N, Bhesdadiya R H, et al. Training multi-layer perception in neural network using whale optimization algorithm[J]. *Indian Journal of Science and Technology*, 2016,9(19):28 – 36.
[14] Khaled M, Samir S, Abdelghani B. Whale optimization algorithm based optimal reactive power dispatch:a case study of the Algerian power system[J]. *Electric Power Systems Research*,2018,163:696 – 705.
[15] Prakash D B, Lakshminarayana C. Optimal siting of capacitors in radial distribution network using whale optimization algorithm[J]. *Alexandria Engineering Journal*, 2017,56(4):499 – 509.
[16] Wang G G, Suash D, Gao X Z, et al. A new metaheuristic optimisation algorithm motivated by elephant herding behaviour [J]. *International Journal of Bio-Inspired Computation*,2016,8(6):394 – 409.
[17] Mirjalili S. SCA: a sine cosine algorithm for solving optimization problems[J]. *Knowledge-Based Systems*,2016, 96:120 – 133.

(上接第 34 页)

[8] Hyndman R J, Athanasopoulos G. Forecasting:principles and practice[M]. Melbourne:OTexts,2018.
[9] Jaccard P. Etude comparative de la distribution florale dans une portion des Alpes et des Jura[J]. *Bulletin del la Société Vaudoise des Sciences Naturelles*,1901,37:547 – 579.
[10] Adamic L A, Adar E. Friends and neighbors on the web[J]. *Social Networks*,2003,25(3):211 – 230.
[11] Holland P W, Leinhardt S. Transitivity in structural models of small groups[J]. *Social Networks*,1977,2(2):49 – 66.
[12] Cohen J. Applied multiple regression/correlation analysis for the behavioral sciences[M]. Mahwah: Lawrence Erlbaum Associates,2003.
[13] Ricardo P, Soares S, Prudencio R. Time series based link prediction[C]//International Joint Conference on Neural Networks. Brisbane,2012:1 – 7.
[14] Xu M, Yin Y C. A similarity index algorithm for link prediction[C]//2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE). Nanjing,2017:1 – 6.
[15] Zhou T, Lyu L Y, Zhang Y C. Predicting missing links via local information[J]. *European Physical Journal B*,2009,71(4):623 – 630.
[16] Jeh G, Widom J. SimRank: a measure of structural-context similarity[C]//Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining—KDD′02. Edmonton:ACM Press,2002:538 – 543.