

# 带全局-局部最优步长比例因子的布谷鸟搜索算法

梁毛毛<sup>1,2</sup>, 肖文<sup>1,2</sup>, 王李进<sup>1,2</sup>, 钟一文<sup>1,2</sup>

(1. 福建农林大学计算机与信息学院, 福建 福州 350002)

(2. 福建农林大学智慧农林福建省高校重点实验室, 福建 福州 350002)

**[摘要]** 布谷鸟搜索算法利用 Lévy Flights 随机走动和 Biased 随机走动过程完成全局搜索和局部开发. 针对原始的 Lévy Flights 随机走动仅采用固定的常数步长因子, 介绍了一种使用每一代中个体的全局和局部最优适应值动态设置步长因子的方法, 并提出了一种带全局-局部最优步长比例因子的布谷鸟搜索算法. 在测试函数上的运行结果证明, 该方法是可行的, 且能够全面有效地加强布谷鸟搜索算法的收敛速度和求精能力, 其性能总体上比采用固定因子、基于均匀分布随机数或基于贝塔分布随机数比例因子的布谷鸟搜索算法更优.

**[关键词]** 布谷鸟搜索算法, 全局-局部最优适应值, 比例因子, 可变因子, 函数优化

**[中图分类号]** O643; X703 **[文献标志码]** A **[文章编号]** 1672-1292(2022)02-0056-07

## Cuckoo Search Algorithm with Global-Local Best Scaling Factor

Liang Maomao<sup>1,2</sup>, Xiao Wen<sup>1,2</sup>, Wang Lijin<sup>1,2</sup>, Zhong Yiwen<sup>1,2</sup>

(1. College of Computer and Information Sciences, Fujian Agriculture and Forestry University, Fuzhou 350002, China)

(2. Key Laboratory of Smart Agriculture and Forestry, Fujian Agriculture and Forestry University, Fuzhou 350002, China)

**Abstract:** Lévy Flights random walk and Biased/selective random walk is employed in cuckoo search algorithm to search for new solutions. Instead of a fixed scaling factor in standard Lévy Flights, in our study, global and local best fitness of individuals in each generation are utilized to dynamically define the scaling factor. And a modified cuckoo search algorithm, called GbestCS is proposed. Comprehensive experiments demonstrate that this strategy is feasible, and it can effectively strengthen the convergence speed and improve accuracy of cuckoo search algorithm. In addition, the performance of the proposed algorithm is generally better than that of cuckoo search algorithm that uses a constant scaling factor, either based on uniformly distributed random numbers or based on a beta distribution random numbers.

**Key words:** cuckoo search algorithm, global-local best fitness, scaling factor, varied factor, function optimization

元启发式算法是一种基于经验的非确定性算法, 在处理结构不明确、缺少条件或大规模的优化问题上具有优势, 在环形电感建模<sup>[1]</sup>、优化配电网网架<sup>[2]</sup>和储能配置<sup>[3]</sup>、求取自由曲面最大主曲率<sup>[4]</sup>等方面有广泛应用. 布谷鸟搜索(cuckoo search, CS)算法<sup>[5-6]</sup>是其中的一种自然启发式算法, 起源于布谷鸟寻巢产卵并让其他鸟为其孵卵的行为. CS 算法利用 Lévy Flights 随机走动和 Biased 随机走动迭代地发现新的个体.

为了得到性能更好的 CS 算法, 部分学者结合其他优化方法. Wang 等<sup>[7]</sup>将和声搜索与 CS 算法混合, 提出 HS/CS 算法; Rosli 等<sup>[8]</sup>结合正余弦算法, 提出一种混合布谷鸟搜索算法(HMSCACSA); Shehab 等<sup>[9-10]</sup>不仅提出与爬山算法结合的稳定的布谷鸟搜索算法(CSAHS), 且引入蝙蝠算法来求解全局数值优化; 还有研究<sup>[11-12]</sup>将合作协同进化框架、单位位置继承等技术引入 CS 算法. 部分学者致力于改进 Lévy Flights 和 Biased 随机走动的过程. Wang 等<sup>[13]</sup>提出采用均匀分布的随机数动态设置步长因子的布谷鸟搜索算法(VCS); 王李进等<sup>[14-15]</sup>分别采用逐维搜索和正交交叉操作加强了 Biased 随机走动搜索效率. 为了改善 CS 算法在求解复杂问题时的性能退化, Salgotra 等<sup>[16]</sup>提出了自适应的 CS 算法; Wei 等<sup>[17]</sup>提出一种基于柯西分布和莱默平均值更新 Biased 随机走动中的变异算子的方法, 并命名为 CSAPC; Wang 等<sup>[18]</sup>利

收稿日期: 2021-08-31.

基金项目: 福建省自然科学基金项目(2021J01127)、福建农林大学科技创新专项基金项目(CXZX2020148C、CXZX2020150C)、数字福建旅游大数据研究所开放基金项目(DFJTBDR12020103)..

通讯作者: 王李进, 博士, 教授, 研究方向: 仿生优化算法及其应用. E-mail: lijnawang@fafu.edu.cn

用指数曲线模拟步长因子和发现概率的变化趋势,提出基于指数函数的 CS 算法;Chen 等<sup>[19]</sup>在 Lévy Flights 随机走动中采用基于最优解逐维更新个体的策略,提出逐维增强的 CS 算法。

Lévy Flights 随机走动中的步长比例因子是 CS 算法中的重要控制参数,能够影响算法的搜索能力并调整算法的收敛速度。在标准的 CS 算法中,步长比例因子为固定的常数,算法存在收敛速度慢、求精能力不足等缺陷,因而有很多学者围绕该点展开了改进。林要华等<sup>[20]</sup>引入贝塔分布动态设置步长因子;Walton 等<sup>[21]</sup>提出利用迭代次数更新步长因子;Valian 等<sup>[22]</sup>使用迭代次数构建指数函数更新步长因子;Layeb 等<sup>[23]</sup>将量子计算的概念应用在步长因子上。然而,以上优化步长比例因子的研究均未利用到种群中个体本身的条件。本文采用一种基于个体适应值的步长因子更新策略,提出带全局-局部最优步长比例因子的布谷鸟搜索算法(cuckoo search algorithm with global-local best scaling factor, GLbestCS)。

从仿真实验的结果可以看出, GLbestCS 可行且能有效改善 CS 算法的求精能力和收敛速度,其性能总体上比采用固定常数、均匀分布的随机数以及贝塔分布的随机数步长因子的 CS 算法更优。

## 1 布谷鸟算法

2009 年, Yang 等<sup>[5-6]</sup>提出 CS 算法。初始化种群后, CS 算法迭代执行 Lévy Flights 随机走动和 Biased 随机走动来搜索新个体。每次随机走动结束, CS 算法都会以适应值为依据贪婪地选择较优个体来更新种群。

Lévy Flights 是一种从 Lévy 分布中获取步长的随机走动,主要作用为全局搜索。在第  $G$  代( $G>0$ ), Lévy Flights 随机走动可用以下公式表示:

$$X_{i,G+1} = X_{i,G} + \alpha_0 \frac{\phi \times \mu}{|\nu|^{1/\beta}} (X_{i,G} - X_{\text{best}}), \quad (1)$$

式中,  $\alpha_0$  为比例步长因子(通常  $\alpha_0 = 0.01$ );  $X_{\text{best}}$  表示目前发现的最优个体;  $\beta$  为常数,在 CS 算法中通常设置为 1.5;  $\mu$  和  $\nu$  是随机数,服从均值为 0、标准差为 1 的正态分布;  $\Gamma$  是一个伽马(Gama)函数,

$$\phi = \left[ \frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi \times \beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\frac{\beta-1}{2}}}\right]^{\frac{1}{\beta}}. \quad (2)$$

Biased 随机走动是一个局部随机游走过程,主要作用为局部开发,以当前个体为基础,根据在搜索池中随机选择的两个个体产生新个体。Biased 随机走动可用下式表示:

$$V_i^G = X_i^G + \alpha_s \otimes H(p_a - \varepsilon) \otimes (X_m^G - X_n^G), \quad (3)$$

式中,  $\otimes$  为乘法算子;  $m$  和  $n$  为随机下标;  $p_a$  为发现概率,其值为小数。

## 2 带全局-局部最优步长比例因子的布谷鸟搜索(GLbestCS)算法

### 2.1 全局-局部最优步长比例因子

步长比例因子  $\alpha_0$  是 CS 算法中的重要参数之一。当其值较大时,全局搜索能力较强,收敛速度较快,但算法的求精能力差;当其值较小时,局部搜索能力增强,收敛速度变慢,易陷入局部最优的情况。因此,如何设定步长比例因子决定了能否实现全局与局部搜索之间的有效平衡。

参考文献[24],本文引入种群中个体的信息更新 CS 算法中的步长比例因子,增加一个全局最优适应值和个体最优适应值的比值来调节算法的收敛度速度和求精能力之间的平衡,最终得到步长比例因子公式为:

$$\alpha_{0i} = k - \frac{f_{g\text{best}}}{f_{pi}}, \quad (4)$$

式中,  $\alpha_{0i}$  是第  $i$  代的步长比例因子;  $f_{g\text{best}}$  代表全局最优适应值;  $f_{pi}$  代表该个体的最优适应值;  $k$  为常数。当个体靠近最优情况,  $f_{g\text{best}}/f_{pi}$  减小,算法的寻优精度增强;当个体远离最优情况,  $f_{g\text{best}}/f_{pi}$  增大,算法的全局搜索速度加快。文献[10]中采用均匀分布的随机数动态设置步长因子,服从  $[0, 1]$  均匀分布的均值为 0.5。故在后续实验中令  $k = 0.5$ , 即:

$$\alpha_{0i} = 0.5 - \frac{g_{\text{best}_i}}{p_{\text{best}_i}}. \quad (5)$$

2.2 GLbestCS 算法

算法 1 GLbestCS 算法
输入:发现概率 $p$ 、种群大小 $NP$ ;
输出:最优个体 $X_{best}$ .
1. 初始化种群,利用适应值发现最优个体;
2. 当迭代终止条件未满足 ( $FES < MaxFES$ ):
For $i = 1$ to $NP$ ;
$\alpha_{0i} \leftarrow$ 采用式(5)动态设置 Lévy Flights 随机走动的步长比例因子 $\alpha_{0i}$ ;
根据式(1)生成新的个体 $V_i$ ,若 $V_i$ 的适应值优于 $X_i$ ,则用 $V_i$ 更新 $X_i$ ;
End for;
For $i = 1$ to $NP$ ;
根据式(3)生成候选个体 $V_i$ ,若 $V_i$ 的适应值优于 $X_i$ ,则用 $V_i$ 更新 $X_i$ ;
End for;
更新当前最优个体;
3. 输出最优个体.

与 CS 算法相比, GLbestCS 算法未引入新的操作, 仅用适应值更新步长比例因子  $\alpha_{0i}$ , 因此认为 GLbestCS 与 CS 时间复杂度相近. 同时, GLbestCS 算法的性能显著更优, 即更有效率. 因此, 本文认为, GLbestCS 算法未增加算法整体的时间复杂度.

3 仿真结果

本文利用测试函数来评估 GLbestCS 算法的性能. 选择 Noman 等<sup>[25]</sup>提出的 20 个基准函数, 在每个测试函数上独立运行 25 次. 为了使误差最小化, 限制最大适应值评估次数 ( $MaxFES$ ) 为  $10\ 000 \times D$  ( $D$  为函数维度); 发现概率  $p_a$  的值取 0.25; 当  $D = 10$  时, 种群规模  $NP$  为 30; 当  $D$  取其他值时, 令  $NP = D$ .

3.1 性能分析

表 1 所列当  $D = 30$  时, CS 和 GLbestCS 算法在所有测试函数上的表现. “ $AVG_{Er} \pm SD_{Er}$ ”代表平均适应值误差和标准方差. 最后一行是基于  $\alpha = 0.05$  的 Wilcoxon 符号秩检验的统计结果, “+”代表 GLbestCS 算法的表现显著优于 CS 算法, “-”相反, “=”为两者不存在明显差异.

表 1 CS 和 GLbestCS 求解测试函数的平均适应值误差 ( $D = 30$ )  
Table 1 Average fitness error obtained by CS and GLbestCS for benchmark functions ( $D = 30$ )

Function	AVG <sub>Er</sub> ±SD <sub>Er</sub>		Function	AVG <sub>Er</sub> ±SD <sub>Er</sub>	
	CS	GLbestCS		CS	GLbestCS
$F_{sph}$	9.24e-31±1.17e-30	1.84e-57±2.20e-57	$F_1$	1.47e-29±4.96e-29	8.08e-30±4.04e-29
$F_{ros}$	1.18e+01±1.29e+01	6.19e+00±1.39e+01	$F_2$	6.64e-03±7.90e-03	6.28e-03±4.05e-03
$F_{ack}$	1.49e-01±3.48e-01	7.11e-15±0.00e+00	$F_3$	2.27e+06±5.53e+05	3.26e+06±1.13e+06
$F_{grw}$	2.96e-04±1.48e-03	0.00e+00±0.00e+00	$F_4$	1.82e+03±1.28e+03	3.78e+02±2.90e+02
$F_{ras}$	2.36e+01±4.42e+00	1.47e+01±6.08e+00	$F_5$	3.12e+03±9.86e+02	1.47e+03±7.89e+02
$F_{sch}$	1.49e+03±2.26e+02	5.39e+02±4.05e+02	$F_6$	2.44e+01±2.53e+01	9.01e+00±8.31e+00
$F_{sal}$	3.68e-01±6.25e-02	2.04e-01±2.00e-02	$F_7$	9.17e-04±2.05e-03	4.48e-03±8.97e-03
$F_{wht}$	3.73e+02±5.61e+01	2.50e+02±6.10e+01	$F_8$	2.09e+01±5.48e-02	2.09e+01±5.92e-02
$F_{pn1}$	6.75e-19±2.45e-18	1.57e-32±5.59e-48	$F_9$	2.73e+01±6.37e+00	1.71e+01±7.87e+00
$F_{pn2}$	3.50e-28±1.28e-27	1.35e-32±5.59e-48	$F_{10}$	1.64e+02±3.03e+01	1.09e+02±1.90e+01
			+/-/+		
			14/5/1		

由表 1 可以看出, 在函数  $F_{sph}$ 、 $F_{ros}$ 、 $F_{ack}$ 、 $F_{grw}$ 、 $F_{ras}$ 、 $F_{sch}$ 、 $F_{sal}$ 、 $F_{wht}$ 、 $F_{pn1}$ 、 $F_{pn2}$ 、 $F_1$ 、 $F_2$ 、 $F_4$ 、 $F_5$ 、 $F_6$ 、 $F_9$ 、 $F_{10}$  上, GLbestCS 算法的表现优于 CS 算法; 在函数  $F_3$ 、 $F_7$  上则相反; 仅在函数  $F_8$  上两者的表现近似. 同时, GLbestCS 算法在 14 个函数上表现更优, 在 5 个函数上与 CS 算法近似, 仅在 1 个函数上略差. 且 GLbestCS 算法在函数  $F_{grw}$  上能够取得全局最优解. 图 1 展示了 CS 和 GLbestCS 算法在测试函数中的一些有代表性的收敛过程. 可以看出, GLbestCS 算法在前期有更快的收敛速度、在后期有更强的求精能力.

因此, 可以认为, 用全局-局部最优适应值动态设置步长比例因子是可行的, 且能够改善 CS 算法的性能.

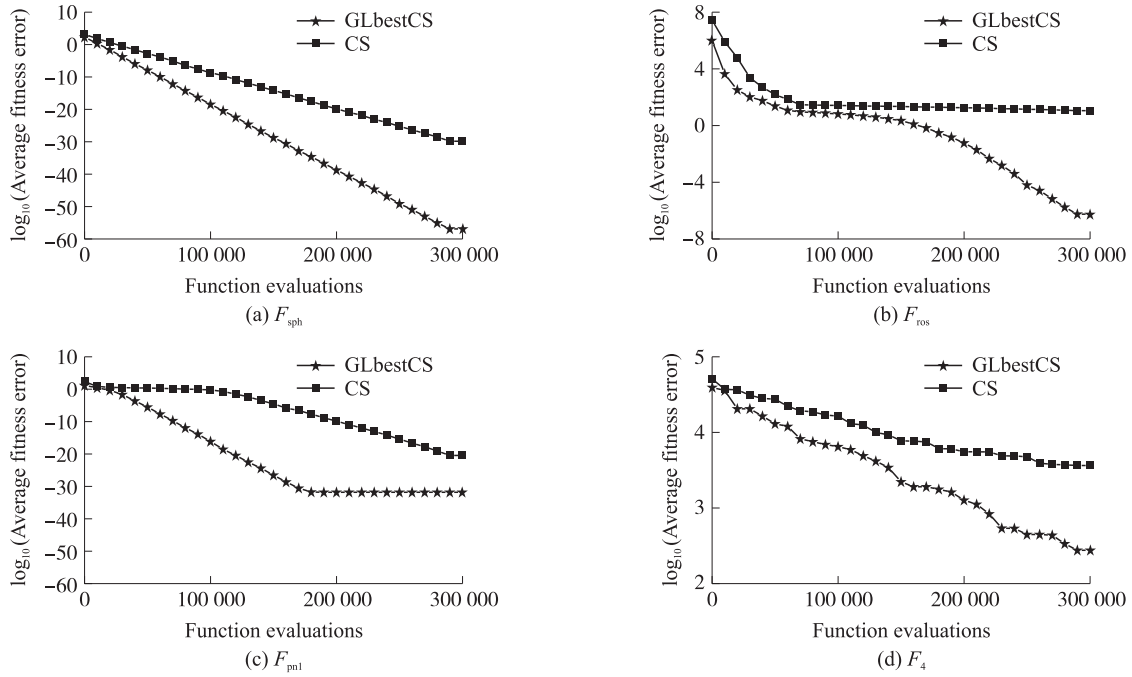


图 1 CS 和 GLbestCS 的收敛曲线

Fig. 1 The convergence curves of CS and GLbestCS

### 3.2 维数变化对算法性能影响分析

表 2 比较了 CS 和 GLbestCS 在不同维度的运行结果,可以看出 GLbestCS 算法的性能基本上稳定地优于 CS 算法.

 表 2 CS 和 GLbestCS 求解测试函数的平均适应值误差 ( $D=10$  and  $D=50$ )

 Table 2 Average fitness error obtained by CS and GLbestCS for benchmark functions ( $D=10$  and  $D=50$ )

Function	$D=10$		$D=50$	
	CS	GLbestCS	CS	GLbestCS
$F_{sph}$	$5.13e-26 \pm 8.08e-26$	$2.69e-57 \pm 3.58e-57$	$3.81e-17 \pm 2.31e-17$	$2.94e-31 \pm 2.24e-31$
$F_{ros}$	$9.99e-01 \pm 1.01e+00$	$9.28e-02 \pm 2.69e-01$	$4.04e+01 \pm 1.41e+01$	$3.61e+01 \pm 1.02e+01$
$F_{ack}$	$2.51e-11 \pm 3.45e-11$	$3.41e-15 \pm 7.11e-16$	$3.26e-02 \pm 1.52e-01$	$1.73e-14 \pm 2.77e-15$
$F_{grw}$	$3.13e-02 \pm 1.18e-02$	$2.39e-02 \pm 1.66e-02$	$2.34e-11 \pm 1.16e-10$	$0.00e+00 \pm 0.00e+00$
$F_{ras}$	$2.80e+00 \pm 1.12e+00$	$1.23e+00 \pm 1.19e+00$	$8.37e+01 \pm 9.89e+00$	$1.08e+02 \pm 2.46e+01$
$F_{sch}$	$7.21e+01 \pm 5.90e+01$	$5.74e-05 \pm 2.85e-04$	$4.95e+03 \pm 3.15e+02$	$4.32e+03 \pm 1.34e+03$
$F_{sal}$	$9.99e-02 \pm 2.15e-05$	$9.99e-02 \pm 2.24e-07$	$6.72e-01 \pm 8.91e-02$	$3.36e-01 \pm 4.90e-02$
$F_{whl}$	$2.31e+01 \pm 5.11e+00$	$1.57e+01 \pm 4.81e+00$	$1.33e+03 \pm 9.39e+01$	$1.27e+03 \pm 2.21e+02$
$F_{pn1}$	$8.18e-19 \pm 2.28e-18$	$4.71e-32 \pm 1.12e-47$	$2.06e-04 \pm 6.70e-04$	$2.40e-25 \pm 7.54e-25$
$F_{pn2}$	$2.14e-23 \pm 3.30e-23$	$1.35e-32 \pm 5.59e-48$	$8.23e-14 \pm 2.99e-13$	$4.48e-29 \pm 4.20e-29$
$F_1$	$6.13e-26 \pm 6.91e-26$	$0.00e+00 \pm 0.00e+00$	$2.46e-16 \pm 2.24e-16$	$0.00e+00 \pm 0.00e+00$
$F_2$	$1.13e-13 \pm 1.24e-13$	$3.49e-20 \pm 3.31e-20$	$2.34e+02 \pm 8.61e+01$	$7.77e+02 \pm 2.18e+02$
$F_3$	$2.02e+02 \pm 1.26e+02$	$9.26e+02 \pm 5.59e+02$	$8.46e+06 \pm 1.41e+06$	$2.05e+07 \pm 5.41e+06$
$F_4$	$8.84e-06 \pm 8.90e-06$	$8.99e-11 \pm 1.29e-10$	$2.69e+04 \pm 3.41e+03$	$1.98e+04 \pm 3.97e+03$
$F_5$	$1.30e-04 \pm 1.00e-04$	$4.30e-11 \pm 3.23e-11$	$1.06e+04 \pm 1.66e+03$	$6.01e+03 \pm 6.69e+02$
$F_6$	$1.58e+00 \pm 1.58e+00$	$9.91e-02 \pm 2.56e-01$	$5.82e+01 \pm 2.55e+01$	$4.96e+01 \pm 2.42e+01$
$F_7$	$4.76e-02 \pm 1.65e-02$	$5.37e-02 \pm 2.93e-02$	$1.81e-03 \pm 2.40e-03$	$1.25e-03 \pm 2.06e-03$
$F_8$	$2.03e+01 \pm 8.36e-02$	$2.04e+01 \pm 6.30e-02$	$2.11e+01 \pm 3.03e-02$	$2.11e+01 \pm 3.90e-02$
$F_9$	$2.47e+00 \pm 8.67e-01$	$1.08e+00 \pm 1.28e+00$	$1.20e+02 \pm 1.29e+01$	$9.81e+01 \pm 3.02e+01$
$F_{10}$	$1.96e+01 \pm 4.64e+00$	$1.41e+01 \pm 4.15e+00$	$3.86e+02 \pm 5.46e+01$	$2.61e+02 \pm 2.39e+01$
+/-/-	15/4/1		13/4/3	

### 3.3 与其他改进的 CS 算法比较分析

表 3 所列为 VCS、BetaCS 和 GLbestCS 算法在测试函数上的表现. 表 4 的方法是 Wilcoxon 配对秩和检验<sup>[26]</sup>, 其中“R+”表示 GLbestCS 算法获得的平均适应值误差排序总和比另一算法的更好, “R-”相反. 表 5 给出了 Friedman 检验<sup>[27]</sup>结果. 综合而言, GLbestCS 算法的性能优于 VCS 和 BetaCS 算法.

表 3 VCS、BetaCS 和 GLbestCS 求解测试函数的平均适应值误差 ( $D=30$ )

Table 3 Average fitness error obtained by VCS, BetaCS and GLbestCS for benchmark functions ( $D=30$ )

Function	AVGEr±SDEr		AVGEr±SDEr	
	VCS	GLbestCS	BetaCS	GLbestCS
$F_{sph}$	2.96e-49±6.25e-49	1.84e-57±2.20e-57	2.44e-57±4.05e-57	1.84e-57±2.20e-57
$F_{ros}$	4.50e+00±2.64e+00	6.19e+00±1.39e+01	2.82e+00±3.54e+00	6.19e+00±1.39e+01
$F_{ack}$	7.25e-15±7.11e-16	7.11e-15±0.00e+00	7.11e-15±0.00e+00	7.11e-15±0.00e+00
$F_{grw}$	0.00e+00±0.00e+00	0.00e+00±0.00e+00	0.00e+00±0.00e+00	0.00e+00±0.00e+00
$F_{ras}$	1.61e+01±6.27e+00	1.47e+01±6.08e+00	1.70e+01±6.58e+00	1.47e+01±6.08e+00
$F_{sch}$	6.13e+02±3.94e+02	5.39e+02±4.05e+02	4.04e+02±2.09e+02	5.39e+02±4.05e+02
$F_{sal}$	2.24e-01±4.36e-02	2.04e-01±2.00e-02	2.16e-01±3.74e-02	2.04e-01±2.00e-02
$F_{wht}$	2.56e+02±6.65e+01	2.50e+02±6.10e+01	2.74e+02±5.91e+01	2.50e+02±6.10e+01
$F_{pn1}$	1.57e-32±5.59e-48	1.57e-32±5.59e-48	1.57e-32±5.59e-48	1.57e-32±5.59e-48
$F_{pn2}$	1.35e-32±5.59e-48	1.35e-32±5.59e-48	1.35e-32±5.59e-48	1.35e-32±5.59e-48
$F_1$	0.00e+00±0.00e+00	8.08e-30±4.04e-29	0.00e+00±0.00e+00	8.08e-30±4.04e-29
$F_2$	1.49e-02±9.52e-03	6.28e-03±4.05e-03	5.91e-03±3.84e-03	6.28e-03±4.05e-03
$F_3$	3.60e+06±9.64e+05	3.26e+06±1.13e+06	3.31e+06±1.02e+06	3.26e+06±1.13e+06
$F_4$	4.45e+02±3.82e+02	3.78e+02±2.90e+02	4.45e+02±3.35e+02	3.78e+02±2.90e+02
$F_5$	1.88e+03±6.89e+02	1.47e+03±7.89e+02	1.90e+03±6.78e+02	1.47e+03±7.89e+02
$F_6$	1.05e+01±1.39e+01	9.01e+00±8.31e+00	9.68e+00±1.82e+01	9.01e+00±8.31e+00
$F_7$	4.81e-03±5.95e-03	4.48e-03±8.97e-03	3.51e-03±4.57e-03	4.48e-03±8.97e-03
$F_8$	2.10e+01±4.63e-02	2.09e+01±5.92e-02	2.09e+01±4.99e-02	2.09e+01±5.92e-02
$F_9$	1.88e+01±5.88e+00	1.71e+01±7.87e+00	1.69e+01±7.99e+00	1.71e+01±7.87e+00
$F_{10}$	1.00e+02±1.51e+01	1.09e+02±1.90e+01	1.05e+02±2.06e+01	1.09e+02±1.90e+01
+/-/-	4/16/0		1/19/0	

表 4 VCS、BetaCS 和 GLbestCS 的 Wilcoxon 配对秩和检验 ( $D=30$ )

Table 4 Wilcoxon matched-pairs signed-rank test for VCS, BetaCS and GLbestCS ( $D=30$ )

Algorithm	R+	R-	P-value	$\alpha=0.05$	$\alpha=0.1$
GLbestCS vs. VCS	168.40	41.60	0.017 938	+	+
GLbestCS vs. BetaCS	97.25	112.75	0.772 330	=	=

表 5 3 个算法求解测试函数的平均 Friedman 检验秩 ( $D=30$ )

Table 5 Average ranking of three algorithms by the Friedman test for 20 functions ( $D=30$ )

Algorithm	GLbestCS	BetaCS	VCS
ranking	1.73	1.85	2.42

### 3.4 对 $k$ 值的讨论

表 6 给出当  $k$  取值不同时 GLbestCS 算法在测试函数上取得的结果.

表 6 GLbestCS 求解测试函数的平均适应值误差 ( $k=0.5, k=1$  and  $k=1.1$ )

Table 6 Average fitness error obtained by GLbestCS for benchmark functions ( $k=0.5, k=1$  and  $k=1.1$ )

Function	$k=1.1$ GLbestCS	P-value	$k=0.5$ GLbestCS	$k=1.0$ GLbestCS	P-value	$k=0.5$ GLbestCS
$F_{sph}$	3.48e-43±3.94e-43	+ 0.000 012	1.84e-57±2.20e-57	4.54e-48±7.39e-48	+ 0.000 012	1.84e-57±2.20e-57
$F_{ros}$	2.15e+00±2.61e+00	- 0.039 554	6.19e+00±1.39e+01	1.49e+01±1.41e+01	+ 0.000 328	6.19e+00±1.39e+01
$F_{ack}$	6.96e-15±7.11e-16	= 1	7.11e-15±0.00e+00	7.53e-15±1.56e-15	= 0.5	7.11e-15±0.00e+00
$F_{grw}$	0.00e+00±0.00e+00	= 1	0.00e+00±0.00e+00	0.00e+00±0.00e+00	= 1	0.00e+00±0.00e+00
$F_{ras}$	1.17e+01±3.61e+00	= 0.103 553	1.47e+01±6.08e+00	1.77e+01±6.34e+00	= 0.121 828	1.47e+01±6.08e+00
$F_{sch}$	4.49e+02±2.21e+02	= 0.581 228	5.39e+02±4.05e+02	9.45e+02±4.50e+02	+ 0.001 303	5.39e+02±4.05e+02
$F_{sal}$	2.08e-01±2.77e-02	= 0.264 15	2.04e-01±2.00e-02	2.68e-01±4.76e-02	+ 0.000 364	2.04e-01±2.00e-02
$F_{wht}$	2.19e+02±8.20e+01	= 0.103 553	2.50e+02±6.10e+01	2.83e+02±6.77e+01	= 0.264 15	2.50e+02±6.10e+01



续表 6

Table 6 continued

Function	$k=1.1$ GLbestCS	$P$ -value	$k=0.5$ GLbestCS	$k=1.0$ GLbestCS	$P$ -value	$k=0.5$ GLbestCS
$F_{pn1}$	$1.57e-32 \pm 5.59e-48$	= 1	$1.57e-32 \pm 5.59e-48$	$4.15e-03 \pm 2.07e-02$	= 1	$1.57e-32 \pm 5.59e-48$
$F_{pn2}$	$1.35e-32 \pm 2.47e-34$	= 1	$1.35e-32 \pm 5.59e-48$	$1.35e-32 \pm 5.59e-48$	= 1	$1.35e-32 \pm 5.59e-48$
$F_1$	$0.00e+00 \pm 0.00e+00$	= 1	$8.08e-30 \pm 4.04e-29$	$0.00e+00 \pm 0.00e+00$	= 1	$8.08e-30 \pm 4.04e-29$
$F_2$	$5.26e-03 \pm 3.15e-03$	= 0.562 928	$6.28e-03 \pm 4.05e-03$	$3.03e-03 \pm 1.69e-03$	= 0.001 721	$6.28e-03 \pm 4.05e-03$
$F_3$	$3.19e+06 \pm 1.17e+06$	= 0.637 733	$3.26e+06 \pm 1.13e+06$	$2.79e+06 \pm 8.79e+05$	= 0.115 475	$3.26e+06 \pm 1.13e+06$
$F_4$	$3.04e+02 \pm 2.72e+02$	= 0.339 479	$3.78e+02 \pm 2.90e+02$	$4.03e+02 \pm 2.54e+02$	= 0.618 641	$3.78e+02 \pm 2.90e+02$
$F_5$	$1.65e+03 \pm 6.07e+02$	= 0.252 813	$1.47e+03 \pm 7.89e+02$	$2.31e+03 \pm 6.58e+02$	+ 0.000 194	$1.47e+03 \pm 7.89e+02$
$F_6$	$8.57e+00 \pm 1.72e+01$	= 0.150 003	$9.01e+00 \pm 8.31e+00$	$1.98e+01 \pm 2.08e+01$	+ 0.017 253	$9.01e+00 \pm 8.31e+00$
$F_7$	$5.36e-03 \pm 6.81e-03$	= 0.231 167	$4.48e-03 \pm 8.97e-03$	$9.61e-03 \pm 1.02e-02$	+ 0.034 67	$4.48e-03 \pm 8.97e-03$
$F_8$	$2.09e+01 \pm 5.56e-02$	= 0.459 336	$2.09e+01 \pm 5.92e-02$	$2.09e+01 \pm 4.71e-02$	= 0.069 337	$2.09e+01 \pm 5.92e-02$
$F_9$	$1.43e+01 \pm 5.28e+00$	= 0.191 898	$1.71e+01 \pm 7.87e+00$	$2.13e+01 \pm 7.42e+00$	= 0.073 565	$1.71e+01 \pm 7.87e+00$
$F_{10}$	$1.01e+02 \pm 1.70e+01$	= 0.165 837	$1.09e+02 \pm 1.90e+01$	$8.92e+01 \pm 1.41e+01$	= 0.001 186	$1.09e+02 \pm 1.90e+01$
+/-/-		1/18/1			7/11/2	

可以看出,当  $k=0.5$  和  $k=1.1$  时算法的性能近似,均优于  $k=1.0$  时. 在表 7 中,部分情况下  $k=1.1$  的算法性能略优. 由于个体靠近全局最优时,方程(5)后半部分的比值变大,  $\alpha_{0i}$  的值逐渐变小. 当  $k=0.5$  时,若当前个体达到全局最优,即  $f_{gbest}/f_{pi} = 1$  时,  $\alpha_{0i}$  取得最小值-0.5, Lévy Flights 随机游走的方向改变,算法会继续反向搜索. 故本文最终选择了较优的  $k=0.5$ .

表 7 GLbestCS 求解测试函数的平均 Friedman 检验秩 ( $D=30$ )Table 7 Average ranking of GLbestCS by the Friedman test for 20 functions ( $D=30$ )

algorithm	$k=0.5$ , GLbestCS	$k=1.0$ , GLbestCS	$k=1.1$ , GLbestCS
ranking	2.00	2.35	1.65

### 3.5 在其他算法上应用的讨论

将本文所提算法应用于文献[28]提出的 CCS 算法,命名为 GLbestCCS 算法. 表 8 和表 9 列出了 CCS 和 GLbestCCS 算法在  $D=30$  时的运行结果. 可以看出, GLbestCCS 算法的性能显著优于 CCS 算法.

表 8 CCS 和 GLbestCCS 求解测试函数的平均适应值误差 ( $D=30$ )Table 8 Average fitness error obtained by CCS and GLbestCCS for benchmark functions ( $D=30$ )

Function	AVG <sub>Er</sub> $\pm$ SD <sub>Er</sub>		Function	AVG <sub>Er</sub> $\pm$ SD <sub>Er</sub>	
	CCS	GLbestCS		CCS	GLbestCS
$F_{sph}$	$5.08e-55 \pm 9.57e-55$	$2.75e-58 \pm 8.33e-58$	$F_1$	$0.00e+00 \pm 0.00e+00$	$0.00e+00 \pm 0.00e+00$
$F_{ros}$	$2.12e-01 \pm 8.10e-01$	$1.62e-01 \pm 7.98e-01$	$F_2$	$5.34e-09 \pm 7.74e-09$	$5.72e-09 \pm 1.31e-08$
$F_{ack}$	$8.24e-15 \pm 2.66e-15$	$9.09e-15 \pm 3.09e-15$	$F_3$	$8.89e+04 \pm 5.33e+04$	$9.99e+04 \pm 5.32e+04$
$F_{grw}$	$0.00e+00 \pm 0.00e+00$	$0.00e+00 \pm 0.00e+00$	$F_4$	$3.20e+01 \pm 9.74e+01$	$7.37e+00 \pm 8.09e+00$
$F_{ras}$	$1.48e+01 \pm 6.90e+00$	$1.28e+01 \pm 7.27e+00$	$F_5$	$6.84e+02 \pm 6.65e+02$	$4.36e+02 \pm 4.24e+02$
$F_{sch}$	$7.65e+02 \pm 2.67e+02$	$6.59e+02 \pm 2.77e+02$	$F_6$	$9.31e-01 \pm 2.58e+00$	$7.26e-01 \pm 1.47e+00$
$F_{sal}$	$2.64e-01 \pm 4.90e-02$	$2.40e-01 \pm 5.00e-02$	$F_7$	$5.71e-03 \pm 6.58e-03$	$4.53e-03 \pm 7.02e-03$
$F_{wht}$	$1.76e+02 \pm 7.48e+01$	$1.77e+02 \pm 7.22e+01$	$F_8$	$2.09e+01 \pm 7.18e-02$	$2.09e+01 \pm 4.98e-02$
$F_{pn1}$	$1.57e-32 \pm 5.59e-48$	$1.57e-32 \pm 5.59e-48$	$F_9$	$1.62e+01 \pm 5.19e+00$	$1.40e+01 \pm 5.79e+00$
$F_{pn}$	$1.35e-32 \pm 5.59e-48$	$1.35e-32 \pm 2.47e-34$	$F_{10}$	$6.14e+01 \pm 1.48e+01$	$6.30e+01 \pm 1.21e+01$

表 9 CCS 和 GLbestCCS 的 Wilcoxon 配对秩和检验 ( $D=30$ )Table 9 Wilcoxon matched-pairs signed-rank test for CCS and GLbestCCS ( $D=30$ )

Algorithm	R+	R-	$P$ -value	$\alpha=0.05$	$\alpha=0.1$
GLbestCS vs. CCS	176.00	34.00	0.008 034	+	+

## 4 结论

本文采用最优适应值动态设置步长比例因子,并提出一种带全局-局部最优步长比例因子的布谷鸟

搜索算法 GLbestCS. 实验结果证明, GLbestCS 算法可有效地改善 CS 算法的性能. 当问题规模发生变化, GLbestCS 算法仍然稳定地优于 CS 算法. 将该策略应用于其他改进的 CS 算法, 结果表明这种优化仍然有效.

### [参考文献] (References)

- [1] 王永安, 赵阳, 蓝雨晨, 等. 基于进化差分算法的环形电感建模及应用[J]. 南京师范大学学报(工程技术版), 2020, 20(3): 32-37.
- [2] 王姝, 张海龙, 王恩荣. 基于改进粒子群算法的配电网网架优化研究[J]. 南京师范大学学报(工程技术版), 2020, 20(1): 15-19.
- [3] 沈燕平, 鞠勇, 居茜. 考虑间歇性负荷的光储微网储能的优化配置技术研究[J]. 南京师范大学学报(工程技术版), 2020, 20(2): 39-43, 92.
- [4] 赵世田, 付莹莹, 曾勇, 等. 基于非线性自适应度函数的遗传算法求取自由曲面最大主曲率研究[J]. 南京师范大学学报(工程技术版), 2018, 18(3): 19-24.
- [5] YANG X S, DEB S. Cuckoo search via Lévy flights[C]//Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC'09). Coimbatore, India: IEEE, 2009.
- [6] YANG X S, DEB S. Engineering optimisation by cuckoo search[J]. International Journal of Mathematical Modelling and Numerical Optimisation, 2010, 1(4): 330-343.
- [7] WANG G G, GANDOMI A H, ZHAO X, et al. Hybridizing harmony search algorithm with cuckoo search for global numerical optimization[J]. Soft Computing, 2016, 20(1): 273-285.
- [8] ROSLI S J, RAHIM H A, ABDUL R K N, et al. A hybrid modified method of the sine cosine algorithm using latin hypercube sampling with the cuckoo search algorithm for optimization problems[J]. Electronics, 2020, 9(11): 1786.
- [9] SHEHAB M, KHADER A T, LAOUCHEDI M. A hybrid method based on cuckoo search algorithm for global optimization problems[J]. Journal of Information and Communication Technology, 2018, 17(3): 469-491.
- [10] SHEHAB M, KHADER A T, LAOUCHEDI M, et al. Hybridizing cuckoo search algorithm with bat algorithm for global numerical optimization[J]. Journal of Supercomputing, 2019, 75(5): 2395-2422.
- [11] LI X T, WANG J N, YIN M H. Enhancing the performance of cuckoo search algorithm using orthogonal learning method[J]. Neural Computing and Applications, 2014, 24(6): 1233-1247.
- [12] WANG L J, ZHONG Y W. One-position inheritance based cuckoo search algorithm[J]. International Journal of Computing Science and Mathematics, 2015, 6(6): 546-554.
- [13] WANG L J, YIN Y L, ZHONG Y W. Cuckoo search with varied scaling factor[J]. Frontiers of Computer Science, 2015, 9(4): 623-635.
- [14] 王李进, 尹义龙, 钟一文. 逐维改进的布谷鸟搜索算法[J]. 软件学报, 2013, 24(11): 2687-2698.
- [15] 王李进, 钟一文, 尹义龙. 带外部存档的正交交叉布谷鸟搜索算法[J]. 计算机研究与发展, 2015, 52(11): 2496-2507.
- [16] SALGOTRA R, SINGH U, SAHA S, et al. Self adaptive cuckoo search: analysis and experimentation[J]. Swarm and Evolutionary Computation, 2021, 60: 100751.
- [17] WEI J M, YU Y G. A novel cuckoo search algorithm under adaptive parameter control for global numerical optimization[J]. Soft Computing, 2019, 24: 4917-4940.
- [18] WANG K, LIAN X F, PAN B. Improved cuckoo search algorithm based on exponential function[C]//Proceedings of the 2019 Chinese Intelligent Automation Conference. Zhenjiang, China: Springer, 2019.
- [19] CHEN L, LU H Q, LI H W, et al. Dimension-by-dimension enhanced cuckoo search algorithm for global optimization[J]. Soft Computing, 2019, 23(21): 11297-11312.
- [20] 林要华, 梁忠, 胡华平. 贝塔分布的布谷鸟搜索算法[J]. 南京大学学报(自然科学版), 2016, 52(4): 638-646.
- [21] WALTON S, HASSAN O, MORGAN K, et al. Modified cuckoo search: a new gradient free optimisation algorithm[J]. Chaos, Solitons & Fractals, 2011, 44(9): 710-718.
- [22] VALIAN E, TAVAKOLI S, MOHANNA S, et al. Improved cuckoo search for reliability optimization problems[J]. Computers & Industrial Engineering, 2013, 64(1): 459-468.
- [23] LAYEB A. A novel quantum inspired cuckoo search for knapsack problems[J]. International Journal of Bio-inspired Computation, 2011, 3(5): 297-305.

(下转第 72 页)