控制与决策
Control and Decision

混合柯西变异和均匀分布的蝗虫优化算法

何庆， 林杰， 徐航

引用本文:
何庆, 林杰, 徐航. 混合柯西变异和均匀分布的蝗虫优化算法[J]. 控制与决策, 2021, 36(7): 1558−1568.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2019.1609

混合柯西变异和均匀分布的蝗虫优化算法

何 庆[1,2†], 林 杰[1,2], 徐 航[1]

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

摘 要: 由于位置更新公式存在局部开发能力较强而全局探索能力较弱的缺陷,导致蝗虫优化算法(GOA)易陷入局部最优以及早熟收敛,对此,提出一种混合柯西变异和均匀分布的蝗虫优化算法(HCUGOA). 受柯西算子和粒子群算法的启发,提出具有分段思想的位置更新方式以增加种群多样性,增强全局探索能力;将柯西变异算子与反向学习策略相融合,对最优位置即目标值进行变异更新,提高算法跳出局部最优的能力;为了更好地平衡全局探索与局部开发,将均匀分布函数引入非线性控制参数c,构建新的随机调整策略.通过对12个基准函数和CEC2014函数进行仿真实验以及Wilcoxon秩和检验的方法来评估改进算法的寻优能力,实验结果表明,HCUGOA算法在收敛精度和收敛速度等方面都得到极大的改进.

关键词: 蝗虫优化算法；粒子群算法；柯西变异；均匀分布；反向学习；时间复杂度；统计检验

中图分类号: TP301 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2019.1609 **开放科学(资源服务)标识码(OSID):**

引用格式: 何庆,林杰,徐航. 混合柯西变异和均匀分布的蝗虫优化算法[J]. 控制与决策, 2021, 36(7): 1558-1568.

Hybrid Cauchy mutation and uniform distribution of grasshopper optimization algorithm

HE Qing[1,2†], *LIN Jie*[1,2], *XU Hang*[1]

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

Abstract: Due to the strong local exploitation ability and the weak global exploration ability of the location update formula, the grasshopper optimization algorithm (GOA) is easy to fall into local optimum and easy to prematurely converge. Therefore, this paper proposes a hybrid Cauchy mutation and uniform distribution of the grasshopper optimization algorithm (HCUGOA). Firstly, inspired by the Cauchy operator and particle swarm optimization algorithm, a location update method with segmentation idea is proposed to increase the diversity of the population and to enhance the global exploration ability. Then, the fusion of Cauchy mutation and opposition-based learning and the variation of the optimal position which is the target value improve the ability of the algorithm to jump out of the local optimum. Finally, in order to better balance the global exploration and local exploitation, the uniform distribution function is introduced into the nonlinear control parameter c, so that a new random adjustment strategy can be built. The optimization performance of the improved algorithm is evaluated by a sets of simulation experiments and Wilcoxon's test on 12 benchmark functions and modern CEC 2014 functions. The experimental results show that the HCUGOA has been greatly improved in terms of convergence accuracy and convergence speed.

Keywords: grasshopper optimization algorithm；particle swarm optimization algorithm；Cauchy mutation；uniform distribution；opposition-based learning；time complexity；statistical test

0 引 言

蝗虫优化算法(GOA)[1]是一种新型群智能优化算法. 虽然GOA提出时间较短,但GOA不仅与遗传算法(GA)[2]、蚁群算法(ACO)[3]、粒子群算法(PSO)[4]等传统的优化算法相比具有竞争性,而且与鲸鱼算法(WOA)[5]、蚁狮算法(ALO)[6]、樽海鞘群算法(SSA)[7]、

收稿日期: 2019-11-18；**修回日期:** 2020-03-15.

基金项目: 贵州省科技计划重大专项项目(黔科合重大专项字[2018]3002，黔科合重大专项字[2016]3022)；贵州省公共大数据重点实验室开放课题(2017BDKFJJ004)；贵州省教育厅青年科技人才成长项目(黔科合KY字[2016]124)；贵州大学培育项目(黔科合平台人才[2017]5788).

责任编委: 陈家伟.

†通讯作者. E-mail: qhe@gzu.edu.cn.

正弦余弦算法(SCA)[8]等新型群智能算法相比也具有一定的竞争性. 目前,GOA已成功地应用于多个工程领域[9-12].

GOA具有原理简单、易于实现等优点,但该算法仍存在易陷入局部最优和易早熟收敛等缺点,因而众多学者提出了许多改进优化算法. 文献[13]利用动态权重机制促进算法开发,并引入随机跳跃策略避免算法陷入局部最优;文献[14]采用高斯变异和Levy-fight策略来增强种群多样性;文献[15]提出了混合策略蝗虫优化算法(MGOA),通过变异更新和随机突变来增大搜索区域以避免陷入局部值;文献[16]引入自适应策略而提出的SA-CGOA具有更好的求解质量和收敛速度.

针对GOA存在的问题,本文提出混合柯西变异和均匀分布的蝗虫优化算法(HCUGOA),并通过5组仿真实验验证本文所提出算法的有效性和优越性,实验结果表明,HCUGOA能显著提高算法寻优性能.

1　蝗虫优化算法

蝗虫优化算法(GOA)仿生原理:根据蝗虫成虫大范围搜索和幼虫小范围移动,并整体向着食物源(即目标)移动的方式,实现蝗虫位置优化过程,其中将蝗虫个体映射为搜索空间的点;用数学模型来模拟个体间相互作用;寻找食物源为寻优过程;蝗虫位置映射为解,位置的优劣映射为适应度;位置不断更新映射为优化过程中最优解的选取过程.

GOA的具体实现步骤如下:

1) 初始化种群大小N、空间维度dim、最大迭代次数$T_{\max}$及参数$c_{\max}$和$c_{\min}$,初始化种群位置X_i,$i=(1,2,\dots,N)^{\dim}$.

2) 计算个体适应度,并将最优值存到目标值$\hat{T}_d$.

3) 通过下式更新参数c:

$$c=c_{\max}-t\frac{c_{\max}-c_{\min}}{T_{\max}}. \tag{1}$$

其中:$c_{\max}$、$c_{\min}$是参数c的最大值和最小值,$T_{\max}$是最大迭代次数,t是当前迭代次数.

4) 通过下式更新蝗虫位置:

$$x_i^d=c\Big(\sum_{j=1,j\neq i}^{N}c\frac{\mathrm{ub}_d-\mathrm{lb}_d}{2}s(|x_j^d-x_i^d|)\frac{x_j-x_i}{d_{ij}}\Big)+\hat{T}_d. \tag{2}$$

其中:x_i^d是第i只蝗虫在第d维的位置,$d=(1,2,\dots,\dim)$;ub_d、lb_d是第i只蝗虫在第d维的上下界;$\hat{T}_d$为蝗虫群的目标位置;$d_{ij}=|x_j^d-x_i^d|$是第i只与第j只蝗虫在第d维的距离;$(x_j-x_i)/d_{ij}$是第i只蝗虫与第j只蝗虫的单位向量;s是个体之间的相互影响函数,有

$$s(r)=f\mathrm{e}^{-r/l}-\mathrm{e}^{-r}, \tag{3}$$

f、l分别为吸引强度参数和吸引尺度参数,本文取值$f=0.5,l=1.5$.

5) 计算每个个体的适应度,并择优更新目标值.

6) 判断是否到达最大迭代次数,若是,则停止循环,进行步骤7);否则,重复步骤3)～5).

7) 输出目标位置$\hat{T}_d$和全局最优值.

2　HCUGOA算法

2.1　服从均匀分布随机调整策略

GOA的控制参数c是协调探索与开发的关键. 由式(1)可知,c是随着迭代次数动态变化的且是线性递减函数,在搜索过程中难以适应优化的实际情况:在迭代前期,参数c下降过快易导致蝗虫不能遍历更多搜索空间,最终导致全局探索不足;在迭代后期,c下降过慢,会导致局部开发受限,收敛速度慢. 为解决这一问题,重新构建新函数$c(t)$,即

$$c(t)=\delta\Big(\lg\Big(\frac{c_{\max}}{T_{\max}}\Big)^{-20t}+\exp\Big(\frac{t}{T_{\max}}\Big)^{-20(c_{\max}-c_{\min})}\Big), \tag{4}$$

其中δ是服从$[0,T_{\max}]$之间均匀分布的随机数,且δ的取值范围介于0～1之间.

由式(4)可知,$c(t)$是非线性递减函数,$c(t)$随迭代次数增加在迭代期间呈非线性状态减小,具体表现为:前期函数下降缓慢,给全局探索充分时间;后期函数迅速下降收敛,加快算法收敛速度. 同时,引入服从均匀分布随机数δ,前期让参数c有可能继续保持较大值,一定程度上加强全局搜索;后期参数c随着迭代次数增加而减小,且变化比较缓慢. 此时服从均匀分布的随机数可能产生较大的值让参数c实现动态变化,从而加快算法开发和收敛速度.

2.2　结合柯西算子和分段思想的更新策略

由式(2)可知,基本GOA位置更新由自身位置与其他所有蝗虫间交互力及目标位置决定,因此,GOA具有很强的局部开发能力而全局探索能力不足. 为更好地提升算法性能,本节对算法进行分段处理.

在算法搜索前半段,受PSO对每代最优解进行记忆保存的思想启发,对式(2)进行改进,有

$$X_{\mathrm{new}}(t+1)=b_1\times W+b_2\times r_1(X_{\mathrm{best}}(t)-X_i(t))+b_2\times r_2(X_j(t)-X_k(t)). \tag{5}$$

其中:$X_{\mathrm{new}}(t+1)$是第i只蝗虫在第$t+1$代的位置;$X_{\mathrm{best}}(t)$是第t代最优位置;$X_i(t)$是第i只蝗虫在第

t代的位置; $X_j(t)$和$X_k(t)$是第t代两个随机位置; r_1和r_2分别为区间$[0,1]$内产生的随机数; b_1是记忆系数, b_2是信息交流系数, 有

$$b_1 = \exp\left(c(t) - 30\frac{t}{T_{\max}}\right)^{-t}, \tag{6}$$

$$b_2 = \left(\frac{T_{\max}-t}{T_{\max}}\right)^{N}; \tag{7}$$

W是与所有蝗虫相互联系并决定位置更新的步长函数, 即

$$W = \sum_{j=1,j\neq i}^{N} c(t)\frac{\mathrm{ub}_d - \mathrm{lb}_d}{2}s(|x_j^d - x_i^d|)\frac{x_j - x_i}{d_j}; \tag{8}$$

其余参数已在第1节阐述. 与式(2)相比, 式(5)分为3个部分: 1)利用了除最优解外的全部位置信息, 有助于个体间进行信息交互, 增强全局探索能力; 2)由自身位置和最优位置决定, 加强算法开发能力和收敛速度; 3)通过随机选择两个个体引导位置更新, 增强了全局探索能力.

在算法搜索后半段, 引入柯西算子作为变异步长, 当个体陷入局部最优时, 较大的步长可以帮助个体跳出局部极值; 当个体接近收敛, 并在搜索最优解时, 较小步长可以加速个体的收敛, 公式如下:

$$X_i{}^d(t+1) = X_i{}^d(t) + \mathrm{Cauchy} \oplus (X_{\mathrm{best}}{}^d(t) - X_k{}^d(t)). \tag{9}$$

其中: X_k^d是随机选择的第k只蝗虫在第d维的位置, Cauchy是柯西算子. 一维标准的柯西分布的概率密度函数表达式如下:

$$f(x) = \frac{1}{\pi}\left(\frac{1}{x^2+1}\right), \ -\infty < x < \infty. \tag{10}$$

由数学知识可知, 柯西算子具有较长的“尾巴”, 两端较长的分布可使个体具有更高概率跳到更好位置, 使个体逃离局部最优, 同时, 中心点较小的峰值表明柯西算子搜索领域空间的时间花费更少.

判断算法是进行前半段探索还是后半段开发由控制概率P_c决定, P_c可表示为

$$P_c = \frac{1}{2}T_{\max}. \tag{11}$$

算法前期需进行大范围全局探索, 后期需进行小范围的局部开发并避免算法过早收敛. 所以对位置更新引入分段思想: 前期对最优解进行记忆保存, 加强全局寻优能力, 并引入记忆系数和交流系数, 通过个体自身位置、最优位置及两个随机位置共同决定新位置; 后期引入柯西算子, 一定程度上降低算法陷入局部最优概率, 并加快算法的收敛速度.

算法1 结合柯西算子和分段思想的策略.

1) if rand $< P_c$

2) 根据式(5)～(8)更新个体位置;

3) else

4) 根据式(9)和(10)更新个体位置;

5) end

2.3 融合柯西变异和反向学习策略

基本GOA中, 目标位置更新依赖于每次迭代后的位置更新, 重新计算适应度, 选择最优位置对目标位置进行取代, 但未对目标位置进行主动的扰动更新, 导致算法易陷入局部最优. 所以, 本节融合柯西变异和反向学习策略, 依概率对目标位置进行随机扰动更新, 避免算法陷入局部最优.

反向学习是Tizhoosh[17]提出的一种新技术, 目的是基于当前解, 寻到其对应的反向解, 通过评估选择并保存更好的解. 为更好地引导个体寻到最优解, 将反向学习融入GOA中, 数学描述如下:

$$X_{\mathrm{best}}^*(t) = \mathrm{ub} + r \oplus (\mathrm{lb} - X_{\mathrm{best}}(t)), \tag{12}$$

$$X_{\mathrm{new}}(t+1) = b_3 \oplus (X_{\mathrm{best}}(t) - X_{\mathrm{best}}^*(t)). \tag{13}$$

其中: $X_{\mathrm{best}}^*(t)$是第t代目标解X_{best}的反向解; $X_{\mathrm{new}}(t+1)$是第$t+1$代目标解; ub和lb是上下界; r是服从$(0,1)$标准均匀分布$1 \times \dim$的随机数矩阵(其中dim为算法搜索空间维数); b_3是伪信息交流系数, 表达式如下:

$$b_3 = \left(\frac{T_{\max}-t}{T_{\max}}\right)^{t}. \tag{14}$$

将柯西算子引入目标位置更新, 发挥柯西算子的调节能力, 增强算法跳出局部最优的能力, 有

$$X_{\mathrm{new}}(t+1) = \mathrm{Cauchy} \oplus X_{\mathrm{best}}(t). \tag{15}$$

为提高算法寻优性能, 将反向学习策略和柯西算子扰动策略在一定概率下交替执行, 动态地随机更新目标位置. 反向学习策略中, 通过一般反向学习策略得到反向解, 增大算法搜索范围, 同时, 式(12)中的上下界ub和lb是动态变化的, 相对于固定边界的策略更利于算法优化; 柯西变异策略中运用变异算子对最优位置进行变异以产生新解, 一定程度上改善了算法易陷入局部最优的缺陷. 决定选择何种策略进行更新的选择概率P_s定义如下:

$$P_s = -\exp\left(1 - \frac{t}{T_{\max}}\right)^{20} + \theta, \tag{16}$$

其中θ为调节因子, 经多次实验, θ取值为0.05时函数优化结果最优.

算法2 融合柯西变异和反向学习策略.

1) if rand$< P_s$

2) 根据式(12)～(14)反向学习策略更新目标位置;

3) else

4) 根据式(15)柯西变异策略更新目标位置;

5) end

对目标位置进行扰动更新虽能让算法跳出局部最优,但不能保证新位置优于原目标位置,因此,在扰动更新操作后加入贪婪机制,通过比较新旧目标位置的适应度后再决定是否更新目标位置.

算法3　贪婪算法.

1) if $f(X_{\text{new}}) < f(X_{\text{best}})$

2) $X_{\text{best}} = X_{\text{new}}$;

3) else

4) $X_{\text{best}} = X_{\text{best}}$;

5) end

基于贪婪选择策略促进算法向着期望寻找到的目标位置方向进化,让目标位置能够充分发挥引导作用,使算法获得更好的收敛速度和精度.

2.4　HCUGOA算法的实现步骤

综上所述,本文提出的改进算法的流程如图1所示.

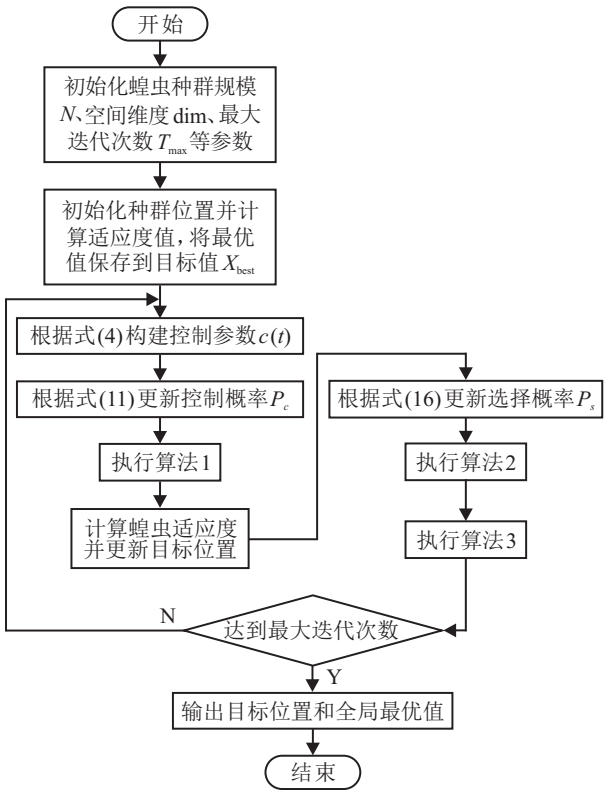

图 1　HCUGOA算法流程

2.5　算法时间复杂度分析

时间复杂度间接反映了算法的收敛速度. 在GOA中,假设参数初始化(种群规模N、空间维度n等参数)所需执行时间为x_1,产生均匀分布随机数时间为x_2,求给定适应度函数时间为$f(n)$,则GOA种群初始阶段的时间复杂度为

$$O(x_1 + N(nx_2 + f(n))) = O(n + f(n)). \tag{17}$$

设更新函数c的时间为x_3,每一维按式(2)更新位置所需时间为x_4,比较位置优劣和择优更新目标位置的时间分别为x_5和x_6,该阶段的时间复杂度为

$$O(N(x_3 + nx_4 + f(n) + x_5) + x_6) = O(n + f(n)). \tag{18}$$

所以基本GOA求解最优解的总时间复杂度为

$$T(n) = O(n + f(n)) + O(n + f(n)) = O(n + f(n)). \tag{19}$$

在HCUGOA中,初始化参数所需时间与基本GOA相同,因此,HCUGOA在初始化阶段的时间复杂度与式(17)一致. 在算法循环部分,设更新函数$c(t)$的时间为w_1,更新P_c的时间为w_2,执行算法1的时间为w_3,更新P_s的时间为w_4,执行算法2和算法3的时间为w_5和w_6,则循环部分的时间复杂度为

$$O(w_1 + w_2 + N(nw_3 + f(n) + x_5) + x_6 + w_4 + N(w_5 + w_6)) = O(n + f(n)). \tag{20}$$

由上述分析可得HCUGOA求解每一代最优解的总时间复杂度为

$$T(n) = O(n + f(n)) + O(n + f(n)) = O(n + f(n)). \tag{21}$$

综上所述,HCUGOA与基本GOA相比,时间复杂度一致,从而表明本文针对GOA缺陷所提出的改进策略没有增加算法时间复杂度.

3　仿真实验与结果分析

为全面验证本文算法性能,同时验证每种改进策略的有效性,本文的仿真实验分为5个部分进行:

1) 将HCUGOA与基本GOA比较,通过实验数据对比分析本文改进算法的有效性和可行性;

2) 将HCUGOA与改进控制参数c的GOA1、结合柯西算子和分段思想的更新策略的GOA2、融合柯西变异和反向学习策略的GOA3进行对比,以验证不同改进策略的有效性;

3) 将HCUGOA与混合策略改进鲸鱼算法(MS-WOA)[5]、正弦余弦改进算法(m-SCA)[8]以及最新的蝗虫改进算法IGOA[14]和MGOA[15]进行比较,以验证本文改进算法的优越性和竞争性;

4) 通过Wilcoxon秩和检验验证本文算法与其他对比算法之间的显著性差异;

5) 在CEC2014基准函数中选取部分单峰、多峰、混合和复合类型的函数进行优化测试,以验证本文算

法的有效性和鲁棒性.

实验中引入12个具有不同寻优特征的经典测试函数,如表1所示. 其中: $F_1 \sim F_6$ 是连续单峰函数, $F_7 \sim F_{11}$ 是复杂非线性多峰函数, F_{12} 是固定低维函数. 单峰函数局部最优即为全局最优,用来测试算法收敛速度和收敛精度;多峰函数具有多个局部极值,常用来测试算法跳出局部最优的能力和全局搜索能力.

表1 测试函数

编号	函数名称	维度	定义域	理论值
F_1	Sphere	10/30/200	$[-100,100]$	0
F_2	Schwefel 2.22	10/30/200	$[-10,10]$	0
F_3	Schwefel 1.2	10/30/200	$[-100,100]$	0
F_4	Schwefel 2.21	10/30/200	$[-100,100]$	0
F_5	Rosenbrock	10/30/200	$[-30,30]$	0
F_6	Quartic	10/30/200	$[-1.28,1.28]$	0
F_7	Schwefel 2.26	10/30/200	$[-500,500]$	−12 569.5
F_8	Rastrigin	10/30/200	$[-5.12,5.12]$	0
F_9	Ackley	10/30/200	$[-32,32]$	0
F_{10}	Griewank	10/30/200	$[-600,600]$	0
F_{11}	Penalized	10/30/200	$[-50,50]$	0
F_{12}	Kowalik	4	$[-5,5]$	0.000 3

为公平起见,算法基本参数设置相同:种群规模 $N=30$,空间维数 $\dim=10/30/200$, 最大迭代次数 $T_{\max}=500$. 算法具体参数设置:对于GOA, $c_{\max}=1$, $c_{\min}=0.000\,04$;对于MS-WOA, $a_{\max}=2$, $a_{\min}=0$;对于m-SCA, $a=2$;对于GOA3和HCUGOA, $\theta=0.05$;其他改进GOA与基本GOA设置一致.

3.1 与基本GOA算法比较

将基本GOA与HCUGOA在空间维数 $\dim=30$ 的条件下对10个测试函数进行求解,并通过5个性能评估仿真结果,算法独立运行30次的数据如表2所示.

最优值和最差值反映了算法寻优结果的质量. 从表2数据可知:对于函数 F_1、F_2、F_3、F_4、F_8、F_{10}, HCUGOA均寻到理论最优值0;对于函数 F_5 和 F_6, HCUGOA虽没有寻到理论值,但是与GOA相比最优值均相差7个数量级;对于函数 F_9,其理论值为0, HCUGOA没有寻到理论值,但是与基本GOA相比,具有更高的收敛精度;对于函数 F_7,与基本GOA相比, HCUGOA能够得到更优的结果.

平均值反映算法收敛速度,标准差反映算法稳定性和鲁棒性.对于函数 F_6,虽未寻到理论值,但从平均值和标准差可知HCUGOA具有更高的寻优精度且寻优结果稳定;对于 F_9,与基本GOA相比,标准差相差32个数量级,说明HCUGOA对 F_9 求解时,得到的高精度结果并非偶然. 总之,对于10个测试函数,与GOA相比, HCUGOA能够取得更优的结果,具有更强的有效性和可行性.另外从平均耗时可以看出,本文改进算法的平均耗时比原始算法少,从而表明本文提出的改进策略没有增加算法时间复杂度.

表2 HCUGOA与GOA结果比较

	算法	最优值	最差值	均值	标准差	耗时/s
F_1	GOA	2.77e+02	9.83e+02	7.02e+02	2.03e+02	53.781 8
	HCUGOA	0.00e+00	0.00e+00	0.00e+00	0.00e+00	28.733 5
F_2	GOA	1.12e+01	2.86e+01	2.05e+01	4.58e+00	55.356 4
	HCUGOA	0.00e+00	0.00e+00	0.00e+00	0.00e+00	29.275 3
F_3	GOA	1.99e+03	4.50e+03	3.18e+03	7.73e+02	53.511 9
	HCUGOA	0.00e+00	0.00e+00	0.00e+00	0.00e+00	28.470 0
F_4	GOA	2.05e+01	3.71e+01	2.50e+01	4.00e+00	52.692 5
	HCUGOA	0.00e+00	0.00e+00	0.00e+00	0.00e+00	27.700 6
F_5	GOA	1.71e+04	1.17e+05	7.05e+04	3.00e+04	52.176 5
	HCUGOA	1.78e-03	2.12e+01	6.89e+00	7.26e+00	27.978 9
F_6	GOA	3.00e+00	5.68e+00	3.96e+00	7.06e-01	52.867 6
	HCUGOA	4.96e-07	2.31e-05	1.08e-05	6.64e-06	28.110 2
F_7	GOA	−6.74e+03	−5.11e+03	−6.29e+03	4.39e+02	52.822 7
	HCUGOA	−9.99e+122	−3.44e+92	−4.37e+121	−1.85e+122	28.153 7
F_8	GOA	9.83e+01	1.87e+02	1.57e+02	2.19e+01	52.867 6
	HCUGOA	0.00e+00	0.00e+00	0.00e+00	0.00e+00	28.110 2
F_9	GOA	1.16e+00	3.40e+00	2.09e+00	4.88e-01	53.053 1
	HCUGOA	8.88e-16	8.88e-16	8.88e-16	1.00e-31	28.033 9
F_{10}	GOA	1.67e+00	9.61e+00	6.60e+00	2.27e+00	53.019 2
	HCUGOA	0.00e+00	0.00e+00	0.00e+00	0.00e+00	28.202 6

3.2 与不同改进策略对比

将HCUGOA与改进控制参数 $c(t)$ 的GOA1、结合柯西算子和分段思想的更新策略算法GOA2、融合柯西变异和反向学习策略算法GOA3进行对比,进一步验证不同改进策略的有效性.算法参数与3.1节相同,算法独立运行30次的仿真结果如表3所示.

由表3可知, HCUGOA对单峰函数寻优时,前4个函数的4个性能指标均达到理论值,而对于 F_5 和 F_6, HCUGOA的收敛精度也得到了一定程度的提升. 对于多峰函数, HCUGOA对 F_8 和 F_{10} 寻到理论值0, F_7 和 F_9 虽未寻到最优,但相对于其他改进算法, HCUGOA有更高的求解精度和稳定性; HCUGOA对 F_{11} 求解时,标准差没有GOA3的显著,说明前两种改进策略相对抑制了HCUGOA对 F_{11} 的寻优.

具体来说, GOA2和GOA3对 $F_1 \sim F_4$、F_8 和 F_{10} 的有效性显著,从前4个评价指标可以看出,结合柯

表3　不同改进策略对比结果

	算法	最优值	最差值	均值	标准差	耗时/s
F_1	GOA	2.77e+02	9.83e+02	7.02e+02	2.03e+02	53.781 8
	GOA1	4.49e+00	3.00e+01	1.96e+01	5.85e+00	52.323 7
	GOA2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	**26.839 0**
	GOA3	0.00e+00	0.00e+00	0.00e+00	0.00e+00	50.382 3
	HCUGOA	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	28.733 5
F_2	GOA	1.12e+01	2.86e+01	7.02e+02	4.58e+00	55.356 4
	GOA1	2.49e+00	8.42e+00	1.96e+01	1.84e+00	53.480 9
	GOA2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	**27.098 1**
	GOA3	0.00e+00	0.00e+00	0.00e+00	0.00e+00	52.735 9
	HCUGOA	0.00e+00	0.00e+00	0.00e+00	0.00e+00	29.275 3
F_3	GOA	1.99e+03	4.50e+03	7.02e+02	7.73e+02	53.511 9
	GOA1	1.10e+03	2.50e+03	1.96e+01	3.82e+02	50.751 7
	GOA2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	**26.433 1**
	GOA3	0.00e+00	0.00e+00	0.00e+00	0.00e+00	50.950 0
	HCUGOA	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	28.470 0
F_4	GOA	2.05e+01	3.71e+01	7.02e+02	4.00e+00	52.320 5
	GOA1	4.55e+00	1.23e+01	1.96e+01	2.12e+00	50.588 1
	GOA2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	**25.988 2**
	GOA3	0.00e+00	0.00e+00	0.00e+00	0.00e+00	50.479 5
	HCUGOA	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	28.049 6
F_5	GOA	1.71e+04	1.17e+05	7.02e+02	3.00e+04	52.692 5
	GOA1	2.03e+02	2.46e+03	1.96e+01	7.03e+02	50.099 1
	GOA2	2.89e+01	2.90e+01	0.00e+00	**2.12e-02**	**26.028 4**
	GOA3	2.84e+01	2.87e+01	0.00e+00	5.04e-02	50.324 9
	HCUGOA	**1.78e-03**	**2.12e+01**	**0.00e+00**	7.26e+00	27.700 6
F_6	GOA	3.00e+00	5.68e+00	7.02e+02	7.06e-01	52.176 5
	GOA1	1.79e-02	3.99e-02	1.96e+01	6.50e-03	50.986 0
	GOA2	2.83e-06	8.96e-05	0.00e+00	2.62e-05	**26.100 2**
	GOA3	9.64e-07	3.29e-05	0.00e+00	8.63e-06	50.432 2
	HCUGOA	**4.96e-07**	**2.31e-05**	**0.00e+00**	**6.64e-06**	27.978 9
F_7	GOA	−6.74e+03	−5.11e+03	7.02e+02	4.39e+02	52.822 7
	GOA1	−48.38e+03	−7.58e+03	1.96e+01	2.23e+02	50.733 9
	GOA2	−1.08e+04	−6.71e+03	0.00e+00	9.37e+02	**26.099 8**
	GOA3	−1.84e+86	−2.74e+50	0.00e+00	3.35e+85	50.610 4
	HCUGOA	**−9.99e+122**	**−3.44e+92**	**0.00e+00**	**1.85e+122**	28.153 7
F_8	GOA	9.83e+01	1.87e+02	7.02e+02	2.19e+01	52.867 6
	GOA1	3.58e+01	9.23e+01	1.96e+01	1.53e+01	51.143 6
	GOA2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	**26.479 2**
	GOA3	0.00e+00	0.00e+00	0.00e+00	0.00e+00	50.937 0
	HCUGOA	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	28.110 2
F_9	GOA	1.16e+00	3.40e+00	7.02e+02	4.88e-01	53.053 1
	GOA1	7.99e-15	1.16e+00	1.96e+01	5.32e-01	50.983 1
	GOA2	8.88e-16	8.88e-16	0.00e+00	1.00e-31	**26.231 4**
	GOA3	8.88e-16	8.88e-16	0.00e+00	1.00e-31	50.844 7
	HCUGOA	**8.88e-16**	**8.88e-16**	**0.00e+00**	**1.00e-31**	28.033 9
F_{10}	GOA	1.67e+00	9.61e+00	7.02e+02	2.27e+00	53.019 2
	GOA1	8.86e-01	1.10e+00	1.96e+01	4.40e-02	50.9324
	GOA2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	**26.238 5**
	GOA3	0.00e+00	0.00e+00	0.00e+00	0.00e+00	50.821 5
	HCUGOA	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	28.202 6
F_{11}	GOA	6.11e+00	1.73e+01	7.02e+02	3.44e+00	43.111 2
	GOA1	2.36e+00	7.70e+00	1.96e+01	1.63e+00	41.519 7
	GOA2	3.80e-01	9.97e-01	0.00e+00	1.55e-01	**21.206 2**
	GOA3	9.36e-05	**3.51e-04**	0.00e+00	**6.16e-05**	41.824 3
	HCUGOA	**2.66e-06**	4.50e-03	**0.00e+00**	1.51e-03	23.809 1

西算子和分段思想的更新策略以及融合柯西变异和反向学习策略可有效提高算法寻优能力. GOA1对函数的寻优结果是3种改进策略算法中最差的,但其收敛精度相对GOA也得到了一定提升,尤其是F_6和F_9,得到寻优精度更高的结果. 表3的评价指标表明,在单峰和多峰函数中,融合3个改进策略的HCUGOA对于增加种群多样性、有效提高算法收敛精度是有效的.

另外,从运行一次算法的平均耗时来看:相同维度下,结合柯西算子和分段思想的算法GOA2耗时最短,说明在GOA中加入策略2之后,在不同的时期采用不同的更新公式有效增强了算法的全局探索能力;加入柯西变异和反向学习的算法GOA3与加入均匀分布的策略算法GOA1的耗时无显著差别,但均比基本GOA耗时短,表明了另外两种改进策略的有效性;另外,HCUGOA的耗时比GOA2稍长,这是因为在GOA中加入了3种改进策略之后,找到的解更多,导致寻优时间变长,即耗时变长.

为了从多样性的角度分析改进策略的有效性,图2给出了GOA和改进策略算法以及HCUGOA的部分函数独立运行30次的平均收敛曲线,各算法的图例如图2(a)所示. 从图2可明显看出,无论是针对单峰函数还是多峰函数,HCUGOA都具有更高的收敛速度和收敛精度. 具体来说:图2(a)~(e)为单峰函数平均收敛曲线,可以看出GOA2和GOA3具有更高的收敛精度和更快的寻优速度,GOA1是3个改进策略里寻优效果较差的,但其寻优精度也比GOA的高;图2(f)~(h)为多峰函数平均收敛曲线,从图可知,3种改进策略算法的平均收敛曲线均位于GOA收敛曲线下方,且下降速度快,解的分布范围广,使改进策略算法在相同的迭代次数下具有更高的收敛精度,在相同的收敛精度下具有更快的收敛速度.

综上所述,表3和图2的结论一致表明了3种改进策略的有效性.

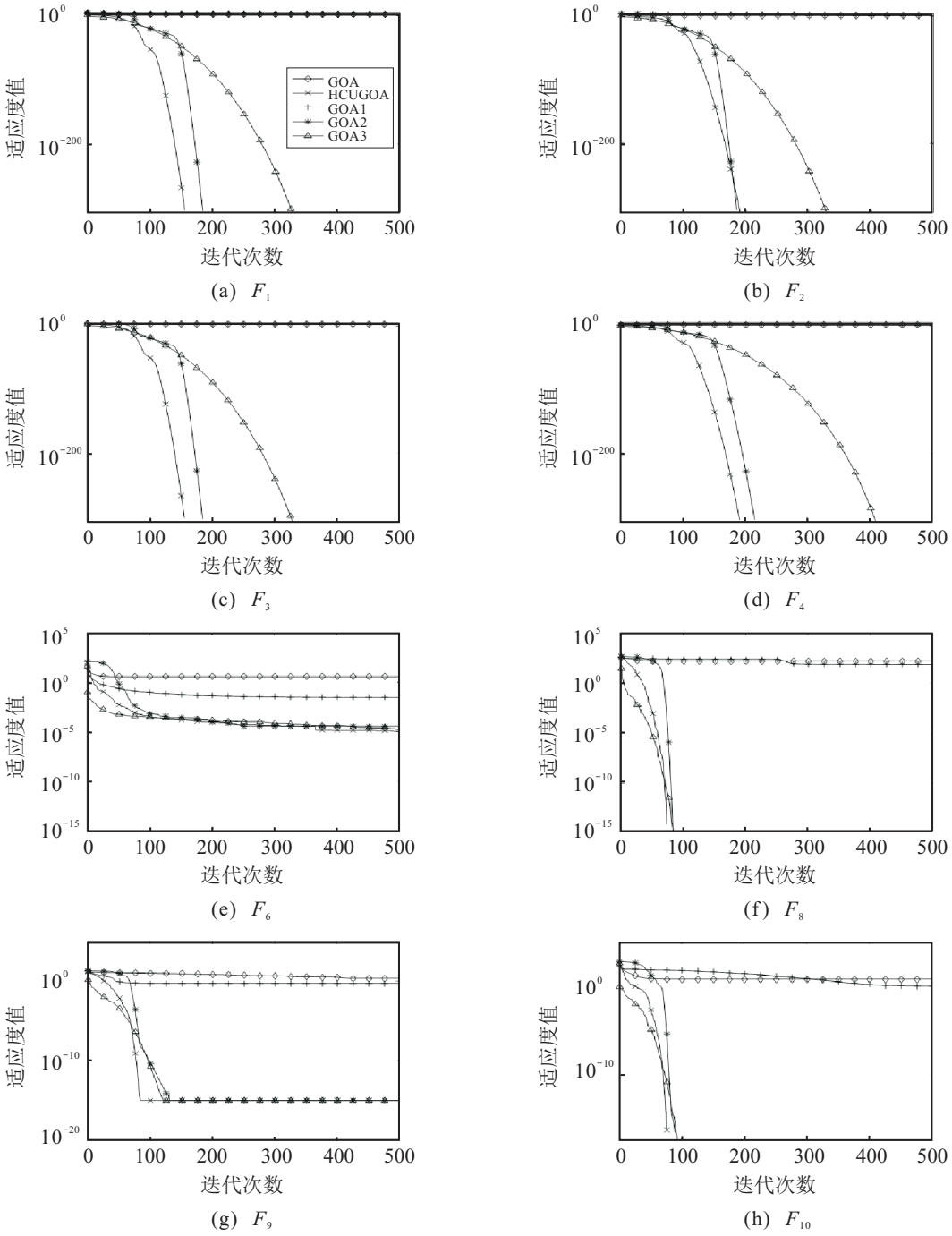

图 2 不同策略算法的平均收敛曲线

3.3 与最新的群智能算法比较

将混合策略改进鲸鱼算法MS-WOA[5]、正弦余弦改进算法m-SCA[8]、最新的蝗虫改进算法IGOA[14]及MGOA[15]与HCUGOA进行对比,分别在空间维度为10维、30维、200维的情况下对12个测试函数进行寻优,独立运行30次的结果如表4和表5所示(其中“—”表示参考文献未给出相应数据).

由表4和表5可知,总体上m-SCA寻优结果相对较差,IGOA和MGOA寻优精度相差不大,MS-WOA寻优精度是4种对比算法中最高的,但HCUGOA收敛精度和稳定性明显优于4种对比算法且耗时最短.

纵向观察数据:MS-WOA只在对F_1、F_8和F_{10}求解时才寻到理论值,但由标准差可知MS-WOA寻到理论值不具稳定性;对于$F_5 \sim F_7$和$F_9 \sim F_{12}$,在4种对比算法求解精度低或无法求解情况下,HCUGOA仍具较高寻优精度和稳定性. 其中:MGOA寻优结果是3种算法中较差的;IGOA次之,且IGOA在对F_8和F_{10}求解时,寻到理论最优值0;HCUGOA对12个测试函数均取得较好结果,尤其是$F_1 \sim F_4$、F_8和F_{10},寻到理论值. 横向观察数据,当维度从10维到30维

表4　各群智能算法在不同维度下的结果比较

函数	算法	dim = 10			dim = 30			dim = 200		
		平均值	标准差	耗时/s	平均值	标准差	耗时/s	平均值	标准差	耗时/s
F_1	MS-WOA[5]	0.00e+00	0.00e+00	—	0.00e+00	0.00e+00	—	0.00e+00	0.00e+00	—
	m-SCA[8]	—	—	—	5.70e-03	2.63e- 02	—	—	—	—
	IGOA	5.23e-17	5.76e-11	26.142 3	1.68e-17	3.28e-17	55.776 5	2.18e+01	8.04e+02	130.917
	MGOA	2.05e-92	4.85e-77	25.624 6	1.06e-12	2.15e-12	55.342 7	4.48e-84	3.10e-74	129.391
	HCUGOA	**0.00e+00**	**0.00e+00**	**14.767 9**	**0.00e+00**	**0.00e+00**	**28.733 5**	**0.00e+00**	**0.00e+00**	**69.216 5**
F_2	MS-WOA[5]	0.00e+00	0.00e+00	—	1.31e-180	0.00e+00	—	6.59e-175	0.00e+00	—
	m-SCA[8]	—	—	—	9.11e-04	1.90e-03	—	—	—	—
	IGOA	1.20e-13	2.21e-09	26.633 2	5.32e-10	6.84e-10	55.826 6	7.46e-03	6.05e-01	139.757 9
	MGOA	5.66e-58	1.69e-52	25.962 8	2.76e-07	2.80e-07	55.271 1	1.34e-56	2.39e-49	137.905 8
	HCUGOA	**0.00e+00**	**0.00e+00**	**14.669 8**	**0.00e+00**	**0.00e+00**	**29.275 3**	**0.00e+00**	**0.00e+00**	**73.867 1**
F_3	MS-WOA[5]	0.00e+00	0.00e+00	—	5.63e-322	0.00e+00	—	4.74e-312	0.00e+00	—
	m-SCA[8]	—	—	—	8.48e+02	5.49e+02	—	—	—	—
	IGOA	5.84e-08	7.97e-02	27.399 5	1.98e-16	5.11e-16	45.811 1	1.97e+04	1.52e+04	142.447 4
	MGOA	3.18e-02	3.20e+02	26.244 7	1.80e-12	3.66e-12	45.132 9	1.31e+05	4.18e+04	140.484 3
	HCUGOA	**0.00e+00**	**0.00e+00**	**15.076 4**	**0.00e+00**	**0.00e+00**	**28.470 0**	**0.00e+00**	**0.00e+00**	**75.814 3**
F_4	MS-WOA[5]	0.00e+00	0.00e+00	—	9.64e-170	0.00e+00	—	1.12e-166	0.00e+00	—
	m-SCA[8]	—	—	—	7.07e-01	5.37e-01	—	—	—	—
	IGOA	1.60e-06	1.02e-03	16.947 6	2.85e-09	3.53e-09	45.203 0	5.66e+01	6.59e+00	140.080 4
	MGOA	2.02e-01	8.53e+00	16.829 1	5.88e-07	8.29e-07	44.101 8	1.20e-03	2.82e+01	138.726 2
	HCUGOA	**0.00e+00**	**0.00e+00**	**11.737 1**	**0.00e+00**	**0.00e+00**	**28.049 6**	**0.00e+00**	**0.00e+00**	**73.396 2**
F_5	MS-WOA[5]	—	—	—	2.73e+01	3.64e-01	—	—	—	—
	m-SCA[8]	—	—	—	2.95e+01	2.43e+00	—	—	—	—
	IGOA	6.78e+00	4.39e-01	17.629 4	7.57e+00	3.33e-01	47.246 0	2.09e+05	7.93e+06	138.221 1
	MGOA	5.86e+00	4.16e-01	16.465 3	6.82e+00	1.31e+00	47.916 0	4.74e+01	3.56e-01	139.023 3
	HCUGOA	**2.04e-02**	**1.97e+00**	**14.649 1**	**1.78e-03**	**7.26e+00**	**27.700 6**	**9.71e-04**	**8.65e+00**	**75.547 5**
F_6	MS-WOA[5]	4.77e-05	512e-05	—	1.05e-04	7.57e-05	—	1.58e-04	1.44e-04	—
	m-SCA[8]	—	—	—	1.95e-02	6.87e-03	—	—	—	—
	IGOA	3.08e-04	1.62e-03	18.142 6	9.06e-05	9.35e-05	45.909 0	4.39e-01	4.36e+00	141.562 2
	MGOA	6.94e-06	2.95e-03	18.937 7	9.16e-05	9.14e-05	45.623 9	1.06e-04	4.56e-03	139.583 2
	HCUGOA	**5.49e-08**	**9.74e-06**	**14.674 2**	**4.96e-07**	**6.64e-06**	**28.033 9**	**3.42e-07**	**7.22e-06**	**73.377 3**
F_7	MS-WOA[5]	—	—	—	−1.26e+04	1.55e+01	—	—	—	—
	m-SCA[8]	—	—	—	−4.26e+02	2.88e+02	—	—	—	—
	IGOA	−2.48e+03	1.63e+02	26.803 7	−2.31e+03	2.17e+02	51.586 2	−6.11e+03	4.14e+02	136.171 9
	MGOA	−4.19e+03	6.35e+02	26.643 5	−7.98e+03	2.23e+03	51.558 5	−2.09e+04	3.00e+03	137.357 5
	HCUGOA	**−1.12e+128**	**2.24e+127**	**14.811 3**	**−9.99e+122**	**1.85e+122**	**27.978 9**	**−5.76e+125**	**1.05e+125**	**73.617 2**
F_8	MS-WOA[5]	0.00e+00	0.00e+00	—	0.00e+00	0.00e+00	—	0.00e+00	0.00e+00	—
	m-SCA[8]	—	—	—	7.81e+01	5.21e+01	—	—	—	—
	IGOA	0.00e+00	5.20e+00	18.071 8	0.00e+00	0.00e+00	52.886 8	6.23e+00	6.10e+01	139.894 4
	MGOA	0.00e+00	0.00e+00	19.042 2	8.57e-14	1.08e-13	52.347 8	0.00e+00	0.00e+00	138.389 1
	HCUGOA	**0.00e+00**	**0.00e+00**	**14.823 8**	**0.00e+00**	**0.00e+00**	**28.110 2**	**0.00e+00**	**0.00e+00**	**73.403 3**
F_9	MS-WOA[5]	8.88e-16	0.00e+00	—	8.88e-16	0.00e+00	—	8.88e-16	0.00e+00	—
	m-SCA[8]	—	—	—	—	—	—	—	—	—
	IGOA	1.76e-10	1.21e+00	26.621 8	2.19e-09	4.83e-09	47.890 5	5.88e-04	9.18e+00	141.231 5
	MGOA	8.88e-16	2.65e-15	26.205 3	2.94e-07	2.09e-07	47.445 3	8.88e-16	2.87e-15	139.562 6
	HCUGOA	**8.88e-16**	**1.00e-31**	**16.411 9**	**8.88e-16**	**1.00e-31**	**28.033 9**	**8.88e-16**	**1.00e-31**	**74.768 2**
F_{10}	MS-WOA[5]	0.00e+00	0.00e+00	—	0.00e+00	0.00e+00	—	0.00e+00	0.00e+00	—
	m-SCA[8]	—	—	—	3.84e-02	7.15e-02	—	—	—	—
	IGOA	1.45e-12	9.50e-02	27.988 9	0.00e+00	0.00e+00	56.121 1	1.01e+00	5.34e+00	140.650 4
	MGOA	0.00e+00	2.03e-01	26.388 3	1.12e-11	2.22e-11	55.409 7	0.00e+00	5.19e-02	138.625 3
	HCUGOA	**0.00e+00**	**0.00e+00**	**14.883 4**	**0.00e+00**	**0.00e+00**	**28.202 6**	**0.00e+00**	**0.00e+00**	**73.995 7**
F_{11}	MS-WOA[5]	—	—	—	1.15e-02	5.43e-02	—	5.84e+00	4.29e+00	—
	m-SCA[8]	—	—	—	1.45e-01	8.19e-02	—	—	—	—
	IGOA	4.95e-02	5.41e-02	26.968 1	7.05e-03	2.10e-03	47.048 9	5.99e+00	1.48e+07	102.698 3
	MGOA	2.41e-03	3.06e-01	26.166 8	9.99e-06	4.65e-06	46.719 5	1.25e-02	1.25e-02	101.220 5
	HCUGOA	**1.73e-06**	**6.52e-06**	**15.175 7**	**2.66e-06**	**1.51e-03**	**23.809 1**	**3.57e-07**	**7.00e-04**	**55.063 1**

表5 各群智能算法在固定低维(dim = 4)下的结果比较

函数	算法	平均值	标准差	耗时/s
	MS-WOA[5]	4.06e-04	1.05e-04	—
	m-SCA[8]	5.10e-04	1.00e-04	—
F_{12}	IGOA	3.21e-04	5.19e-06	10.875 9
	MGOA	3.08e-04	6.07e-07	10.251 3
	HCUGOA	**3.07e-04**	**5.86e-07**	**7.233 2**

再到200维,算法对求解精度和鲁棒性均有所下降,这是因为维度增加使函数更复杂,寻优时需做更多调整. 但相对于4种对比算法,HCUGOA求解精度仍最高,从而验证了HCUGOA在低维和高维情况下均有更好的求解能力;另外,从平均耗时来看:相同维度下,HCUGOA相对于其他两种GOA算法的平均耗时最短;不同维度下,随着维度的增加,3种算法的平均运行时间都变长,这是因为从低维到高维,搜索空间增大,寻优难度也呈指数增加.

3.4 Wilcoxon秩和检验

上述仿真中,均未对每次结果进行独立比较.为体现算法的稳定性和公平性,本文采用Wilcoxon秩和检验来验证本文改进算法每次运行结果是否在统计上与其他算法存在显著性差异.秩和检验在5%的显著性水平下进行:p值小于5%时,拒绝H0假设,说明两种对比算法具显著性差异;否则接受H0假设,说明两种算法寻优结果在整体上是相同的[18].

表6给出了测试函数下GOA、IGOA、MGOA、GOA1、GOA2和GOA3共6种算法与HCUGOA的秩和检验中计算的p值,因最佳算法不能与自身进行比较,故表6中标记为“Na”表示不适用,即无法进行显著性判断,“R”为显著性判断结果,“+”、“−”、“=”分别表示HCUGOA性能优于、劣于和相当于对比算法. 从表6中可以看出大部分p值远小于5%,“R”为“+”,拒绝零假设. 因此,总体上HCUGOA与其他6种算法之间具显著性差异,且HCUGOA显著更优.

表6 Wilcoxon秩和检验p值

函数	GOA		IGOA		MGOA		GOA1		GOA2		GOA3	
	p	R	p	R	p	R	p	R	p	R	p	R
F_1	1.21e-12	+	1.21e-12	+	1.21e-12	+	1.21e-12	+	Na	=	Na	=
F_2	1.21e-12	+	1.21e-12	+	1.21e-12	+	1.21e-12	+	Na	=	Na	=
F_3	1.21e-12	+	1.21e-12	+	1.21e-12	+	1.21e-12	+	Na	=	Na	=
F_4	1.21e-12	+	1.21e-12	+	1.21e-12	+	1.21e-12	+	Na	=	Na	=
F_5	3.02e-11	+	3.02e-11	+	3.02e-11	+	3.02e-11	+	3.02e-11	+	3.02e-11	+
F_6	3.02e-11	+	3.02e-11	+	3.02e-11	+	3.02e-11	+	8.29e-06	+	1.15e-01	−
F_7	3.02e-11	+	3.02e-11	+	3.02e-11	+	3.02e-11	+	3.02e-11	+	3.02e-11	+
F_8	1.21e-12	+	1.61e-01	−	1.21e-12	+	1.21e-12	+	Na	=	Na	=
F_9	1.21e-12	+	1.16e-08	+	1.21e-12	+	1.16e-12	+	Na	=	Na	=
F_{10}	1.21e-12	+	1.61e-01	−	1.21e-12	+	1.21e-12	+	Na	=	Na	=
F_{11}	3.02e-11	+	3.02e-11	+	3.02e-11	+	3.02e-11	+	3.02e-11	+	9.21e-05	+

3.5 在CEC 2014基准函数上进行测试

为了进一步验证HCUGOA的有效性和鲁棒性,本文在CEC2014单目标优化函数中选取部分单峰、多峰、混合和复合类型的函数进行优化求解. 对选取的部分函数的总结如表7所示,本实验种群规模为30,最大迭代次数为1 000,维度dim = 30.

表7 CEC2014函数(部分)

函数	维度	特征	定义域	最佳值
CEC 01	30	UN	[−100,100]	100
CEC 05	30	MN	[−100,100]	500
CEC 17	30	HF	[−100,100]	1 700
CEC 18	30	HF	[−100,100]	1 800
CEC 24	30	CF	[−100,100]	2 400
CEC 25	30	CF	[−100,100]	2 500

CEC2014函数因具有复杂的特征,常用来验证算法的鲁棒性. 将本文算法与3个改进策略算法、2个GOA变体以及标准PSO[4]和典型DE算法进行比较. 其中:因L-SHADE[19]在CEC 2014函数中的出色表现,常作为比较算法;另外,PSO和L-SHADE已在参考文献中对CEC 2014函数进行了测试. CEC 2014中部分函数独立运行30次的结果如表8所示.

由表8可知:在单峰函数上,L-SHADE具有明显优势,而PSO与HCUGOA性能相似,例如对于

表8　CEC 2014优化结果比较

算法	指标	CEC 01	CEC 05	CEC 17	CEC 18	CEC 24	CEC 25
HCUGOA	mean	1.214 2e+06	5.208 8e+02	5.208 8e+02	2.848 0e+03	2.600 2e+03	2.707 1e+03
	std	4.989 8e+05	5.202 1e+02	5.202 1e+02	3.746 4e+02	0.000 0e+00	0.000 0e+00
GOA1	mean	5.881 9e+08	5.228 4e+02	5.228 4e+02	9.900 0e+05	2.132 5e+03	2.742 6e+03
	std	8.343 7e+07	5.219 2e+02	5.219 2e+02	6.837 4e+04	6.200 0e+00	2.630 1e+00
GOA2	mean	3.502 2e+06	5.208 4e+02	5.208 4e+02	8.130 4e+03	2.615 6e+03	2.720 1e+03
	std	6.954 9e+05	5.209 2e+02	5.209 2e+02	3.747 6e+03	0.000 0e+00	0.000 0e+00
GOA3	mean	2.216 6e+07	5.210 5e+02	5.210 5e+02	9.132 0e+04	2.415 9e+03	2.608 5e+03
	std	2.537 7e+06	5.208 9e+02	5.208 9e+02	7.009 8e+03	7.616 0e+00	0.000 0e+00
IGOA	mean	4.715 0e+08	5.223 4e+02	5.223 4e+02	2.070 5e+08	2.910 0e+03	2.742 1e+03
	std	3.878 4e+08	5.218 3e+02	5.218 3e+02	4.319 4e+08	6.615 6e+00	5.624 5e+01
MGOA	mean	7.831 9e+06	5.210 1e+02	5.210 1e+02	3.036 0e+07	2.523 1e+03	2.726 0e+03
	std	5.631 4e+06	5.206 9e+02	5.206 9e+02	2.188 6e+07	4.616 2e+00	2.640 4e+00
PSO[4]	mean	6.69e+06	2.09e+01	2.09e+01	3.97e+03	2.30e+02	2.09e+02
	std	8.99e+06	8.52e-02	8.52e-02	4.62e+03	6.64e+00	1.64e+00
L-SHADE[19]	mean	1.42e-14	2.01e+01	2.01e+01	6.58e+00	2.24e+02	2.03e+02
	std	3.61e-15	1.70e-02	1.70e-02	2.74e+00	1.15e+00	4.97e+02

CEC 01, L-SHADE寻优精度是e-14, 而HCUGOA和其他GOA变体的最优精度是e+05; 在多峰问题上, 6个GOA变体算法表现非常相似; 在混合和复合函数中, 除L-SHADE之外, HCUGOA的寻优结果比对比算法更接近理论值, 且比PSO更稳定, 验证了HCUGOA具有良好的有效性和稳定性. 此外, 从实验数据可以看出, L-SHADE与GOA变体相比具有明显的优势, HCUGOA以及其他GOA算法均无法达到L-SHADE的水平, 这可能是因为差分进化算法本身在寻优过程中比GOA更具优势的原因.

4　结　论

为改善GOA缺陷, 本文受柯西变异和PSO的启发, 提出了新的位置更新方式以增加种群多样性, 增强全局探索能力; 在柯西变异和反向学习的基础上, 对目标值进行变异更新, 使算法避免陷入局部最优; 为了更好地平衡全局探索与局部开发, 将均匀分布函数引入控制参数c, 构建新函数. 5组实验显示: HCUGOA对基准测试函数的求解结果优于MS-WOA、m-SCA等群智能算法, 与新改进的蝗虫算法IGOA和MGOA相比也具有竞争性; Wilcoxon秩和检验验证了本文算法具有更优的显著性差异; 通过对CEC 2014部分函数求解结果也表明了本文算法的有效性. 在后续研究中, 考虑将改进的蝗虫优化算法应用到WSN路由能耗寻优问题中, 找到能耗最小的广播路径, 以进一步验证算法的性能.

参考文献(References)

[1] Saremi S, Mirjalili S, Lewis A. Grasshopper optimization algorithm: Theory and application[J]. Advances in Engineering Software, 2017, 105: 30-47.

[2] Kapilevich V, Seno S, Matsuda H, et al. Chromatin 3D reconstruction from chromosomal contacts using a genetic algorithm[J]. ACM Transactions on Computational Biology and Bioinformatics, 2019, 16(5): 1620-1626.

[3] Jayaprakash A, KeziSelvaVijila C. Feature selection using ant colony optimization (ACO) and road sign detection and recognition (RSDR) system[J]. Cognitive Systems Research, 2019, 58: 123-133.

[4] 徐桂萍. 基于维度学习策略的粒子群算法的研究与应用[D]. 长春: 吉林大学, 2019.
(Xu G P. Study on particle swarm optimization based with dimensional learning strategy[D]. Changchun: Jilin University, 2019.)

[5] 何庆, 魏康园, 徐钦帅. 基于混合策略改进的鲸鱼优化算法[J]. 计算机应用研究, 2019, 36(12): 3647-3651.
(He Q, Wei K Y, Xu Q S. Mixed strategy based improved whale optimization algorithm[J]. Application Research of Computers, 2019, 36(12): 3647-3651.)

[6] 徐钦帅, 何庆, 魏康园. 改进蚁狮算法的无线传感器网络覆盖优化[J]. 传感技术学报, 2019, 32(2): 266-275.
(Xu Q S, He Q, Wei K Y. Wireless sensor network coverage optimization based on improved ant lion algorithm[J]. Journal of Transduction Technology, 2019, 32(2): 266-275.)

[7] 张达敏, 陈忠云, 辛梓芸, 等. 基于疯狂自适应的樽海鞘群算法[J]. 控制与决策, 2020, 35(9): 2112-2120.
(Zhang D M, Chen Z Y, Xin Z Y, et al. Salpswarm algorithm based on craziness and adaptive[J]. Control and Decision, 2020, 35(9): 2112-2120.)

[8] Gupta S, Deep K. A hybrid self-adaptive sine cosine algorithm with opposition based learning[J]. Expert Systems with Applications, 2019, 119: 210-230.

[9] 闫旭, 叶春明. 混合蝗虫优化算法求解作业车间调度问题[J]. 计算机工程与应用, 2019, 55(6): 257-264.
(Yan X, Ye C M. Hybrid grasshopper optimization algorithm for job shop scheduling problem[J]. Computer Engineering and Applications, 2019, 55(6): 257-264.)

[10] 黄超, 梁圣涛, 张毅, 等. 基于多目标蝗虫优化算法的移动机器人路径规划[J]. 计算机应用, 2019, 39(10): 2859-2864.
(Huang C, Liang S T, Zhang Y, et al. Mobile robot path planning based on multi-target grasshopper optimi algorithm[J]. Computer Application, 2019, 39(10): 2859-2864.)

[11] 崔鹤, 薛媛. 基于GOA-SVR的在线英语学习人数预测[J]. 微型电脑应用, 2019, 35(5): 128-131.
(Cui H, Xue Y. Prediction of online English learning population based on GOA-SVR[J]. Microcomputer Applications, 2019, 35(5): 128-131.)

[12] 潘峰, 孙红霞. 基于蝗虫算法的图像多阈值分割方法[J]. 电子测量与仪器学报, 2019, 33(1): 149-155.
(Pan F, Sun H X. Image multi-threshold segmentation method based on grasshopper optimization algorithm[J]. Journal of Electronic Measurement and Instrument, 2019, 33(1): 149-155.)

[13] Zhao R, Ni H, Feng H N, et al. A dynamic weight grasshopper optimization algorithm with random jumping[C]. Advances in Intelligent Systems and Computing. Singapore: Springer, 2019: 401-413.

[14] Luo J, Chen H, Zhang Q, et al. An improved grasshopper optimization algorithm with application to financial stress prediction[J]. Applied Mathematical Modelling, 2018, 64: 654-668.

[15] Taher M A, Kamel M S, Jurado F, et al. Modified grasshopper optimization framework for optimal power flow solution[J]. Electrical Engineering, 2019, 101(1): 121-148.

[16] 李洋州, 顾磊. 一种基于曲线自适应和模拟退火的蝗虫优化算法[J]. 计算机应用研究, 2019, 36(12): 3637-3643.
(Li Y Z, Gu L. A grasshopper optimization algorithm based on curve adaptive and simulated annealing[J]. Computer Application Research, 2019, 36(12): 3637-3643.)

[17] Tizhoosh H R. Opposition-based learning: A new scheme for machine intelligence[C]. Proceedings of Computational Intelligence for Modelling. Vienna: Computer Society, 2005, 1: 695-701.

[18] Derrac J, Garcia S, Molina D, et al. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms[J]. Swarm and Evolutionary Computation, 2011, 1(1): 3-18.

[19] Tanabe R, Fukunaga A S. Improving the search performance of SHADE using linear population size reduction[C]. 2014 IEEE Congress on Evolutionary Computation (CEC). Beijing: IEEE, 2014: 1658-1665.

作者简介

何庆 (1982−), 男, 副教授, 博士, 从事大数据应用、进化计算等研究, E-mail: qhe@gzu.edu.cn;

林杰 (1995−), 女, 硕士生, 从事进化计算、数据挖掘的研究, E-mail: 1376135318@qq.com;

徐航 (1995−), 男, 硕士生, 从事认知无线电的研究, E-mail: 1518818139@qq.com.

(责任编辑：李君玲)