小型微型计算机系统
Journal of Chinese Computer Systems
DOI: 10.20009/j.cnki.21-1106/TP.2021-0022
2022年8月 第8期
Vol.43 No.8 2022

融合改进天牛须和正余弦的双重搜索优化算法

姚信威,王佐响,姚 远,黄 伟

(浙江工业大学 计算机科学与技术学院,杭州 310023)
E-mail: xwyao@zjut.edu.cn

摘 要: 针对标准正余弦优化算法在搜索时存在的计算精度低、容易陷入局部最优值等缺点,借鉴天牛须算法运算简单、搜索速度较快的特点,本文提出了一种融合改进天牛须和正余弦的双重搜索优化算法(BAS-SCA).首先,在现有标准正余弦算法基础上,引入动态自适应权重机制来平衡全局搜索和局部搜索,提升收敛速度;其次,提出了一种新的转换参数模型,该参数模型通过结合指数型函数和余弦函数来替换传统的线性衰减函数;最后,为了提升正余弦的搜索精度和速度,同时尽可能跳出局部最优解,提出了改进的天牛须搜索算法,引入动态步长搜索机制将固定步长搜索改为变步长搜索,创新性地将改进的天牛须算法与改进的正余弦算法进行融合实现双重搜索优化,有效避免局部极值问题.实验表明,通过14个标准测试函数验证,所提双重搜索优化算法BAS-SCA相较于其它现有优化算法,具有更高的寻优精度和更快的收敛速度.

关 键 词: 正余弦优化;动态惯性权重;指数衰减函数;天牛须搜索;全局寻优;测试函数
中图分类号: TP18 **文献标识码**: A **文 章 编 号**: 1000-1220(2022)08-1644-09

Dual Search Optimization Algorithm Based on Improved Beetle Antennae Search and Sine Cosine Algorithm

YAO Xin-wei, WANG Zuo-xiang, YAO Yuan, HUANG Wei
(College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract: Aiming at the shortcomings of standard sine and cosine optimization algorithm, such as low calculation accuracy and easy to fall into local optimum value, in this paper, based on the advantages of beetle antennae search algorithm in terms of simple operation and fast speed in search, a dual search optimization algorithm (BAS-SCA) is proposed by integrating the improved sine and cosine optimization algorithm and improved beetle antennae search algorithm. First, dynamic adaptive weight mechanism is adopted to balance global search and local search, as well as improving the convergence speed; Second, a new conversion parameter model is proposed to replace the traditional linear decay function by combining exponential function and cosine function. Finally, in order to obtain better solution in a short time and avoid the local extreme problem, fixed search step in the traditional beetle antennae search algorithm is replaced by designed variable step strategy. The improved beetle antennae search algorithm and the improved sine cosine algorithm are combined to achieve double search optimization innovatively. Experimental results on fourteen standard test functions indicate that the proposed dual search algorithm has higher search accuracy and faster convergence rate comparing with other existing optimization algorithms.

Key words: sine and cosine optimization algorithm; dynamic inertia weight; exponential attenuation function; beetle antennae search; global optimization; test function

1 引 言

随着科学技术的不断发展,生产力的迅猛提升使得优化问题逐渐成为现阶段研究的热点.近年来,不断有新的算法被提出和拓展,如遗传算法(Genetic Algorithm, GA)[1],模拟退火算法(Simulated Annealing Algorithm)[2],粒子群优化算法(Particle Swarm Optimization, PSO)[3],蝙蝠算法(Bat Algorithm, BA)[4],蚁群优化算法(Ant Colony Optimization, ACO)[5],风驱动算法(Wind Driven Optimization, WDO))[6]等等,而这些算法也可以分为两类:基于个体和基于群体的.第1类只生成优化一种解决方案,而第2类会生成多个解决方案并优化.这种特点使得上述算法在科学和工业领域得到广泛关注,尽管现阶段在该领域已经有很多优化算法,但是没有一种算法是能够满足所有的优化问题的[7],所以仍有大量优化算法不断在被提出和改进.

正余弦优化算法(Sine Cosine Algorithm, SCA)[8]是2016年由Seyedali、Mirjalili教授提出的一种群智能优化算法,该算法基于正余弦数学函数设计,算法始于一组随机解,通过目

收稿日期:2021-02-08 收修改稿日期:2021-04-05 基金项目:国家自然科学基金项目(61772471,61906169,61771430)资助;浙江省自然科学基金项目(LQ19F030009)资助. 作者简介:**姚信威**(通讯作者),男,1986年生,博士,副教授,博士生导师,CCF会员,研究方向为群智感知与协同、智联网、智能机器人;**王佐响**,男,1997年生,硕士研究生,研究方向为智能优化算法和机器人路径规划;**姚 远**,男,1990年生,博士,讲师,CCF会员,研究方向为人工智能和智能算法;**黄 伟**,男,1981年生,博士,副教授,CCF会员,研究方向为信号处理及网络科学理论.

标函数反复评估此随机解,并通过作为优化技术核心的一组规则对其进行改进,位置的更替通过线性递减函数即转换参数完成,该算法被验证有效并已经应用到很多现实问题上,如天气多方位分析预测系统[9]、城市动力系统全局调度优化[10]、城市土地优化配置问题[11]、图像分割[12]、结构损伤检测[13]、数据挖掘等许多问题上,但是在其他高维或者更加复杂的情况中,往往会受到一些瓶颈,如陷入局部最优解、计算精度低等.

为了解决传统正余弦优化算法存在的问题,很多研究者把目光朝向了改变转换参数和加入动态调整等方向. 转换参数对整个正余弦优化算法的搜索能力有很大的影响,对正余弦的进一步研究可以了解到,其先是进行全局范围内的搜索,再对局部进行进一步优化分析,所以转换参数应该随着搜索过程而改变,根据这个原理,刘勇等[14]针对转换参数的设置进行分析,提出将传统正余弦算法中的线性递减函数更改为非线性递减函数,采用抛物线函数来设置参数,称为(Parabolic Sine Cosine Algorithm,PSCA); Li Ning 等[15]提出了一种新的正余弦优化算法(Bare bones Sine Cosine Algorithm,BBSCA),该算法结合了高斯搜索方程和指数递减策略来生成个体; Shubham Gupta 等[16]针对 SCA 的停止和局部最优解的情况提出用扰动率产生相反种群并引入自适应权重从而使算法能充分搜索函数区域,其把该算法称为修正正余弦算法(Modified Sine Cosine Algorithm,MSCA); Yetao Ji 等[17]发现 SCA 在算法后期处理复杂问题容易陷入局部最优和惰性收敛,为了让 SCA 算法在全局搜索和局部探索之间的变换更加灵活,提出一种基于自适应参数和混沌开发策略的改进 SCA 来增强算法的局部搜索能力; Wen Long 等[18]发现传统 SCA 以及变体并没有在高维全局优化问题中得到广泛应用,为了解决高维问题,提出了一种引入惯性权重的改进 SCA(Improved Sine Cosine Algorithm,ISCA),同时采用了一种基于高斯函数的非线性转换参数减少策略来平衡 SCA 的两个探索过程; Rizk M 等[19]考虑到 SCA 无法对解决方案的多样性进行保护以及没能采用强调策略引导搜索区域,提出了一种基于正交并行信息的 SCA(Sine Cosine Algorithm via orthogonal parallel information,SCA-OPI),引入多正交并行信息,采用一种基于经验的对立方向策略来保持搜索能力; Turgut[20]结合回溯算法的优点,提出了一种混合正余弦优化算法(Backtracking Search Algorithm & Sine Cosine Algorithm,BSA-SCA)并在此优化算法基础上实现了管壳式蒸发器的优化设计; Chegini 等[21]将正余弦位置更新方程与粒子群优化算法以及 Levy flight(LF)方法结合,提出了新的混合算法,LF 方法是一个随机游动,通过 Levy 分布生成搜索步骤,随着跳转次数的增加,在搜索空间内能进行更加有效的搜索,提高基础正余弦的搜索能力,避免陷入局部极小值; Issa 等[22]通过引入调优参数并结合粒子群优化来改进正余弦算法.

尽管已经有部分学者对 SCA 算法进行改进,如优化参数或结合粒子群等其它算法,但是该算法存在的易陷入局部极小值问题仍无法完全避免,算法收敛精度较低还是需要改善,针对上述问题,本文提出了融合改进天牛须搜索算法(Beetle Antennae Search, BAS) 和正余弦的双重优化算法(BAS-SCA). 天牛须搜索(Beetle Antennae Search,BAS),也叫甲壳虫须搜索,是 Jiang Xiang-yuan[23]在 2017 年提出的一种高效智能优化算法. 与遗传算法、蚁群算法、风驱动等智能优化算法不同,天牛须搜索可以在忽略函数的具体信息如函数的梯度等情况下完成寻优目标,并且天牛须搜索只需要一个个体就能够进行寻优计算,因此天牛须算法的运算量远小于粒子群算法. 但是基础的天牛须搜索步长是固定的,为了更好的改善性能,将天牛须搜索的固定步长改为变步长提升正余弦优化算法搜索性能. 在正余弦中提出了一种新的自适应转换参数模型,改变了传统的线性递减参数设置,此外,引入了一个全新的自适应惯性权重机制来保证算法后期的收敛性,变步长天牛须搜索对改进正余弦算法进行二次寻优,增强了正余弦优化算法的局部搜索能力. 通过 14 个标准测试函数上的仿真实验,与抛物线正余弦优化算法[14]和指数正余弦优化算法[15]进行对比,本文所提算法在寻优精度和寻优准确度上都有较大的性能优势,证明了算法的可行性.

2　正余弦优化算法

标准的正余弦算法的位置更新公式如式(1)所示,正余弦搜索过程可以分为两个过程[24]: 探索和开发,探索过程中,算法以高概率将随机解组合,寻求在搜索空间内最有希望的区域.

$$X_i^{t+1}=\begin{cases}X_i^t+r_1\times\sin(r_2)\times|r_3P_j^t-X_i^t|, r_4<0.5\\ X_i^t+r_1\times\cos(r_2)\times|r_3P_j^t-X_i^t|, r_4\geqslant 0.5\end{cases}\tag{1}$$

其中,t 为当前迭代次数,X_i^t 为在第 t 次迭代下第 i 个粒子在第 j 维度下的最优位置,$r_2/r_3/r_4$ 是 3 个参数,r_2 变化范围在 $[0,2\pi]$,r_3 变化范围在 $[-2,2]$,r_4 变化范围在 $[-1,1]$,P_j^t 为在 t 次迭代下的目标点,r_1 称为转换参数并且为一个线性递减函数,它会随着迭代次数的增加从常数 a 线性递减至 0,它的线性变化公式如式(2)所示:

$$r_1=a-t\frac{a}{T}\tag{2}$$

其中,T 为最大迭代次数,$a>0$.

正弦余弦算法的迭代过程如图 1 所示,图中的 r_1 设为 2,可以看出,当 $r_1\sin(r_2)$ 或者 $r_1\cos(r_2)$ 的值在区间(1,2]或[-2,-1)之内时,SCA 对全局环境进行搜索; 当 $r_1\sin(r_2)$ 或者 $r_1\cos(r_2)$ 的值在区间 $[-1,1]$ 之内时,SCA 对已探索空间进行局部寻优操作. 可以看出,正余弦优化算法的性能好坏很大程度上取决于全局和局部搜索两者能否达到一种动态平衡,全局搜索用于快速定位最优解的范围,而局部解则在该范

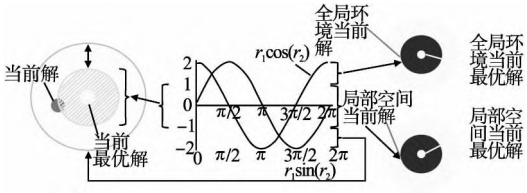


图 1　正余弦算法原理图

Fig. 1　Schematic diagram of sine and cosine algorithm

围内来寻找最优解,全局搜索过多会使得搜索效率低下,而局部搜索过多会使得算法陷入局部最优解的情况.

3 天牛须搜索算法

天牛须算法是一种有效的元启发式算法,其灵感来自于天牛的觅食行为[25],如图 2 所示,将天牛简化模型分为天牛左须和右须,两须之间的距离为 d_0,两须到天牛质心的距离是相等的,天牛须搜索分为两个步骤: 搜索行为和检测更新行为. 天牛在每次迭代中都会根据随机方向进行移动,该算法的实现需要先对搜索方向进行矢量归一化处理,如式(3) 所示:

$$\vec{b} = \frac{\text{rand}(\text{j},1))}{\| \text{rand}(\text{j},1) \|} \tag{3}$$

其中,rand 函数表示一个随机函数,j 代表当前位置的维度.

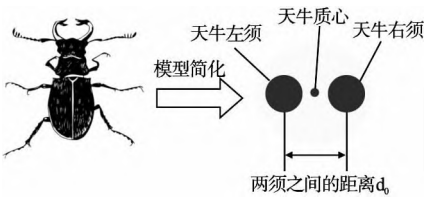


图 2 天牛须搜索原理图

Fig. 2 Schematic diagram of beetle antennae search

用式(4) 初始化天牛须的双向搜索行为:

$$\begin{cases} x_r = x^t + d^t \vec{b} \\ x_l = x^t - d^t \vec{b} \end{cases} \tag{4}$$

其中,x_r 代表的是右向的搜索空间,x_l 代表的是左向的搜索空间,d^t 是天牛须到天牛质心之间的距离,更准确的说,d^t 是某个特定时刻对应的天牛的搜索区域.

天牛须当前位置根据式(5) 的规则进行更新移动:

$$x^{t+1} = x^t + \delta^t \vec{b} sign(f(x_r) - f(x_l)) \tag{5}$$

其中,δ^t 是搜索步长,$f(x)$ 用来计算对应函数的适应度值,sign 函数是用来确定天牛之后的搜索方向,如果右向适应度较大,则 sign 函数取 1,天牛就朝着该方向继续前进,反之,朝相反方向移动,移动步长为 δ^t,在传统算法中,d^t 和 δ^t 一般取 0. 95 随迭代次数做线性缩减. 据上,一次迭代过程中只探索了随机方向上的某一个位置. 同时,根据公式(5) 可知,每一次迭代的目标函数无论是较大还是较小,天牛的位置和步长都会进行更新,固定步长就会使得天牛容易陷入局部最优位置无法跳出.

4 融合改进天牛须和正余弦的优化算法

分析发现,标准的正余弦优化算法在维度较高时容易陷入局部最优解并且收敛精度较低,针对这一问题,引入了全新的自适应惯性权重 w,w 能够随着迭代次数的改变而动态影响算法所寻找的最优位置,最大限度保证算法的收敛性; 另外,考虑到标准正余弦优化算法中线性递减函数的固定递减对算法性能的影响,提出了一种新的自适应转换参数模型,引入指数型函数和余弦函数对正余弦算法进行改进,提高搜索精度; 为了尽可能的降低正余弦算法陷入局部最优解的可能性,创新地将高运算效率的标准天牛须算法进行改进,将标准的固定步长改进为变步长并将其融入改进的正余弦算法中,通过改进天牛须算法的二次搜索寻优来帮助正余弦跳出局部最优解.

4.1 引入自适应权重

受蚁群优化算法[5]的启发,在正余弦优化算法的位置更新公式中引入动态自适应权重 w. 在算法进行探索阶段时,自适应权重能有效减小当前最优个体位置对全局搜索的影响,而随着正余弦算法逐渐进入到开发局部搜索阶段时,自适应权重能随着迭代次数逐步提升最优个体位置的影响力,同时帮助其余个体尽快收敛到最优个体位置. 随着正余弦优化算法迭代次数 t 改变的自适应惯性权重如式(6) 所示:

$$w(t) = 0.2\cos\left(\frac{\pi}{2} \cdot \left(1 - \frac{t}{T}\right)\right) \tag{6}$$

其中,T 为最大迭代次数,t 为当前迭代次数. 在正余弦中引入自适应惯性权重之后,随着迭代次数的改变,每个位置上对下个寻优位置的影响将不再是固定的,而是由一种非线性的规律变化,对该函数分析可以看出,前期在寻找最优值中,权重较低,对下次更新位置影响较小,但是随着迭代次数的增加,权重会逐渐变大,这样之后的迭代位置变化范围就会逐渐变小,能够最大限度保证算法的收敛性.

4.2 递减参数 r_1 的改变

r_1 参数对整个正余弦优化算法的移动方向起着关键作用[26],但是传统的线性递减函数使得正余弦的搜索方式显得很单一,每一次寻找都是按着固定的变值接近最优值,这样很大程度会陷入局部最优解. 为了进一步加强迭代后期局部搜索能力,用更加灵活的搜索方式寻找,引入动态变化的思想,将线性递减函数 r_1 变成指数型递减函数[27],同时引入余弦函数,提出了一种新的转换参数模型,指数型函数由于其自身的单调性,能够保证变化是单向的,余弦函数随着迭代次数的增加,会非线性逐渐减小,反应在指数函数上,r_1 参数就会随着迭代次数的增加非线性减小,从而逐渐逼近最优值,增强正余弦对未知区域的搜索能力,如式(7) 所示:

$$r_1 = \alpha e^{\cos\left(\pi \cdot \frac{t}{T+t}\right)} \tag{7}$$

其中,α 为一个常数,本文中取 0. 05.

引入自适应权重和指数型递减函数之后,正余弦算法位置更新公式如式(8) 所示:

$$X_i^{t+1} = \begin{cases} w(t) \cdot X_i^t + r_1 \times \sin(r_2) \times |r_3 P_j^t - X_i^t|, r_4 < 0.5 \\ w(t) \cdot X_i^t + r_1 \times \cos(r_2) \times |r_3 P_j^t - X_i^t|, r_4 \geqslant 0.5 \end{cases} \tag{8}$$

4.3 变步长搜索机制

为了增强正余弦的搜索能力和效率,改善 SCA 算法易陷入局部最优、迭代效率低、搜索精度低等缺点,在改进正余弦算法中引入了具有高运行效率的天牛须搜索,同时为了规避标准天牛须算法自身参数简单易找不到最优解等情况,对天牛须搜索进行参数优化. 为了使天牛须搜索能够更好的在正余弦上寻优,结合动态变化思想,将其搜索的固定步长改为变步长搜索[28],变化步长为公式(9) :

$$\delta^t = s_1 \times \left(\frac{s_0}{s_1}\right)^{\frac{T}{T+10t}} \tag{9}$$

其中,T 为最大迭代次数,s_0 和 s_1 为常数,本文取 0. 9 和 0. 4,t 为当前迭代次数. 经过天牛须的二次搜索得到的值为 X_*^{t+1}.

改进的天牛须算法对改进正余弦优化算法搜索出的最佳位置进行二次寻优,再在计算适应度值的基础上用贪婪策略判断两次位置最优情况,得到更新的位置.

根据贪婪策略[29]来对搜索的值进行判断,具体公式如式(10)所示:

$$\widetilde{X^{t+1}}=\begin{cases}X_i^{t+1}, f(X_i^{t+1}) \leqslant f(X_*^{t+1})\\ X_*^{t+1}, f(X_*^{t+1}) < f(X_i^{t+1})\end{cases} \tag{10}$$

其中,$f(X)$为在位置 x 处求得的适应度函数,比较原正余弦位置和天牛须二次搜索位置的适应度值,若二次搜索更优则替换原位置[30],反之则不替换.

4.4　算法流程

融合改进天牛须和正余弦的双重搜索优化算法的具体流程如图 3 所示.

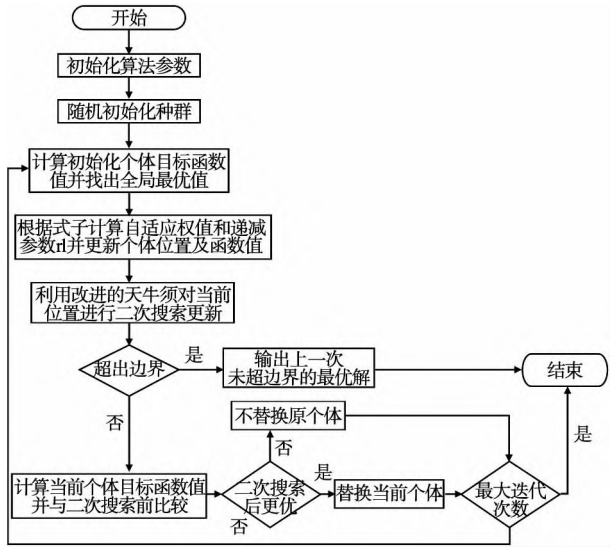


图 3　算法流程图

Fig. 3　Algorithm flow chart

BAS-SCA 算法的实现步骤如下:

步骤 1. 初始化算法参数,包含迭代次数,搜索维度,以及种群规模;

步骤 2. 根据参数的限定范围,随机初始化种群,根据目标函数计算找出全局最优值;

步骤 3. 计算自适应权值和递减参数,通过公式(8)更新最优个体位置及对应的适应度值;

步骤 4. 通过公式(5)和公式(9)对步骤 3 得到的最优个体位置进行二次搜索寻优更新,并计算适应度值;

步骤 5. 判断得到的位置是否超过步骤 2 设定的参数范围,如果超过则输出上一次未超过边界的最优解,;反之,则继续向下进行;

步骤 6. 运用贪婪策略,通过公式(10)判断最终的最优个体位置,如果二次搜索后更优,则替换当前个体,反之不替换;

步骤 7. 判断是否达到 BAS-SCA 的最大迭代次数,若达到,则输出最优个体位置及最优解;反之,继续进入步骤 3 进行.

5　算法性能测试

5.1　测试函数及性能指标

为了测试 BAS-SCA 算法的寻优能力,参考标准正余弦优化算法(Sine Cosine Algorithm,SCA)[8]、粒子群算法(Particle Swarm Optimization,PSO)[3]、基于变换函数与填充函数的模糊粒子群优化算法[31]、基于自适应搜索中心的骨干粒子群算法[32]中所选的测试函数,选择了 14 个选用率较高的基准函数进行测试,函数特性如表 1 所示,基准函数中包含单峰函数和多峰函数,其中,f_1-f_6 以及 f_{11} 为单峰函数,f_7-f_{10} 以及 f_{12}-f_{14} 为多峰函数,单峰函数可以评测算法的收敛精度和寻优能力,多峰函数可以评测算法解决多维环境等复杂优化问题下的寻优能力. 本文选取改进算法与标准 SCA、粒子群算法(Particle Swarm Optimization,PSO),改进的抛物线函数正余弦算法(Parabolic Sine Cosine Algorithm,PSCA)[14]和改进的指数正余弦算法(Exponential Sine Cosine Algorithm,ESCA)[15]作为比较对象进行实验. 本文在正余弦算法的改进思路上,引入了指数函数和余弦函数,为了证明是两个函数的有效结合带来了算法的提升,与改进的抛物线函数正余弦算法和改进的指数正余弦算法进行比对. 另为了对比该算法与其他自然算法的性能,选择了具有代表性的粒子群优化算法作为对比,粒子群优化算法自提出以来,便因其简单易用以及高效性被广泛研究改进,故本文选取了经典粒子群优化算法作为对比算法.

算法寻优精度为实际寻得最优解与理论最优解之间的误差绝对值,本文采用两个评价指标[33]:寻优精度的平均值(Ave)和标准差(Std),两者的计算公式如式(11)和式(12):

$$Ave=\frac{\sum_{i=1}^{t}|f(X^*)-f(X_{opt})|}{T} \tag{11}$$

$$Std=\sqrt{\frac{\sum_{i=1}^{t}(f(X^*)-Ave)^2}{T}} \tag{12}$$

式中,X_{opt} 为理论最优解,X^* 为每次算法得到的最优个体位置,T 为最大迭代次数. 寻优精度的平均值是在指定循环次数下算法得到的值和理论最优值差的绝对值的平均值,平均值越小,说明寻优结果越好;寻优精度标准差是实际最优值与平均值之间的标准差,标准差越小,则说明算法整体的稳定性越好.

在选取的 14 个基准测试函数中,函数的搜索区间及全局最优值已在表 1 中列出. 几种算法的参数设置相同:个体规模设置 Agents 为 30,最大迭代次数为 500,每种算法都独立运行 50 次,并且考虑到高维的测试环境,这里设置了 Dimension = 30 和 200 两种情况分别测试,所有的测试在 Windows10 操作系统,Intel i5-10400,16GB 内存,MATLAB2020a 仿真平台上运行,通过统计得出优化结果的平均值和标准差如表 2 所示.

5.2　优化算法的性能分析

表 2 给出了 5 种算法在 14 个标准测试函数下的寻优指标对比,分析单峰函数可以看出,在维度为 30 的情况下,算法寻优精度平均值和标准差上,标准正余弦算法都陷入了局部最优解的情况,PSCA 和 ESCA 虽然在结果上比 SCA 有一定程度的改善,但是两者精度还是不高,对比之下,BAS-SCA 算法在单峰函数寻优上都好于实验的其它算法. BAS-SCA 算法在 f_1-f_4 函数中直接找到了全局最优解 0,精度达到 100%;在函数 f_5 和 f_{11} 中,虽然没能找到全局最优解,但是其寻优精度也是对比算法中最高的;在函数 f_6 中,BAS-SCA 算法虽然没能比 PSO 算法好,但是相较于其他改进的 SCA 算法,其仍具有优势.

表 1 标准测试函数

Table 1 Benchmark test function

函数表达式	搜索区间	最优值
$f_1(x) = \sum_{i=1}^{t} x_i^2$	$[-100,100]$	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10,10]$	0
$f_3(x) = \sum_{i=1}^{t} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100,100]$	0
$f_4(x) = max\{ \lvert x_i \rvert, 1 \leqslant i \leqslant t \}$	$[-100,100]$	0
$f_5(x) = \sum_{i=1}^{t-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-100,100]$	0
$f_6(x) = \sum_{i=1}^{t} (\lvert x_i + 0.5 \rvert)^2$	$[-100,100]$	0
$f_7(x) = \sum_{i=1}^{t} i x_i^4 + random(0,1)$	$[-1.28,1.28]$	0
$f_8(x) = \sum_{i=1}^{t} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12,5.12]$	0
$f_9(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{t} \sum_{i=1}^{t} x_i^2} \right) - \exp\left(\frac{1}{t} \sum_{i=1}^{t} \cos(2\pi x_i) \right) + 20 + e$	$[-32,32]$	0
$f_{10}(x) = \frac{1}{4000} \sum_{i=1}^{t} x_i^2 - \prod_{i=1}^{t} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$	$[-600,600]$	0
$f_{11}(x) = \sum_{i=1}^{t} -x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-500,500]$	$-418.98 * dim$
$f_{12}(x) = \frac{\pi}{t} \{ 10 \sin(\pi y_1) + \sum_{i=1}^{t-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})] + [(y_t - 1)^2 \} + \sum_{i=1}^{t} u(x_i, 10, 100, 4)$	$[-50,50]$	0
$f_{13}(x) = 0.1\sin^2(3\pi x_1) + \sum_{i=1}^{t} (x_i - 1)^2 [1 + \sin^2(3\pi x_1 + 1)] + (x_t - 1)^2 [1 + \sin^2(2\pi x_t)] + \sum_{i=1}^{t} u(x_i, 5, 100, 4)$	$[-50,50]$	0
$f_{14}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$	$[-5,5]$	-1.0316

表 2 算法寻优指标对比

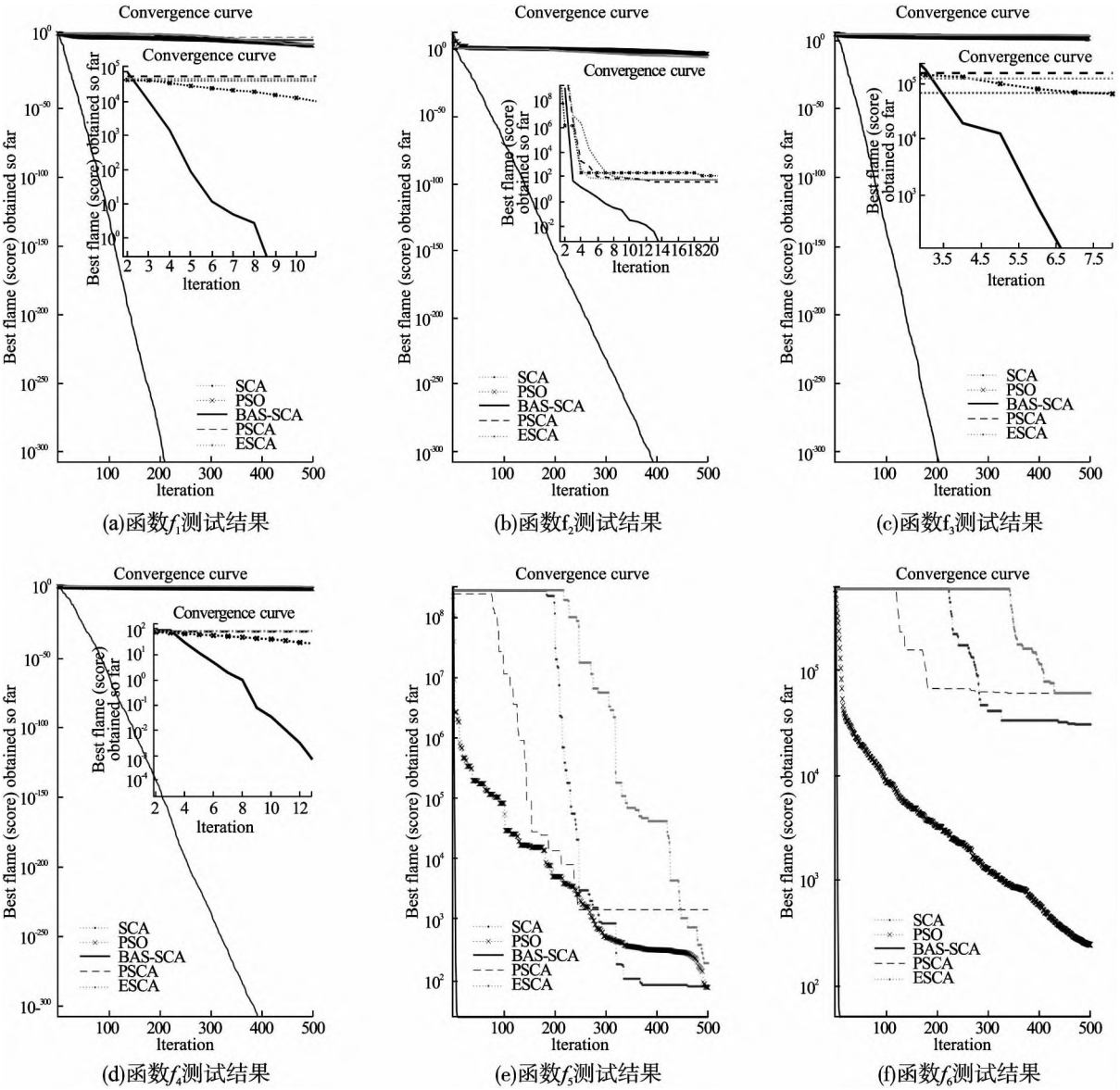
Table 2 Comparison of optimization indexes

函数	Dim	SCA		PSO		BAS-SCA		PSCA		ESCA	
		Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std
f_1	30	1.07E-002	7.57E-002	0.00E+000	1.00E-004	**0.00E+000**	**0.00E+000**	6.42E-002	4.54E-001	2.00E-004	1.50E-003
	200	6.51E+002	4.60E+003	7.10E+000	5.05E+001	**0.00E+000**	**0.00E+000**	8.53E+002	6.03E+002	6.89E+002	4.87E+004
f_2	30	3.00E-004	2.40E-003	5.00E-004	3.70E-003	**0.00E+000**	**0.00E+000**	1.60E-003	1.16E-002	**0.00E+000**	1.00E-004
	200	1.21E+000	8.55E+000	6.18E+000	4.37E+001	**0.00E+000**	**0.00E+000**	3.79E-001	2.68E+000	3.42E-001	2.42E+000
f_3	30	5.52E+001	3.91E+002	2.30E+000	1.66E+001	**0.00E+000**	**0.00E+000**	1.07E+002	7.57E+002	2.87E+002	2.03E+003
	200	2.33E+004	1.65E+005	2.39E+003	1.69E+004	**0.00E+000**	**0.00E+000**	1.67E+004	1.18E+005	3.04E+004	2.15E+005
f_4	30	7.90E-001	5.59E+000	1.49E-002	1.05E-001	**0.00E+000**	**0.00E+000**	8.55E-001	6.05E+000	1.09E+000	7.77E+000
	200	1.93E+000	1.37E+001	3.89E-001	2.75E+000	**0.00E+000**	**0.00E+000**	1.87E+000	1.33E+001	1.96E+000	1.39E+001
f_5	30	1.15E+002	8.11E+002	4.00E+000	2.86E+001	**6.00E-001**	**4.10E+000**	7.67E+002	5.42E+003	2.10E+000	1.50E+001
	200	9.92E+006	7.02E+007	1.00E+004	8.00E+004	**3.98E+000**	**2.81E+001**	3.92E+006	2.77E+007	3.52E+007	2.49E+008
f_6	30	2.45E-001	1.73E+000	**0.00E+000**	**1.00E-004**	1.50E-001	1.06E+000	3.58E-001	2.53E+000	1.20E-001	8.49E-001
	200	7.89E+002	5.58E+003	6.10E+000	4.32E+001	**1.00E+000**	**7.10E+000**	7.67E+002	5.43E+003	8.64E+002	6.11E+003
f_7	30	6.00E-004	4.60E-003	4.50E-003	3.20E-002	**7.18E-007**	**5.07E-006**	1.50E-003	1.09E-002	2.20E-003	1.55E-002
	200	3.97E+001	2.80E+002	1.47E+002	1.04E+003	**0.00E+000**	**1.00E-001**	4.17E+001	2.95E+002	4.39E+001	3.11E+002
f_8	30	5.73E-001	4.05E+000	1.00E+000	7.08E+000	**0.00E+000**	**0.00E+000**	7.21E-002	5.10E-001	1.64E-001	1.16E+000
	200	1.96E+001	1.39E+002	3.80E+001	2.68E+002	**0.00E+000**	**0.00E+000**	6.87E+000	4.86E+001	8.04E+000	5.68E+001
f_9	30	4.07E-001	2.88E+000	1.00E-004	1.00E-003	**0.00E+000**	**0.00E+000**	1.56E-002	1.10E-001	2.52E-001	1.78E+000
	200	4.13E-001	2.92E+000	1.40E-001	9.88E-001	**0.00E+000**	**0.00E+000**	4.14E-001	2.92E+000	4.14E-001	2.93E+000
f_{10}	30	2.16E-002	1.53E-001	5.00E-004	3.80E-003	**0.00E+000**	**0.00E+000**	3.41E-002	2.41E-001	1.93E-002	1.36E-001
	200	2.57E+000	1.81E+001	3.54E-002	2.50E-001	**0.00E+000**	**0.00E+000**	2.72E+000	1.93E+001	1.70E+001	1.20E+002
f_{11}	30	7.14E+001	5.05E+002	1.35E+002	9.57E+002	**4.27E+001**	**3.17E+002**	8.57E+001	5.99E+002	8.68E+001	6.14E+002
	200	7.05E+001	5.09E+002	6.06E+001	4.30E+002	**3.68E+001**	**3.64E+002**	7.21E+001	4.99E+002	7.02E+001	4.99E+002
f_{12}	30	0.00E+000	3.00E+000	**0.00E+000**	**0.00E+000**	**0.00E+000**	**0.00E+000**	1.91E+003	1.35E+004	**0.00E+000**	**0.00E+000**
	200	4.80E+002	3.41E+003	**0.00E+000**	**0.00E+000**	**0.00E+000**	**0.00E+000**	**0.00E+000**	**0.00E+000**	4.63E+004	3.27E+005
f_{13}	30	0.00E+000	1.00E+001	**0.00E+000**	**0.00E+000**	**0.00E+000**	**0.00E+000**	1.86E+004	1.31E+005	1.35E+004	9.57E+004
	200	5.61E+004	3.97E+005	**0.00E+000**	**0.00E+000**	**0.00E+000**	**0.00E+000**	1.00E+000	2.00E+001	**0.00E+000**	**0.00E+000**
f_{14}	30	1.64E+000	1.16E+001	2.77E+000	1.96E+001	**7.91E-001**	**5.52E+000**	1.61E+000	1.14E+001	1.57E+000	1.11E+001
	200	5.89E+000	3.82E+001	3.46E+001	2.45E+002	**1.05E+000**	**1.14E+001**	4.91E+000	3.47E+001	4.56E+000	3.66E+001

通过对比观察测试函数f_7-f_{10}在表格中的结果,在多峰函数中,在寻优精度和寻优稳定性上,BAS-SCA 算法都远远好于其他算法,并且在f_8-f_{10}中直接找到了全局最优解,这是其他对比算法都没有达到的,他们都不同程度的陷入局部最优解而得到较差的收敛精度,虽然 BAS-SCA 算法在多峰函数f_7 中没有找到全局最优解,但是其收敛精度也远远高于其他算法. 在测试函数f_{12}和f_{13}中,PSO 算法和 BAS-SCA 算法都能够找到全局最优解,并且有较好的稳定性; 在测试函数f_{14}中,BAS-SCA 也能实现较高的寻优精度及稳定性.

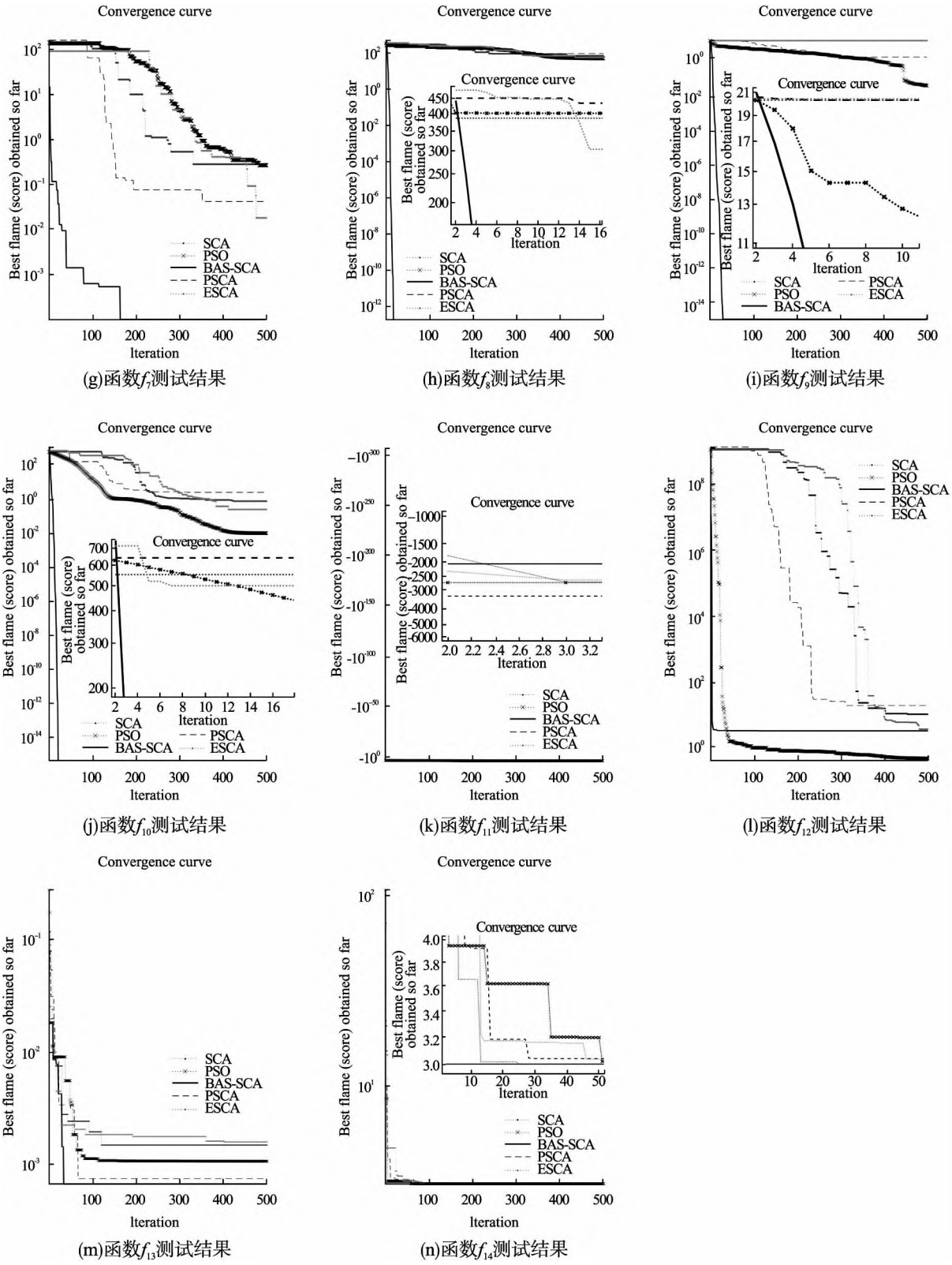
为了进一步测试 BAS-SCA 算法的高维寻优性能,将搜索维度改为 200,其他参数没有改变,对算法重新进行实验分析,从表 2 中可以看出,在维度增加的情况下,BAS-SCA 仍能保持很好的搜索性能,单峰函数f_1-f_4,多峰函数f_7-f_{10}、f_{12}和f_{13}中,都能找到全局最优解. 在函数f_6 中,增加维度情况下,BAS-SCA 算法超过了在 30 维度下性能优于自己的 PSO 算法,得到了较高的收敛精度. 对于函数f_{12}和f_{13},虽然增加维度的情况下指数型正余弦优化算法和抛物线型正余弦优化算法都能够在不同程度上找到全局最优解,但是根据表 2 数据,BAS-SCA 的结果还是优于两者的.

为了能够更好的反应不同算法在不同测试函数下的收敛速度,在图 4 中给出了函数f_1-f_{14}在指定维度为 30 下的收敛对比图,从整体上看,BAS-SCA 算法无论在收敛精度和收敛稳定性上都能大大超过本次比较的现有算法,尤其是图 4(e) 、图 4(h) 、图 4(i) 、图 4(j) ,BAS-SCA 算法在一开始就以很快的收敛速度找到最优解,如图 4(b) 、图 4(g) 、图 4(f) 所示,可以发现,虽然开始标准 SCA 也有很快的收敛速度,但是其很快陷入了局部最优解并无法跳出,其余算法已经陷入了局部最优解,而得益于BAS-SCA算法的二次搜索,BAS-SCA在这几个函数中表现很好,直接跳出了局部最优,找到了全局最优解,图 4(a) 、图 4(c) 、图 4(d) 、图 4(e) 所示,虽然其余算法并没有陷入局部最优,但是其收敛速度和收敛精度是完全比不上 BAS-SCA 的,BAS-SCA 算法在运行一开始就展示出极好的收敛性能,在很短的迭代次数中就找到了全局最优解. 观察图 4(l) ,虽然 BAS-SCA 在收敛精度上略差于 PSO 算法,但是根据表 2 的实验结果,BAS-SCA 仍可以找到最优解,图 4(m) 、图 4(n) 也可看出 BAS-SCA 能得到优于其他算法的结果.



(a)函数f_1测试结果　(b)函数f_2测试结果　(c)函数f_3测试结果

(d)函数f_4测试结果　(e)函数f_5测试结果　(f)函数f_6测试结果



(g)函数f_7测试结果 (h)函数f_8测试结果 (i)函数f_9测试结果

(j)函数f_{10}测试结果 (k)函数f_{11}测试结果 (l)函数f_{12}测试结果

(m)函数f_{13}测试结果 (n)函数f_{14}测试结果

图 4 具体的测试函数收敛对比图

Fig. 4 Comparison chart of convergence of test function

为了更好的验证改进算法的有效性,增加改进变步长搜索的天牛须优化算法以及引入自适应转换参数正余弦优化算法自比实验,维度选取为 30,最大迭代次数为 500 次,其它参数与原实验设置相同,实验结果如表 3 所示.

根据表 3 可知,改进的变步长搜索天牛须优化算法在给定的测试函数中效果表现较差,BAS-SCA 优化算法在寻优精度

及寻优稳定性上都比引入自适应转换参数的正余弦优化算法　　和改进的变步长搜索天牛须优化算法好.

表 3　改进优化算法自比实验

Table 3　Self comparison experiment of improved optimization algorithm

函数	自适应转换正余弦优化算法		变步长搜索天牛须优化算法		BAS-SCA	
	Ave	Std	Ave	Std	Ave	Std
f_1	3.00E +000	2.13E +001	1.41E +003	9.95E +003	**0.00E +000**	**0.00E +000**
f_2	**0.00E +000**	**0.00E +000**	3.21E +004	2.27E +005	**0.00E +000**	**0.00E +000**
f_3	2.60E +001	1.81E +002	2.44E +003	1.73E +004	**0.00E +000**	**0.00E +000**
f_4	6.16E -001	4.35E +000	1.67E +000	1.18E +001	**0.00E +000**	**0.00E +000**
f_5	1.30E +001	2.00E +002	2.62E +003	1.86E +004	**6.00E -001**	**4.10E +000**
f_6	8.00E -001	5.80E +000	1.40E +003	9.89E +003	**1.50E -001**	**1.06E +000**
f_7	4.00E -003	3.10E -002	1.84E +000	1.30E +001	**7.18E -007**	**5.07E -006**
f_8	2.88E -001	2.04E +000	8.46E +000	5.98E +001	**0.00E +000**	**0.00E +000**
f_9	3.19E -002	2.26E -001	4.11E -001	2.90E +000	**0.00E +000**	**0.00E +000**
f_{10}	2.50E -002	1.77E -001	1.27E +001	8.96E +001	**0.00E +000**	**0.00E +000**
f_{11}	5.99E +001	4.24E +002	2.46E +001	1.74E +002	**4.27E +001**	**3.17E +002**
f_{12}	6.28E +001	4.44E +002	4.14E +001	2.93E +002	**3.68E +001**	**3.64E +002**
f_{13}	**0.00E +000**	**0.00E +000**	2.49E +005	1.76E +005	**0.00E +000**	**0.00E +000**
f_{14}	1.39E +000	9.83E +000	8.84E -001	6.25E +000	**0.00E +000**	**0.00E +000**

6　总　结

针对标准正余弦优化算法的搜索性能问题,本文创新性的将改进的天牛须搜索算法融合进改进的正余弦算法,提出了双重搜索优化算法 BAS-SCA. 该算法在正余弦中提出了一种新的自适应转换参数,改变了传统的线性递减参数设置,此外在正余弦中引入了一个全新的自适应惯性权重来保证算法后期的收敛性,此外,引入动态变化思想,将标准天牛须的固定步长改进为变步长搜索,改进的天牛须搜索再对改进正余弦进行二次搜索寻优,运用贪婪策略判断最终位置信息,测试函数实验以及对改进算法的自比实验证明,BAS-SCA 算法能够很好的解决正余弦和天牛须两者后期易陷入局部最优解以及收敛精度较低等缺点,拥有很强的处理单峰以及多峰函数的能力,同时具备很好的寻优稳定性.

References:

[1] Whitley D. A genetic algorithm tutorial [J]. Statistics and Computing,1994,4(2):65-85.

[2] Van Laarhoven P J M,Aarts E H L. Simulated annealing [M]. Simulated Annealing: Theory and Applications, Springer, Dordrecht, 1987:7-15.

[3] Kennedy J,Eberhart R. Particle swarm optimization [C]//Proceedings of ICNN´95-International Conference on Neural Networks, IEEE,1995,4:1942-1948.

[4] Yang X S,Gandomi A H. Bat algorithm: a novel approach for global engineering optimization [J]. Engineering Optimization, 2012, 29(5):464-483.

[5] Dorigo M,Birattari M,Stutzle T. Ant colony optimization [J]. IEEE Computational Intelligence Magazine,2006,1(4):28-39.

[6] Bayraktar Z,Komurcu M,Werner D H. Wind driven optimization (WDO): a novel nature-inspired optimization algorithm and its application to electromagnet-tics [C]//2010 IEEE Antennas and Propagation Society International Symposium,IEEE,2010:1-4.

[7] Xu Li-feng,Huang Zu-sheng,Yang Zhong-zhu,et al. Mixed particle swarm optimization algorithm with multistage disturbances [J]. Journal of Software,2019,30(6):1835-1852.

[8] Mirjalili S. SCA: a sine cosine algorithm for solving optimization problems [J]. Knowledge-based Systems,2016,96:120-133,doi:10.1016/j.knosys.2015.12.022.

[9] Attia A F,El Sehiemy R A,Hasanien H M. Optimalpower flow solution in power systems using a novel sine-cosine algorithm [J]. International Journal of Electrical Power & Energy Systems,2018,99:331-343,doi:10.1016/j.ijepes.2018.01.024.

[10] Wang J,Yang W,Du P,et al. A novel hybrid forecasting system of wind speed based on a newly developed multi-objective sine cosine algorithm [J]. Energy Conversion and Management,2018,163:134-150,doi:10.1016/j.enconman.2018.02.012.

[11] Rizk-Allah R M. Hybridizing sine cosine algorithm with multi-orthogonal search strategy for engineering design problems [J]. Journal of Computational Design and Engineering,2018,5(2):249-273.

[12] Oliva D,Hinojosa S,Abd Elaziz M,et al. Context based image segmentation using antlion optimization and sine cosine algorithm [J]. Multimedia Tools and Applications,2018,77(19):25761-25797.

[13] Bureerat S,Pholdee N. Adaptive sine cosine algorithm integrated with differential evolution for structural damage detection [C]//International Conference on Computational Science and Its Applications,Springer,Cham,2017:71-86.

[14] Liu Yong,Ma Liang. Sine cosine algorithm for nonlinear conversion transformation parameters [J]. Computer Engineering and Application,2017,53(2):1-5 +46.

[15] Li N,Wang L. Bare-bones based sine cosine algorithm for global optimization [J]. Journal of Computational Science,2020,47:101219,doi:10.1016/j.jocs.2020.101219.

[16] Gupta S,Deep K. A hybrid self-adaptive sine cosine algorithm with opposition based learning [J]. Expert Systems with Applications, 2019,119:210-230,doi:10.1016/j.eswa.2018.10.050.

[17] Ji Y,Tu J,Zhou H,et al. An adaptive chaotic sinecosine algorithm for constrained and unconstrained optimization [J]. Complexity, 2020,2020:1-36,doi:10.1155/2020/6084917.

[18] Long W, Wu T, Liang X, et al. Solving high-dimensional global optimization problems using an improvedsine cosine algorithm [J]. Expert Systems with Applications, 2019, 123: 108-126, doi: 10.1016/j.eswa.2018.11.032.

[19] Rizk-Allah R M. An improved sine-cosine algorithm based on orthogonal parallel information for global optimization [J]. Soft Computing, 2019, 23(16): 7135-7161.

[20] Turgut O E. Thermal and economical optimization of a shell and tube evaporator using hybrid backtrackingsearch-sine-cosine algorithm [J]. Arabian Journal for Science and Engineering, 2017, 42(5): 2105-2123.

[21] Chegini S N, Bagheri A, Najafi F. PSOSCALF: a new hybrid PSO based on sine cosine algorithm and levy flight for solving optimization problems [J]. Applied Soft Computing, 2018, 73: 697-726, doi: 10.1016/j.asoc.2018.09.019.

[22] Issa M, Hassanien A E, Oliva D, et al. ASCA-PSO: adaptive sine cosine optimization algorithm integratedwith particle swarm for pairwise local sequence alig-nment [J]. Expert Systems with Applications, 2018, 99: 56-70, doi: 10.1016/j.eswa.2018.01.019.

[23] Zhu Z, Zhang Z, Man W, et al. A new beetle antennae search algorithm for multi-objective energy management in microgrid [C]// 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), IEEE, 2018: 1599-1603.

[24] Wang Wan-liang, Li Wei-kun, Wang Yu-le, et al. An improved multi-objective sine cosine optimiz-ation algorithm [J]. Journal of Chinese Computer Systems, 2019, 40(10): 2102-2108.

[25] Zhao Yu-qiang, Qian Qian, Zhou Tian-jiang, et al. Hybrid algorithm of search and genetic algorithm for longicorn [J]. Journal of Chinese Computer Systems, 2020, 41(7): 1438-1445.

[26] Abd Elaziz M, Oliva D, Xiong S. An improved opposition-based sine cosine algorithm for global optimization [J]. Expert Systems with Applications, 2017, 90: 484-500, doi: 10.1016/j.eswa.2017.07.043.

[27] Gupta S, Deep K. Improved sine cosine algorithm with crossover scheme for global optimization [J]. Knowledge-Based Systems, 2019, 165: 374-406, doi: 10.1016/j.knosys.2018.12.008.

[28] Cheng Y, Li C, Li S, et al. Motion planning of redundant manipulator with variable joint velocity limit based on beetle antennae search algorithm [J]. IEEE Access, 2020, 8: 138788-138799, doi: 10.1109/ACCESS.2020.3012564.

[29] Sun Y, Zhang J, Li G, et al. Optimized neural network using beetle antennae search for predicting the unconfined compressive strength of jet grouting coalcretes [J]. International Journal for Numerical and Analytical Methods in Geomechanics, 2019, 43(4): 801-813.

[30] Hamdia K M, Zhuang X, Rabczuk T. An efficient optimization approach for designing machine learning models based on genetic algorithm [J]. Neural Computing and Applications, 2021, 33(6): 1923-1933.

[31] Lv Bai-quan, Zhang Jing-jing, Li Zhan-pei, et al. Fuzzy particle swarm optimization based on filled function and transformation function [J]. Acta Automatica Sinica, 2018, 44(1): 74-86.

[32] Wang Dong-feng, Meng Li, Zhao Wen-jie. Improved bare bones particle swarm optimization with adaptive search center [J]. Chinese Journal of Computers, 2016, 39(12): 2652-2667.

[33] Faramarzi A, Heidarinejad M, Stephens B, et al. Equilibrium optimizer: a novel optimization algorithm [J]. Knowledge-Based Systems, 2020, 191: 105190, doi: 10.1016/j.knosys.2019.105190.

附中文参考文献:

[7] 徐利锋, 黄祖胜, 杨中柱, 等. 引入多级扰动的混合型粒子群优化算法 [J]. 软件学报, 2019, 30(6): 1835-1852.

[24] 王万良, 李伟琨, 王宇乐, 等. 一种改进的多目标正余弦优化算法 [J]. 小型微型计算机系统, 2019, 40(10): 2102-2108.

[25] 赵玉强, 钱 谦, 周田江, 等. 天牛须搜索与遗传的混合算法 [J]. 小型微型计算机系统, 2020, 41(7): 1438-1445.

[31] 吕柏权, 张静静, 李占培, 等. 基于变换函数与填充函数的模糊粒子群优化算法 [J]. 自动化学报, 2018, 44(1): 74-86.

[32] 王东风, 孟 丽, 赵文杰. 基于自适应搜索中心的骨干粒子群算法 [J]. 计算机学报, 2016, 39(12): 2652-2667.