◆算法研究与应用◆

融合正弦余弦算法的蝴蝶优化算法*

郑洪清[1],冯文健[2**],周永权[3]

(1.广西外国语学院信息工程学院,广西南宁 530222;2.柳州铁道职业技术学院,广西柳州 545616;3.广西民族大学人工智能学院,广西南宁 530006)

摘要:针对蝴蝶优化算法存在收敛速度慢、求解精度差和易陷入局部最优等缺陷,提出一种融合正弦余弦算法的蝴蝶优化算法。首先在蝴蝶自身认知部分引入非线性自适应因子,其次重新定义香味浓度计算公式,最后在局部搜索阶段引入改进的正弦余弦算法。通过19个基准函数的测试,实验结果表明,本算法在收敛速度、寻优精度和鲁棒性方面均优于蝴蝶优化算法(Butterfly Optimization Algorithm,BOA)、正弦余弦算法(Sine Cosine Algorithm,SCA)、樽海鞘群算法(Salp Swarm Algorithm,SSA)、狼群算法(Grey Wolf Optimizer,GWO)和布谷鸟搜索算法(Cuckoo Search Algorithm,CS),与其他改进蝴蝶优化算法相比,在寻优精度方面也具有一定优势。

关键词:正弦余弦算法 函数优化 蝴蝶优化算法 收敛因子 自适应

中图分类号:TP3 **文献标识码**:A **文章编号**:1005-9164(2021)02-0152-08

DOI:10.13656/j.cnki.gxkx.20210610.006

0 引言

由于群智能算法无需先验知识,无需分析数据内部规律和内在关联,只需对数据本身进行学习,自组织、自适应地完成优化问题的求解,近年来大批群智能算法被相继提出,如布谷鸟搜索算法(Cuckoo Search Algorithm,CS)[1]、樽海鞘群算法(Salp Swarm Algorithm,SSA)[2]、狼群算法(Grey Wolf Optimizer,GWO)[3]、正弦余弦算法(Sine Cosine Algorithm,SCA)[4]、鲸鱼优化算法(Whale Optimization Algorithm)[5]、花朵授粉算法(Flower Pollenation Algorithm,FPA)[6]等。这些群智能算法基本上是模拟生物的群体行为,按照某种合作方式一起求解优化问题,并在工程优化、图像处理、特征选择和机器学习等领域得到广泛应用。蝴蝶优化算法(Butterfly Optimization Algorithm,BOA)[7]是模拟自然界的蝴蝶觅食或求偶行为而衍生出的一种仿生群智能算法,已成功解决图像分割[8]、经济负荷调度[9]等问题。但基本BOA也存在收敛速度慢、计算精度差

*国家自然科学基金项目(61463007)和多数据源教学资源库跨库检索关键技术研究项目(2021KY1386)资助。

【作者简介】

郑洪清(1978-),男,副教授,主要从事计算智能研究,E-mail:zhq7972@sina.com。

【**通信作者】

冯文健(1982-),男,高级工程师,主要从事计算机应用研究,E-mail:165600846@qq.com。

【引用本文】

郑洪清,冯文健,周永权.融合正弦余弦算法的蝴蝶优化算法[J].广西科学,2021,28(2):152-159.

ZHENG H Q,FENG W J,ZHOU Y Q. Butterfly Optimization Algorithm Based on Sine Cosine Algorithm [J]. Guangxi Sciences,2021,28(2):152-159.

和易陷入局部最优等缺陷,许多学者提出不同改进策略。如高文欣等[10]首先引入 limit 阈值来限定 BOA 陷入局部最优次数,然后结合单纯形法和正弦余弦算法提升算法性能,但同时也增加了算法时间复杂度且其函数 4 求解精度较低。王依柔等[11]首先在自身认知部分引入自适应惯性权重,其次在全局最优位置引入扰动策略,在花蜜位置引入疯狂因子来平衡算法的局部与全局搜索能力,提出无限折叠迭代混沌映射的蝴蝶优化算法(SIBOA),但该优化后的算法其精度还有待于进一步提升。宁杰琼等[12]首先利用 Circle 映射初始化种群,然后在局部搜索阶段利用动态切换概率,控制改进正弦余弦算法与蝴蝶优化算法的转换,并在全局和局部位置引入自适应权重系数和逐维变异策略,从而改进算法的性能,使求解的精度较高。但该算法的测试函数较少且回避了最难求解的 Rosenbrock 函数测试。高文欣等[13]在全局位置引入柯西分布函数,在局部位置引入自适应权重因子,并引入动态切换概率来改进算法性能,但函数 Apline 未达到理论最优。高文欣等[14]引入收敛因子和黄金正弦指引机制来提升算法效果,但测试函数较少,精度还有待于进一步提升。谢聪等[15]融入差分进化策略和精英策略改进算法,但算法的串联操作相应地增加了算法的时间开销。

虽然上述算法在一定程度上提高了算法性能,但仍有提升空间。本文提出一种融合正弦余弦算法的蝴蝶优化算法(Sine Cosine Algorithm and Butterfly Optimization Algorithm,SCABOA),在 BOA 基础之上从以下 4 个方面进行改进:在蝴蝶自身认知部分引入非线性自适应因子;重新定义香味浓度计算公式;在局部搜索阶段引入改进的正弦余弦算法;为回避转换概率对算法的影响,将种群一分为二。

1 蝴蝶优化算法

蝴蝶优化算法是模拟蝴蝶觅食和繁衍行为而衍生出的一种仿生群智能算法,在 BOA 中每只蝴蝶都会散发出香味,蝴蝶个体能感知香味而互相吸引,香味浓度的计算公式如下:

$$f_i = cI^a, \tag{1}$$

其中,f_i 表示蝴蝶个体 i 感知香味的强度;c 表示感知模态参数;I 表示蝴蝶散发香味的刺激强度,其值由目标函数值决定;a 表示香味吸收程度,$a \in [0,1]$。

当蝴蝶能感知到周围有更浓香味时便朝其聚集,此阶段称为全局搜索,其计算公式如下:

$$x_i^{t+1} = x_i^t + (r^2 \times g^* - x_i^t) \times f_i。 \tag{2}$$

当蝴蝶未能感知周围香味时便随机移动,此阶段称为局部搜索,其计算公式如下:

$$x_i^{t+1} = x_i^t + (r^2 \times x_j^t - x_k^t) \times f_i。 \tag{3}$$

式(2)和式(3)中的 x_i^t 表示蝴蝶个体 i 在第 t 代的位置,x_j^t 表示蝴蝶个体 j 在第 t 代的位置,x_k^t 表示蝴蝶个体 k 在第 t 代的位置,$r \in [0,1]$,g^* 表示全局最优解。

2 融合正弦余弦算法的蝴蝶优化算法

2.1 非线性自适应因子

大量文献记载了引入自适应权重因子的优势,有线性的,也有非线性的,其主要目的是为了更好地平衡算法的全局搜索和局部搜索能力。这也是算法改进的一个突破口,本文首先在自身认知部分引入非线性自适应因子,使算法迭代初期在较宽范围内搜索,迭代后期在较小范围内搜索。非线性自适应因子的计算公式如下:

$$\omega(t) = 2 \times \exp(-(4 \times t/Ngen)^2), \tag{4}$$

其中,exp 为指数函数,$Ngen$ 为最大迭代次数。

2.2 更改香味浓度计算表达式

在 BOA 中,香味浓度的计算公式如式(1)所示,其值取决于感知模态参数 c、香味吸收程度 a 和蝴蝶散发香味的刺激强度 I,其中 I 值是由目标函数值决定,首先观察 Sphere 函数香味浓度变化趋势(其他函数变化曲线类似),如图 1 所示。

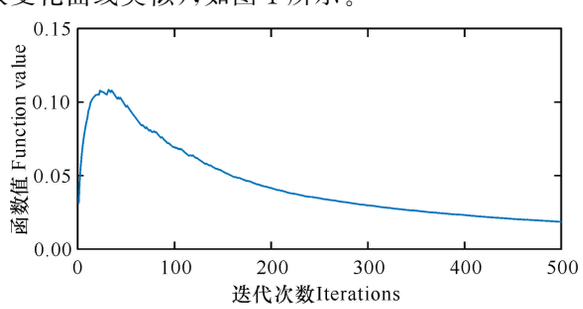

图 1 香味浓度 f_i 变化曲线

Fig 1 Variation curve of f_i fragrance concentration

由图 1 可知,在迭代过程中香味浓度值变化曲线由小变大再变小,变化的范围较小,不利于全局搜索,而且中后期曲线斜率变化较小导致收敛速度缓慢且计算精度不高。因此用变化范围较宽和斜率变化较大的自适应权重因子重新定义香味浓度计算表达式,其计算公式如下:

$$b = 2 - 2 \times t/Ngen。 \tag{5}$$

$$f_i = 2 \times b \times r - b \text{ ,} \tag{6}$$

b 表示随迭代次数变化参数,则蝴蝶全局搜索公式更改为

$$x_i^{t+1} = \omega(t) x_i^t + (r^2 \times g^* - x_i^t) \times f_i \text{。} \tag{7}$$

2.3 正弦余弦算法指引机制

正弦余弦算法于 2016 年由澳大利亚学者 Mirjalili[4] 提出,该算法利用正余弦函数的周期性和振荡性趋于全局最优,算法中自适应参数能较好地平衡算法的全局搜索和局部开发。计算公式如下:

$$x_i^{t+1} = \begin{cases} x_i^t + r_1 \times \sin(r_2) \times | r_3 g^* - x_i^t | & r_4 < 0.5 \\ x_i^t + r_1 \times \cos(r_2) \times | r_3 g^* - x_i^t | & r_4 \geqslant 0.5 \end{cases} \text{,} \tag{8}$$

其中,r_1 等于式(5),$r_2 \in [0,2\pi]$ 之间的随机数,$r_3, r_4 \in [0,1]$ 之间的随机数。

高文欣等[10] 和宁杰琼等[12] 也使用正弦余弦算法来改进 BOA,高文欣等[10] 在算法迭代后期让所有蝴蝶个体执行正弦余弦算法,本质上实行算法的串联操作,但在一定程度上增加了算法的时间开销;宁杰琼等[12] 在局部搜索过程中以一定概率将正弦余弦算法与基本 BOA 的局部搜索公式共同寻优。而本文的操作方式与其不同,首先使用非线性自适应因子式(4)替换式(8)中 r_1,然后用正弦余弦算法替换基本 BOA 的局部搜索公式,更好地平衡正弦余弦算法的全局搜索和局部勘探能力。因此改进后的正弦余弦算法公式如下:

$$x_i^{t+1} = \begin{cases} x_i^t + \omega(t) \times \sin(r_2) \times | r_3 g^* - x_i^t | & r_4 < 0.5 \\ x_i^t + \omega(t) \times \cos(r_2) \times | r_3 g^* - x_i^t | & r_4 \geqslant 0.5 \end{cases} \text{。} \tag{9}$$

2.4 SCABOA 的伪代码

输入:种群规模 n、函数名称和最大迭代次数。

①在边界范围内随机产生种群,并计算此时最优解和最优值

②While $t < Ngen$

③计算式(4)和式(5)

④For $i = 1 : n$

⑤计算式(6)

⑥If $i \leqslant n/2$

⑦执行式(7)

⑧Else

⑨For $j = 1 :$ dim

⑩执行式(9)

⑪End

⑫End if

⑬越界处理

⑭计算蝴蝶个体目标值并与之前个体进行比较,如果优越则替换

⑮End for

⑯End While

输出:最优值和最优解

3 仿真实验与结果

为检验 SCABOA 的性能,选取 19 个基准函数进行测试,函数的具体表达式、维度、取值范围和理论值最优值如表 1 所示。实验环境为 Matlab2016a 和 Windows10,机器主频为 2.60 GHz ,内存为 8 GB。

利用 SCABOA 对 19 个基准函数进行求解,并与基本 BOA、SCA、SSA、GWO 和 CS 算法进行对比,以验证 SCABOA 的效果。参数设置:种群规模为 30,最大迭代次数为 500,其余参数设置与原论文相同。为避免偶然性,每种算法在 Matlab2016a 软件中独立运行 30 次,计算每个函数的平均值 Mean 和方差 Std,结果如表 2 所示,表中“-”表示文献未对相关函数进行测试,表中的加粗字体为所有实验中的最佳结果。

表 1 基准函数

Table 1 Benchmark functions

函数 Function	维度 Dim	范围 Range	最优值 Optimal value
Sphere: $f_1(x) = \sum_{i=1}^{n} x_i^2$	30	[-100,100]	0
Schwefel 2.22: $f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	10	[-10,10]	0
Schwefel 1.2: $f_3(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	10	[-100,100]	0

续表 1

Continued table 1

函数 Function	维度 Dim	范围 Range	最优值 Optimal value
Schwefel 2.21: $f_4(x)=\max_i\{\lvert x_i\rvert,1\leqslant i\leqslant n\}$	10	[-100,100]	0
Rosenbrock: $f_5(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	10	[-30,30]	0
Step: $f_6(x)=\sum_{i=1}^{n}(x_i+0.5)^2$	10	[-100,100]	0
Quartic: $f_7(x)=\sum_{i=1}^{n}ix_i^4+rand(0,1)$	10	[-1.28,1.28]	0
Schwefel: $f_8(x)=\sum_{i=1}^{n}[x_i^2\sin(\sqrt{\lvert x_i\rvert})]$	10	[-500,500]	$-418.983\times n$
Rastrigin: $f_9(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	10	[-5.12,5.12]	0
Ackley: $f_{10}(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_i^n x_i^2})-\exp(\frac{1}{n}\sum_i^n\cos2\pi x_i)+20+e$	10	[-32,32]	0
Griewank: $f_{11}(x)=1+\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})$	10	[-600,600]	0
Penalized: $f_{12}(x)=\frac{\pi}{n}\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4)$	10	[-50,50]	0
Penalized2: $f_{13}(x)=0.1\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_i+1)]+(x_n-1)^2\times[1+\sin^2(2\pi x_n)]+\sum_{i=1}^{n}u(x_i,5,100,4)$	10	[-50,50]	0
Foxholes: $f_{14}(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}(1/j+\sum_{i=1}^{2}(x_i-a_{ij})^6)\right)^{-1}$	2	[-65,65]	1
Kowalik: $f_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_i(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	4	[-5,5]	0.000 3
Six-Hump Camel: $f_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_1^4$	2	[-5,5]	-1.031 6
Branin: $f_{17}(x)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	[-2,2]	3
Hartman 3: $f_{18}(x)=-\sum_{i=1}^{4}c_i\exp(\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2)$	3	[0,1]	-3.86
Hartman 6: $f_{19}(x)=-\sum_{i=1}^{4}c_i\exp(\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2)$	4	[0,10]	-10.153 2

表 2　基准函数的实验结果

Table 2　Experimental results of reference function

f	指标 Index	SCABOA	BOA	SCA	SSA	GWO	CS	SIBOA
$f_1(x)$	Mean	**0**	1.3156e-11	12.606 0	1.8770e-07	1.0907e-27	10.802 3	1.28e-166
	Std	**0**	7.5462e-13	29.551 1	2.5680e-07	1.3910e-27	5.400 1	0
$f_2(x)$	Mean	**0**	4.7102e-09	9.9743e-10	0.045 4	6.5593e-33	0.004 7	8.80e-77
	Std	**0**	6.2189e-10	2.1436e-09	0.127 1	1.4601e-32	0.001 8	3.68e-76

续表 2

Continued table 2

f	指标 Index	SCABOA	BOA	SCA	SSA	GWO	CS	SIBOA
$f_3(x)$	Mean	**0**	1.0717e－11	0.005 7	3.7527e－07	1.1882e－25	0.050 9	5.51e－162
	Std	**0**	1.3529e－12	0.015 5	1.3493e－06	3.3894e－25	0.040 6	0
$f_4(x)$	Mean	**0**	5.2398e－09	0.001 2	2.3093e－05	2.9657e－18	0.117 4	7.96e－79
	Std	**0**	5.1966e－10	0.002 5	9.6970e－06	4.1874e－18	0.033 5	5.25e－78
$f_5(x)$	Mean	**1.3943e－16**	8.935 6	12.789 9	131.882 2	6.746 3	6.072 5	—
	Std	**2.9236e－16**	0.022 7	29.204 1	364.401 3	0.672 6	2.136 5	—
$f_6(x)$	Mean	**4.2398e－19**	1.206 7	0.454 8	9.5161e－10	0.008 1	1.0815e－05	—
	Std	**1.0047e－18**	0.331 9	0.180 6	3.6020e－10	0.044 4	5.5343e－06	—
$f_7(x)$	Mean	1.1994e－04	0.001 6	0.002 8	0.013 1	6.7062e－04	0.012 0	**2.93e－5**
	Std	1.1309e－04	6.1730e－04	0.003 0	0.008 0	4.4036e－04	0.005 2	**3.52e－5**
$f_8(x)$	Mean	**－4.1898e＋03**	－2.1109e＋03	－2.1777e＋03	－2.7982e＋03	－2.6908e＋03	－3.4475e＋03	—
	Std	**3.7002e－12**	192.375 7	147.098 6	305.023 1	390.166 8	141.142 2	—
$f_9(x)$	Mean	**0**	32.432 5	0.032 8	15.786 7	0.170 0	12.632 1	0
	Std	**0**	16.051 6	0.134 4	6.137 4	0.542 7	2.703 2	0
$f_{10}(x)$	Mean	**8.8818e－16**	2.4744e－09	2.4526e－06	0.652 7	7.0462e－15	0.602 5	8.8818e－16
	Std	**2.0059e－31**	1.0478e－09	5.6780e－06	0.946 5	1.5979e－15	0.521 0	2.0059e－31
$f_{11}(x)$	Mean	**0**	1.5542e－13	0.069 4	0.215 3	0.020 4	0.080 6	0
	Std	**0**	8.7604e－14	0.125 2	0.112 7	0.016 8	0.018 7	0
$f_{12}(x)$	Mean	**4.2735e－19**	0.109 7	0.115 8	1.418 1	0.004 6	0.009 5	—
	Std	**7.9882e－19**	0.068 0	0.049 7	1.296 1	0.008 5	0.011 1	—
$f_{13}(x)$	Mean	**1.1956e－18**	0.453 1	0.348 4	0.003 6	0.027 6	2.0490e－04	—
	Std	**1.6505e－18**	0.170 2	0.080 3	0.005 9	0.044 1	1.7105e－04	—
$f_{14}(x)$	Mean	**0.998 0**	1.343 6	1.861 3	1.163 4	4.815 4	**0.998 0**	—
	Std	**6.7752e－16**	0.588 3	0.997 0	0.526 6	4.343 6	6.7752e－16	—
$f_{15}(x)$	Mean	4.0633e－04	**3.8517e－04**	9.4201e－04	0.002 2	0.004 4	4.2505e－04	—
	Std	1.0742e－04	**5.5869e－05**	3.6319e－04	0.005 0	0.008 1	8.9601e－05	—
$f_{16}(x)$	Mean	－1.031 4	－1.031 5	**－1.031 6**	**－1.031 6**	－1.031 6	**－1.031 6**	—
	Std	3.7443e－04	1.4595e－04	4.1014e－05	**0**	3.4575e－08	**0**	—
$f_{17}(x)$	Mean	3.260 3	3.113 4	3.000 0	**3**	3.000 0	**3**	—
	Std	2.224 9	0.178 3	2.9964e－05	**0**	3.8635e－05	**0**	—
$f_{18}(x)$	Mean	－3.769 3	－3.838 0	－3.856 0	**－3.862 8**	－3.861 4	**－3.862 8**	—
	Std	0.081 6	0.027 7	0.003 4	**1.3550e－15**	0.002 0	**1.3550e－15**	—
$f_{19}(x)$	Mean	**－10.153 2**	－5.354 4	－2.411 4	－6.731 9	－9.477 9	**－10.153 2**	—
	Std	**1.8067e－15**	0.914 5	1.948 9	3.564 3	1.746 8	7.6489e－07	—

由表 2 第 3－8 列可知,SCABOA 求解的 19 个基准函数测试中,有 8 个函数(f_1－f_4、f_8、f_9、f_{11}和 f_{19})的平均计算结果达到理论最优值,完全胜出 BOA、SCA、SSA、GWO 和 CS 算法。未达到理论最优值的 11 个函数中,有 4 个函数 f_{15}－f_{18} 比 BOA、SCA、SSA、GWO 和 CS 算法的平均计算结果稍差且差别较小以外,另外 7 个函数的平均计算结果精度高出比较算法十几个数量级。尤其值得一提的是

Rosenbrock 函数,该函数理论最优值位于一个平滑、狭长的抛物线形山谷中,由于函数为优化算法提供的信息非常有限,使得众多算法求解其最优值变得十分困难,如 BOA、SCA、SSA、GWO 和 CS 算法求解的平均值从个位数到百位数不等,而 SCABOA 算法求解的平均值竟达到 1×10^{-16},与理论最优值更为接近。可见,改进的 SCABOA 算法计算精度更高。

为直观展示 SCABOA 的收敛速度和寻优精度,给出部分函数前 100 代的收敛曲线图。从图 2－7 可知,SCABOA 收敛曲线位于图形最下方,收敛速度最快,且计算精度最高。

与高文欣等[10,14]的改进蝴蝶优化算法进行比较,其测试函数部分与本文相同,求解结果相同部分未列出(具体数据可参考文献),挑选求解结果不同的函数进行比较。如高文欣等[10]的 f_4 即本文中的

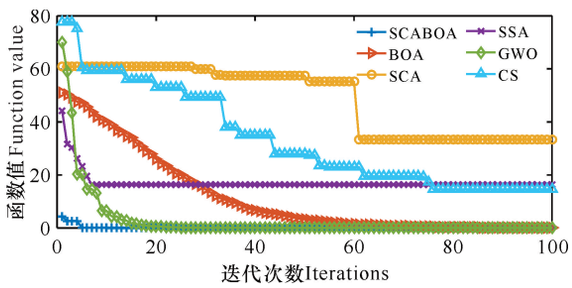

图 2　f_4 函数收敛曲线

Fig. 2　Convergence curve of f_4 function

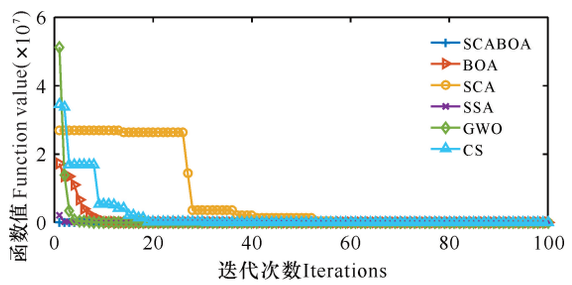

图 3　f_5 函数收敛曲线

Fig. 3　Convergence curve of f_5 function

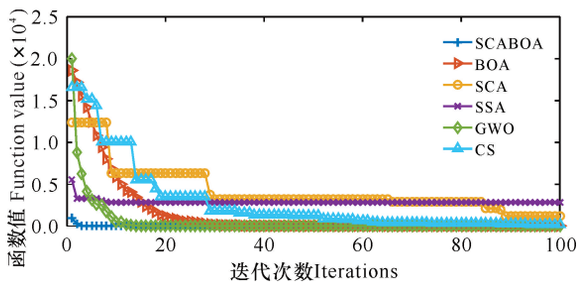

图 4　f_6 函数收敛曲线

Fig. 4　Convergence curve of f_6 function

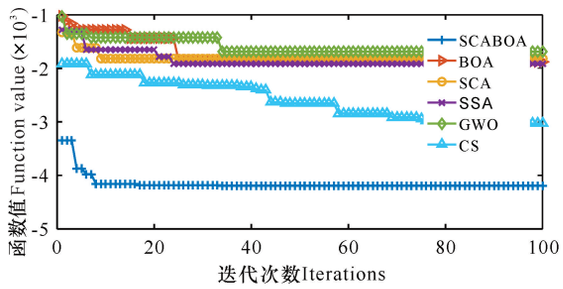

图 5　f_8 函数收敛曲线

Fig. 5　Convergence curve of f_8 function

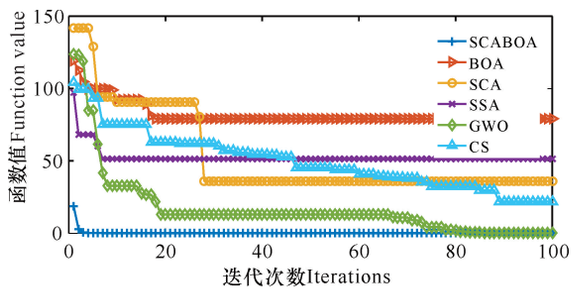

图 6　f_9 函数收敛曲线

Fig. 6　Convergence curve of f_9 function

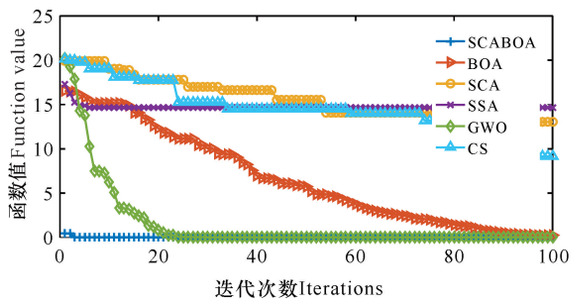

图 7　f_{10} 函数收敛曲线

Fig. 7　Convergence curve of f_{10} function

f_{12},高文欣等[10]求解的平均值精度仅为 10^{-7},而本文算法求解的平均精度高达 10^{-19},高出 12 个数量级;如高文欣等[14]使用的 f_2 即本文中的 f_2,高文欣等[14]求解的平均值精度仅为 $1.36e^{-81}$,而本文算法求解的平均精度达到理论值。再与王依柔等[11]的算法(SIBOA)进行比较,由于本文中 19 个测试函数全部包含王依柔等[11]使用的测试函数,故将其结果列于表 2,王依柔等[11]使用的种群规模为 40,最大迭代次数为 1 000。比较结果:函数 f_9-f_{11} 的计算结果相同,f_7 的计算结果略次于王依柔等[11],函数 f_1-f_4 均优于王依柔等[11];本文算法在种群规模为 30 和最大迭代次数为 500 的情况下,总体效果依然优于王依柔等[11]的算法,可见本文算法的优越性。

4　结束语

本文在 BOA 基础之上引入非线性自适应因子、

重新定义香味浓度计算公式,嵌入正弦余弦算法,提出一种融合正弦余弦算法的改进蝴蝶优化算法(SCABOA)。比较实验结果可知,SCABOA 算法性能表现优越,这与文章第 2 部分的改进策略分析相吻合:①通过在自身认知部分引入非线性自适应因子,使得算法迭代初期在较宽范围内搜索,而迭代后期在较窄范围搜索;②设计变化范围较宽和斜率变化较大的香味浓度计算公式;③在后半个种群中利用正弦余弦算法替换基本 BOA 的随机搜索,使得算法搜索更具有目标性。这 3 种策略的合力结果使得 SCABOA 算法具有更好的全局搜索和局部搜索能力。

参考文献

[1] YANG X S,DEB S. Cuckoo search via levy flights [C]. Proceedings of World Congress on Nature & Biologically Inspired Computing (NaBIC2009). India:IEEE Publications,2009:210-214.

[2] MIRJALILI S,GANDOMI A H,MIRJALILI S Z,et al. Salp swarm algorithm:A bio-inspired optimizer for engineering design problems [J]. Advances in Engineering Software,2017,114:163-191.

[3] MIRJALILI S,MIRJALILI S M,LEWIS A. Grey wolf optimizer [J]. Advances in Engineering Software,2014,69:46-61.

[4] MIRJALILI S. A sine cosine algorithm for solving optimization problems [J]. Knowledge-Based Systems,2016,96:120-133.

[5] MIRJALILI S,LEWIS A. The whale optimization algorithm [J]. Advances in Engineering Software,2016,95:51-67.

[6] YANG X S. Flower pollination algorithm for global optimization [C]. Unconventional Computation and Natural Computation (UCNC) 2012,Lecture Notes in Computer Science. Berlin:Springer,2012,7445:240-249.

[7] 王依柔,张达敏,徐航,等. 基于自适应扰动的疯狂蝴蝶算法 [J]. 计算机应用研究,2020,37(11):3276-3280.

[8] SHARMA S,SAHA A K,MAJUMDER A,et al. MP-BOA-A novel hybrid butterfly optimization algorithm with symbiosis organisms search for global optimization and image segmentation [J]. Multimedia Tools and Applications,2021,80(8):12035-12076.

[9] PROVAS K R,BARUN M,SUBHAM K. Renewable energy-based economic load dispatch using two-step biogeography-based optimization and butterfly optimization algorithm [J]. International Journal of Swarm Intelligence Research,2020,11(4):24-60.

[10] 高文欣,刘升,肖子雅,等. 全局优化的蝴蝶优化算法[J]. 计算机应用研究,2020,37(10):2966-2970.

[11] 王依柔,张达敏,徐航,等. 基于自适应扰动的疯狂蝴蝶算法[J]. 计算机应用研究,2020,37(11):3276-3280.

[12] 宁杰琼,何庆. 混合策略改进的蝴蝶优化算法[J/OL]. 计算机应用研,2020,38(5). [2020-12-17]. https://www.arocmag.com/article/02-2021-05-017.html.

[13] 高文欣,刘升,肖子雅,等. 柯西变异和自适应权重优化的蝴蝶算法[J]. 计算机工程与应用,2020,56(15):43-50.

[14] 高文欣,刘升,肖子雅. 收敛因子和黄金正弦指引机制的蝴蝶优化算法[J]. 计算机工程与设计,2020,41(12):3384-3389.

[15] 谢聪,封宇. 一种改进的蝴蝶优化算法[J]. 数学的实践与认识,2020,50(13):105-114.

Butterfly Optimization Algorithm Based on Sine Cosine Algorithm

ZHENG Hongqing[1],FENG Wenjian[2],ZHOU Yongquan[3]

(1. College of Information Engineering,Guangxi University of Foreign Languages,Nanning,Guangxi,530222,China;2. Liuzhou Railway Vocational Technical College,Liuzhou,Guangxi,545616,China;3. College of Artificial Intelligence,Guangxi University for Nationnalities,Nanning,Guangxi,530006,China)

Abstract:Aiming at the defects of the butterfly optimization algorithm,such as slow convergence speed,poor searching precision and easy to fall into local optimum,a butterfly optimization algorithm based on sine cosine

algorithm is proposed. Firstly, a nonlinear adaptive factor is introduced into the self-cognition part of butterfly. Secondly, the calculation formula of fragrance concentration is redefined. Finally, the improved sine and cosine algorithm is introduced in the local search stage. By testing 19 benchmark functions, the experimental results show that the proposed algorithm is superior to Butterfly Optimization Algorithm (BOA), Sine Cosine Algorithm (SCA), Salp Swarm Algorithm (SSA), Grey Wolf Optimizer (GWO) and Cuckoo Search Algorithm (CS) in terms of convergence speed, optimization accuracy and robustness. Compared with other improved butterfly optimization algorithms, it also has some advantages in optimization accuracy.

Key words: sine cosine algorithm, function optimization, butterfly optimization algorithm, convergence factor, self-adaption

责任编辑:陆　雁