计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.191100207

具有自适应步长的柯西变异乌鸦算法

霍 林 郭雅蓉 覃志健
广西大学计算机与电子信息学院 南宁 530004

摘 要 针对乌鸦算法收敛速度慢、容易陷入局部最优的问题,提出了一种具有自适应步长的柯西变异乌鸦算法(Cauchy mutation crow search algorithm with adaptive step size,CMCSA),对标准乌鸦算法中两种情况下的位置更新策略进行了改进。在每次迭代时,利用柯西变异优化 *gbest* 来增强全局搜索能力和增大变异范围,以提高种群多样性,避免陷入局部最优;引入判别概率,在引导者发现自己被跟随的情况下优化当前个体的位置更新策略;根据当前位置和引导者之间的位置距离,自适应地调整步长,使算法平稳快速地收敛到全局最优,从而控制搜索速度和精度,有效弥补了标准 CSA 寻优方式的盲目性和收敛速度慢的缺陷。为评价 CMCSA 算法的有效性,将其应用于 10 个基本测试函数进行寻优实验,并与其他 8 种智能优化算法进行比较。实验结果表明,所提算法的平均收敛性和鲁棒性都优于其他算法,寻优平均值和标准差的平均排名均为第一,总体性能良好。

关键词:乌鸦算法;柯西变异;函数优化;自适应步长
中图法分类号 TP301

Crow Search Algorithm with Cauchy Mutation and Adaptive Step Size

HUO Lin,GUO Ya-rong and QIN Zhi-jian
School of Computer,Electronics and Information,Guangxi University,Nanning 530004,China

Abstract Aiming at the problems of slow convergence speed and local optimization of crow algorithm,this paper proposes a cauchy mutation crow algorithm with adaptive step size (CMCSA),to improve the position updating strategy of two situations in standard crow algorithm. In each iteration,the Cauchy mutation is used to optimize the *gbest*,to enhance the global search capability and increase the variation range,so as to improve the population diversity and avoid falling into local optimization. The discriminant probability is introduced to optimize the updating strategy of the current individual's position when the leader finds himself followed. The step length is adjusted adaptively according to the position distance between the current position and the leader's position,so that the algorithm converges smoothly and quickly to the global optimum,thus controlling the search speed and accuracy,and effectively compensating for the blindness and slow convergence of the standard CSA. In order to evaluate the effectiveness of the algorithm,the proposed CMCSA is applied to optimize ten basic test functions,and compared with eight other famous and recent intelligent optimization algorithms. The experimental results show that the proposed algorithm is superior to other algorithms in average convergence and robustness. The average ranking of the mean value and standard deviation value of the algorithm is the first,so it has better overall performance.

Keywords Crow algorithm,Cauchy mutation,Function optimization,Adaptive step-size

1 引言

乌鸦搜索算法(Crow Search Algorithm,CSA)[1]是由 Askarzadeh 于 2016 年提出的一种新的自然启发式算法。该算法通过模拟乌鸦觅食和隐藏食物的行为来建立相关数学模型。乌鸦算法由于具有结构简单、控制参数少、易实现的优点,对于计算耗时较长的评估问题,是一个很好的优化器,现已被广泛运用于各种实际优化问题的求解,包括云数据中心虚拟机资源感知布局策略[2]、经济负荷分配[3]、电磁优化[4]、车辆路径管理[5]、电容器适配[6]、数据隐私保护[7]等多个领域。

针对乌鸦算法在实际应用中的不足,研究者进行了相关改进,一类是在基本乌鸦算法上对其参数进行分析和改进,另一类是与其他启发式算法相结合,利用其他算法的优势进行混合优化,以弥补乌鸦算法的不足。对于第一类改进方法,文献[8]通过混沌序列初始化乌鸦的位置,使初始粒子在整个搜

到稿日期:2019-11-27 返修日期:2020-04-30 本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家社科基金(16ZDA092);广西高水平创新团队及卓越学者-数字东盟云大数据安全与挖掘技术项目
This work was supported by the National Social Science Foundation of China (16ZDA092) and High Level Innovation Team and Outstanding Scholar Project of Guangxi.
通信作者:霍林(Lhuo@gxu.edu.cn)

索空间均匀分布，具有多样性，避免陷入局部最优；并对粒子进行贪心修复，使用优化策略，使潜在解成为可行解。文献[9]提出了基于邻代最优解维度交叉策略的思想，引入加权学习因子，使感知概率 AP 和步长自适应调整。Shi 等[10]采用轮盘赌选择方案选取每次迭代中被跟随的乌鸦，引入惯性权重因子改变位置更新策略，从而改进标准 CSA 选择跟随乌鸦的随机性。文献[11]利用迭代过程中当前得到的最差适应度值，动态调整感知概率，并引入莱维飞行进行随机扰动，以提高原始 CSA 的搜索能力。对于第二类改进措施，ARORA 等[12]将 CSA 与灰狼算法 GWO 相结合并将其应用于特征选择。文献[13]提出了一种多目标乌鸦搜索策略，将乌鸦搜索算法与混沌果蝇算法相结合，将所得改进混合算法应用于组合交互测试中的约束处理过程。文献[5]将猫群算法和乌鸦算法相结合，乌鸦算法降低了 CSO 算法获得最优解的复杂度，从而能较好地解决具有时间窗的车辆路径问题。文献[14]将遗传算法的交叉和变异操作用于 CSA 的个体更新，以得到更优质的个体向量，并将其应用于可再生能源光伏/风能/电池混合系统的设计。

本文主要从充分利用最优个体、增加种群多样性、自适应步长 3 方面对基本乌鸦算法进行改进，使其能够更好地权衡勘探和开发。通过采用经柯西变异的全局最优个体引导位置更新，对经典 CSA 进行增强，从而实现了优化过程中丰富种群多样性信息和提高收敛速度的双重目标。为了深入研究 CMCSA 算法的性能，本文使用常用的 10 个基准测试函数进行测试，且与经典的及最新提出的元启发式算法（差分算法、粒子群算法、蝙蝠算法、灰狼算法、萤火虫算法及两种改进的乌鸦算法）进行对比实验，从而验证了改进算法的有效性。

2　乌鸦搜索算法

通过研究发现，乌鸦的大脑相对于身体的比例比其他鸟类大得多，它们拥有较其他鸟类更高的智商，因此乌鸦被认为是非常聪明的鸟类。研究者们通过了解乌鸦的习性发现，乌鸦会观察其他鸟类藏食物的地方，一旦主人离开，乌鸦就会把食物偷走。如果一只乌鸦进行了偷窃行为，它就会采取额外的预防措施，如移动藏身之处，以避免成为之后的受害者。乌鸦利用自己偷走其他鸟类食物的经验来预测其他小偷的行为，并能找到使其贮藏物不被偷窃的最安全的藏身之处。CSA 是一种基于种群的技术，通过模拟乌鸦上述生物行为来找到优化问题的最佳解决方案。CSA 遵守 4 个原则：1）乌鸦以群居的形式生活；2）乌鸦能记住它们的藏身之处；3）乌鸦互相跟踪偷窃；4）乌鸦保护它们的贮藏物最大概率地不被偷窃。标准 CSA 的主要步骤如下。

步骤 1　初始化 CSA 的决策变量和可调控参数：种群大小（n）、最大迭代次数（$itermax$）、飞行步长（fl）和感知概率（AP）。

步骤 2　初始化乌鸦个体的位置和记忆矩阵。在 d 维搜索空间中随机生成 n 只乌鸦，每只乌鸦 $\boldsymbol{x}_i=(X_{i,1},X_{i,2},\cdots,X_{i,d})$ 表示一个问题的可行解。初始化每只乌鸦的记忆矩阵 $\boldsymbol{m}_i$，由于在最初迭代时，乌鸦没有经验，因此假设它们在最初的位置隐藏了食物，即记忆矩阵为初始位置矩阵。

步骤 3　评价每只乌鸦的适应度函数（目标函数）$Fitness(i)$。计算每只乌鸦所在位置的质量。

步骤 4　在搜索空间中生成每只乌鸦的新位置。乌鸦 i 随机选择一只乌鸦（例如乌鸦 j），并跟随它来发现隐藏食物的位置，由式(1)可得乌鸦 i 的新位置：

$$\boldsymbol{x}_i^{t+1}=\begin{cases}x_i^t+r_i\times fl_i^t\times(m_j^t-x_i^t), & r_j\geqslant AP_j^t\\ \text{a random position}, & \text{others}\end{cases}\tag{1}$$

其中，r_i 和 r_j 为 0 至 1 间均匀分布的随机数，AP_j^t 为乌鸦 j 在第 t 次迭代时的感知概率，所有的乌鸦都重复这个过程。

步骤 5　检查新位置的可行性。检查每只乌鸦的新位置是否可行，若可行，则乌鸦更新它的位置，否则乌鸦停留在当前位置，不会移动到生成的新位置。

步骤 6　评价新位置的适应度函数。

步骤 7　更新乌鸦的记忆矩阵：

$$\boldsymbol{m}_i^{t+1}=\begin{cases}\boldsymbol{x}_i^{t+1}, & f(\boldsymbol{x}_i^{t+1})\text{ is better than }f(\boldsymbol{m}_i^t)\\ \boldsymbol{x}_i^t, & \text{others}\end{cases}\tag{2}$$

其中，$f(x)$为适应度函数。若乌鸦的新位置适应度函数值优于记忆位置适应度函数值，则乌鸦根据新位置更新记忆，否则不更新。

步骤 8　重复步骤 4－步骤 7，直到达到终止条件。当满足终止条件时，根据适应度函数值将最佳位置作为优化问题的最优解。

3　具有自适应步长的柯西变异乌鸦算法（CMCSA）

为了提高算法的收敛速度，在寻优过程中必须在开发和勘探之间找到一个良好的平衡点。对于 CSA，导致多样化和集约化的主要参数是乌鸦的感知概率（AP）和步长（fl）。传统 CSA 固定参数 AP 和 fl，不利于算法的寻优。一些研究在一定程度上改进了算法的性能，但均是针对有引导者 j 的情况做出的改进，未能有效改进没有引导者的情况下位置更新的盲目性。另一些研究仅改变初始种群状态，并未对算法的寻优方式进行改进，不能弥补寻优方式的盲目性和收敛速度慢的缺陷。

鉴于乌鸦搜索算法及相关改进算法的不足，本文提出了具有自适应步长的柯西变异乌鸦算法（CMCSA），引入柯西变异算子对全局最优个体 $gbest$ 进行改进，有效解决了算法陷入局部最优的问题，提高了求解精度。在位置更新策略中，在不被引导者 j 发现的情况下，使用全局最优个体引导乌鸦的位置更新；在被引导者 j 发现的情况下，引入判别概率，根据一定策略重新随机选择位置或在当前位置附近进行高斯变异，得到下一步位置，以有效降低算法的盲目性，提高收敛速度。

3.1　全局最优个体的柯西变异策略

为了得到较强的全局搜索能力，对乌鸦群体中的最优个体使用变异策略进行变异操作，使算法跳出局部最优，得到全局搜索能力更高的个体。由于标准柯西分布在零点的波峰低于标准高斯分布，而零点两侧的下降趋势较缓慢，因此可知柯西变异的扰动能力较强，能获得更佳的寻优范围。

柯西变异公式如下：

$$\boldsymbol{x}_i' = \begin{cases} \boldsymbol{x}_i \times Cauchy(0,1), & rand(0,1) \leqslant p \\ \boldsymbol{x}_i, & \text{others} \end{cases} \tag{3}$$

其中，$Cauchy(0,1)$是标准柯西分布函数，$rand(0,1)$为一个 0 至 1 间均匀分布的随机数，p 为随机变异概率。

根据式(3)，引入柯西变异算子对全局最优个体 *gbest* 进行修正，得到柯西全局最优个体，具体操作如下：

$$cauchygbest = \begin{cases} gbest \times Cauchy(0,1), & rand \leqslant p \\ gbest, & \text{others} \end{cases} \tag{4}$$

3.2 新的自适应步长

乌鸦搜索算法中的步长 fl 影响了算法的收敛精度和速度，较小的步长拥有较详细的局部搜索能力，较大的步长能发挥较大的全局搜索能力。在每一次迭代时，步长依据引导者 j 和当前个体之前的距离D_i^t进行自适应调节变化。每次迭代中每个个体的步长变化如下：

$$fl_i^t = D_i^t \times \lambda \tag{5}$$

其中，$D_i^t = \sqrt{\sum_{k=1}^{d} (m_{jk}^t - x_{ik}^t)^2}$ 表示引导者 j 和当前个体之间的距离，λ 是缩放因子。所提算法根据设定的阈值将步长变化控制在合理范围内，从而调整步长的取值大小。

通过D_i^t的表达式可知，若D_i^t较大，则当前位置距离引导者较远，步长较大，所提算法体现全局搜索性能；若D_i^t较小，则距离较近，此时步长较小，在寻优后期当前个体和最优个体之间的距离差逐步缩小时，步长自动减小，在较优的个体周围发挥局部搜索能力，以此来平衡种群的多样性和集约性。

3.3 新的位置更新策略

假设所有乌鸦个体能根据自己的小偷经验和自身记忆矩阵得到全局最优 *gbest* 的位置，在进行下一次位置更新时，假定乌鸦 i 随机选择乌鸦 j 进行跟踪，则有两种情况：

1)乌鸦 j 不知道乌鸦 i 跟踪它，则乌鸦 i 接近乌鸦 j 藏匿食物的最佳位置 M，但是 j 的位置不一定是好的，因此引入全局最优个体 *gbest* 指引位置更新操作，乌鸦 i 能够根据最优位置信息和乌鸦 j 的位置信息来判断下一步的前进方向。为了解决算法陷入局部最优的问题，采用柯西变异后的全局最优 *Cauchygbest* 进行引导。

$$\boldsymbol{x}_i^{t+1} = Cauchygbest + r_{i1} \times fl_i^t \times (m_j^t - x_i^t) \tag{6}$$

2)乌鸦 j 知道乌鸦 i 跟踪它，则其故意把乌鸦 i 带到一个随机位置。但是标准 CSA 中，随机位置存在盲目性，为了避免位置选取的盲目性，假设此时乌鸦 i 具有一定的判别概率 θ(本文中 θ 取值为 0.5，表示选取两种策略的概率相等)。若乌鸦 i 没有判别出自己已被引导者 j 发现，则继续跟随 j 到达随机位置；若乌鸦 i 感知到引导者 j 已发现被跟踪，则不继续跟随 j，而是在当前藏匿位置附近搜索更好的位置，即在当前位置进行高斯变异，得到新的位置。判别策略如下：

$$\boldsymbol{x}_i^{t+1\prime} = \begin{cases} \text{a random position}, & r_{i2} < \theta \\ gbest \times N(0,1), & \text{others} \end{cases} \tag{7}$$

其中，r_{i2}为 0 至 1 间均匀分布的随机数；$N(0,1)$是标准正态分布，即服从均值为 0、方差为 1 的高斯分布函数。

综上，位置更新方式如下：

$$\boldsymbol{x}_i^{t+1} = \begin{cases} Cauchygbest + r_{i1} \times fl_i^t \times (m_j^t - x_i^t), & r_j \geqslant AP_j^t \\ x_i^{t+1\prime}, & \text{others} \end{cases} \tag{8}$$

其中，r_{i1}，r_{i2}，r_j是服从[0,1]均匀分布的随机数，r_{i1}控制乌鸦 i 在情况 1)下位置移动的方向，r_{i2}控制乌鸦 i 在情况 2)下的位置选择情况；$\boldsymbol{m}_j^t$为乌鸦 j 的记忆值；AP_j^t为乌鸦 j 的感知概率，fl_i^t为当前的飞行长度。

3.4 记忆矩阵更新

在进行位置更新之后，需要对每只乌鸦即搜索空间中的每个个体进行记忆矩阵的更新，更新方式如下：

$$m_i^{t+1} = \begin{cases} x_i^{t+1}, & f(x_i^{t+1}) \text{ is better than } f(m_i^t) \\ x_i^t, & \text{others} \end{cases} \tag{9}$$

3.5 带柯西变异及自适应步长的改进乌鸦算法的流程

在标准乌鸦搜索算法的基础上，本文提出的带柯西变异及自适应步长的改进乌鸦算法(CMCSA)的流程如算法 1 所示。

算法 1 CMCSA

输入：乌鸦种群的初始位置

输出：问题最优解(所有乌鸦记忆值中的最好位置)和最优个体

Begin：

步骤 1 初始化。在 d 维的可行域中随机生成 n 只乌鸦，每只乌鸦 X $=[x_1, x_2, \cdots, x_d]$ 表示一个可行解，初始化最大迭代评估次数 Evamax 和感知概率 AP。

步骤 2 初始化乌鸦记忆值，计算适应度值。

步骤 3 更新 gbest。

步骤 4 使用式(3)对 gbest 进行柯西变异，并进行越界处理。

步骤 5 使用式(5)、式(6)对步长 fl 进行自适应调整。

步骤 6 使用式(8)进行位移更新操作。

步骤 7 检查新位置的可行性。

步骤 8 计算新位置的适应度，根据式(9)更新记忆值。

步骤 9 重复步骤 3－步骤 8，直到达到终止条件，则停止迭代。

End

4 实验设置与结果分析

4.1 参数设置

本文选取 8 个对比算法，分别为 DE[15]，PSO[16]，FA[18]，BA[17]，GWO[19]，CSA[1]，ICSA[10]，APP-CSA[7]。由于各算法在各研究中最好效果对应的种群大小不一致，为了保证实验的公平性，采用评估次数作为算法的终止条件，将各算法的所有参数均按照对应文献提出的参数(包括种群大小和各关键参数)进行设置。为了降低偶然因素对实验的影响，最大评估次数 $EvaMax$ 设置为 $dim \times 10^4$，并进行 50 次独立重复实验。实验平台是 Matlab R2018a，Inter(R) Xeon(R) CPU E5-1620 v3 @3.5GHz，Window10 操作系统。

本文算法和 8 种对比算法的寻优精度均设置为 $\lambda = 1 \times 10^{-5}$，即到达寻优精度时，标记成功寻优一次。将 10 个测试函数寻优的平均值作为衡量算法收敛性的指标，以标准方差值为衡量鲁棒性的指标。

4.2 基准函数

为了评估本文 CMCSA 的性能，对不同的函数优化问题

进行了实验。采用 10 个不同的基准函数,这些函数按维数和模态分为 3 类,包括单峰函数、多峰函数和固定多模态函数。表 1 列出了 10 个基准函数,其中 f_1-f_6 的维度为 30 dim,f_7-f_{10} 的维度为 2 dim。

表 1　基准函数表

Table 1　Benchmark functions

Name	Function	dim	Range	$F_{\min}$
Sphere	$f_1(x)=\sum_{i=1}^{n}x_i^2$	30	$[-100,100]$	0
Rosenbrock	$f_2(x)=\sum_{i=1}^{n}[100(x_{i+1}-x_i)^2+(x_i-1)^2]$	30	$[-10,10]$	0
Griewank	$f_3(x)=\frac{1}{4\,000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	$[-600,600]$	0
Rastrigin	$f_4(x)=\sum_{i=1}^{n}x_i^2-10\cos(2\pi x_i+10)$	30	$[-5.12,5.12]$	0
Schwefel	$f_5(x)=\sum_{i=1}^{n}(\sum_{j=1}^{n}x_j^2)^2$	30	$[-100,100]$	0
Ackley	$f_6(x)=-20\exp(-0.2\sqrt{\sum_{i=1}^{n}x_i^2})-\exp\left(\frac{1}{n}\right)\sum_{i=1}^{n}\cos(2\pi x_i)+20+e$	30	$[-32,32]$	0
Six-Hump Camel-Back	$f_7(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	$[-5,5]$	$-1.031\,628$
Goldstein Price	$f_8(x)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\cdot[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	$[-2,2]$	3
Schaffer's F6	$f_9(x)=\frac{\sin^2\sqrt{x_1^2+x_2^2}-0.5}{[1+0.001(x_1^2+x_2^2)]^2}+0.5$	2	$[-100,100]$	0
Branin	$f_{10}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$	2	$-5<x_1<10$, $0<x_2<15$	0.398

4.3　收敛性分析

表 2 列出了所有算法在不同基准函数下的结果,包括每种算法得到的最优解的最大值 max、最小值 min、均值 *avg* 和标准差 *std*,并用黑体突出了最佳结果。为了更直观的比较,根据 *avg* 进行排名得到 Rank_1,其统计结果见后文表 3;根据 std 进行排名得到 Rank_2,其统计结果见 4.4 节表 4。

表 2　9 种算法的实验结果对比

Table 2　Comparison of nine algorithms for f_1-f_{10}

f	dim	Algorithm	max	min	*avg*	Rank_1	*std*	Rank_2
f_1	30	CMCSA	**0**	**0**	**0**	**1**	**0**	**1**
		CSA	3.89×10^{-18}	1.99×10^{-19}	1.23×10^{-18}	5	8.46×10^{-19}	5
		ICSA	3.93×10^{-34}	5.63×10^{-38}	3.05×10^{-35}	4	6.52×10^{-35}	4
		AAP-CSA	7.62×10^{-7}	1.38×10^{-8}	2.18×10^{-7}	6	1.70×10^{-7}	7
		BA	1.28×10^{-5}	7.54×10^{-6}	1.03×10^{-5}	8	1.24×10^{-6}	8
		DE	1.17×10^{-160}	5.17×10^{-164}	5.15×10^{-162}	3	1.66×10^{-161}	3
		GWO	**0**	**0**	**0**	**1**	**0**	**1**
		PSO	0.550 335	1.85×10^{-4}	0.066 608	9	0.110 874	9
		FA	9.50×10^{-7}	1.45×10^{-7}	6.44×10^{-7}	7	1.67×10^{-7}	6
f_2	30	CMCSA	25.858 819	25.356 419	25.630 242	4	**0.114 479**	**1**
		CSA	4.50×10^{2}	11.711 776	68.925 226	8	97.413 23	8
		ICSA	2.86×10^{1}	27.800 178	28.242 214	6	0.183 925	2
		AAP-CSA	1.19×10^{3}	23.526 039	1.38×10^{2}	9	2.27×10^{2}	9
		BA	29.404 653	3.601 504	7.151 497	2	5.957 77	4
		DE	76.358 152	19.910 849	22.722 063	3	7.840 059	5
		GWO	28.758 415	25.043 884	26.553 053	5	1.008 176	3
		PSO	5.68×10^{2}	30.892 825	56.052 57	7	78.290 761	7
		FA	67.551 91	0.001 879	**6.502 717**	**1**	16.427 266	6
f_3	30	CMCSA	**0**	**0**	**0**	**1**	**0**	**1**
		CSA	0.083 541	1.33×10^{-15}	0.013 725	8	0.016 999	8
		ICSA	**0**	**0**	**0**	**1**	**0**	**1**
		AAP-CSA	0.066 355	7.53×10^{-7}	0.008 469	7	0.013 018	7
		BA	6.74×10^{-7}	3.35×10^{-7}	5.33×10^{-7}	4	7.58×10^{-8}	4
		DE	0.007 396	0	2.96×10^{-4}	6	0.001 464	6
		GWO	0.007 497	0	1.50×10^{-4}	5	0.001 06	5
		PSO	0.213 452	0.054 659	0.121 83	9	0.037 035	9
		FA	5.21×10^{-8}	6.65×10^{-9}	3.02×10^{-8}	3	1.06×10^{-8}	3

（续表）

f	dim	Algorithm	max	min	*avg*	Rank_1	*std*	Rank_2
		CMCSA	**0**	**0**	**0**	**1**	**0**	**1**
		CSA	6.99×10^{2}	2.29×10^{2}	4.11×10^{2}	9	1.10×10^{2}	9
		ICSA	**0**	**0**	0	1	0	1
		AAP-CSA	1.93×10^{2}	5.87×10	1.14×10^{2}	7	2.92×10	8
f_4	30	BA	63.679191	12.936619	37.890049	6	11.511926	6
		DE	0.994959	0	0.079597	4	0.272666	4
		GWO	**0**	**0**	**0**	**1**	**0**	**1**
		PSO	1.66×10^{2}	61.340304	1.19×10^{2}	8	25.40097	7
		FA	40.793752	24.874344	32.37634	5	4.130343	5
		CMCSA	**0**	**0**	**0**	**1**	**0**	**1**
		CSA	2.27×10^{2}	44.787999	1.23×10^{2}	6	42.446993	6
		ICSA	3.90×10^{-32}	1.29×10^{-35}	4.91×10^{-33}	2	8.40×10^{-33}	2
		AAP-CSA	9.35×10^{2}	2.00×10^{2}	4.49×10^{2}	8	1.55×10^{2}	7
f_5	30	BA	20.026387	0.245486	1.695506	3	3.512389	3
		DE	1.56×10^{2}	37.664981	1.16×10^{2}	5	21.676838	4
		GWO	1.19×10^{3}	7.18×10^{-4}	1.35×10^{2}	7	2.34×10^{2}	8
		PSO	1.69×10^{2}	5.052136	31.43531	4	31.132339	5
		FA	2.83×10^{3}	57.38253	5.32×10^{2}	9	4.95×10^{2}	9
		CMCSA	8.88×10^{-16}	8.88×10^{-16}	$\mathbf{8.88\times10^{-16}}$	**1**	**0**	**1**
		CSA	19.995022	3.574098	7.080553	7	2.996962	8
		ICSA	8.88×10^{-16}	8.88×10^{-16}	$\mathbf{8.88\times10^{-16}}$	**1**	**0**	**1**
		AAP-CSA	20.002032	1.155149	11.917516	8	8.853047	9
f_6	30	BA	3.78574	2.634678	2.634678	5	0.496025	6
		DE	7.99×10^{-15}	4.44×10^{-15}	7.64×10^{-15}	3	1.08×10^{-15}	3
		GWO	20.838074	20.451339	20.732984	9	0.086805	4
		PSO	3.433312	2.149803	2.656236	6	0.286044	5
		FA	3.222754	3.86×10^{-4}	0.687924	4	1.147772	7
		CMCSA	−1.031628	−1.031628	**−1.031628**	**1**	2.67×10^{-14}	4
		CSA	−1.031628	−1.031628	**−1.031628**	**1**	3.28×10^{-16}	3
		ICSA	−1.031577	−1.031628	−1.031619	7	1.01×10^{-5}	8
		AAP-CSA	−1.031628	−1.031628	**−1.031628**	**1**	8.06×10^{-13}	5
f_7	2	BA	−0.215463	−1.031628	−0.933688	9	0.267914	9
		DE	−1.031628	−1.031628	**−1.031628**	**1**	$\mathbf{2.24\times10^{-16}}$	**1**
		GWO	−1.031628	−1.031628	**−1.031628**	**1**	1.29×10^{-8}	7
		PSO	−1.031628	−1.031628	**−1.031628**	**1**	2.67×10^{-16}	2
		FA	−1	−1	−1	8	9.37×10^{-12}	6
		CMCSA	3.000001	3	**3**	**1**	2.01×10^{-7}	4
		CSA	3	3	**3**	**1**	$\mathbf{1.91\times10^{-15}}$	**1**
		ICSA	3.002915	3.000012	3.000493	6	5.90×10^{-4}	6
		AAP-CSA	3	**3**	**3**	**1**	8.18×10^{-13}	3
f_8	2	BA	8.40×10^{2}	3	3.43×10	8	1.20×10^{2}	9
		DE	3	**3**	**3**	**1**	3.47×10^{-15}	2
		GWO	3.000139	3	3.000024	5	3.12×10^{-5}	5
		PSO	30	3	5.7	7	8.182235	7
		FA	600	84	94.32	9	7.30×10	8
		CMCSA	**0**	**0**	**0**	**1**	**0**	**1**
		CSA	0.009715	0	0.005246	7	0.004891	7
		ICSA	6.37×10^{-5}	0	1.27×10^{-6}	2	9.01×10^{-6}	2
		AAP-CSA	0.009715	2.42×10^{-12}	0.004714	5	0.00474	5
f_9	2	BA	0.178222	1.15×10^{-11}	0.019413	8	0.021821	8
		DE	0.009715	0	0.001783	3	0.003757	3
		GWO	0.009716	0	5.12×10^{-3}	6	0.004857	6
		PSO	0.009716	0	0.002915	4	0.004497	4
		FA	0.101313	0	0.027386	9	0.044872	8
		CMCSA	0.397887	0.397887	**0.397887**	**1**	4.20×10^{-10}	6
		CSA	0.397887	0.397887	**0.397887**	**1**	$\mathbf{3.36\times10^{-16}}$	**1**
		ICSA	0.398108	0.397889	0.397888	7	4.71×10^{-5}	8
		AAP-CSA	0.397887	0.397887	**0.397887**	**1**	8.78×10^{-12}	5
f_{10}	2	BA	0.397887	0.397887	**0.397887**	**1**	5.97×10^{-10}	7
		DE	0.397887	0.397887	**0.397887**	**1**	$\mathbf{3.36\times10^{-16}}$	**1**
		GWO	0.398945	0.397887	0.397936	8	2.12×10^{-4}	9
		PSO	0.397887	0.397887	**0.397887**	**1**	$\mathbf{3.36\times10^{-16}}$	**1**
		FA	7.782704	7.782704	7.782704	9	4.49×10^{-15}	4

为了更加直观地观察收敛速度和收敛精度，图 1—图 10 给出了寻优进化曲线图，其中除了 Six-Hump-Camel-Back（f_7）测试函数进化曲线的适应度值不取对数，其他测试函数进化曲线的纵坐标中的适应度值均取对数 $\log_{10}$，即纵坐标为 $\log_{10}$ Fitness，横坐标为评估次数（对于最优解为 0 的测试函数来说，取对数后，由于 log0 不存在，因此此时寻优曲线无法

显示,即当寻优曲线不显示时即为成功找到了最优解 0)。

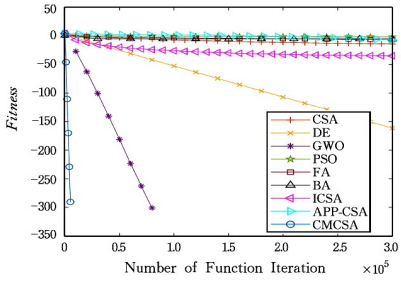

图 1　f_1 寻优进化曲线

Fig. 1　Evolution curve of f_1

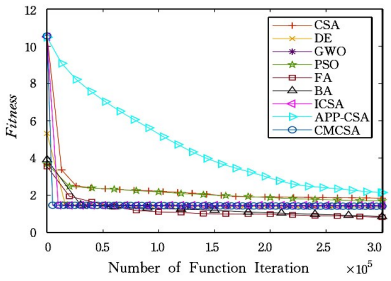

图 2　f_2 寻优进化曲线

Fig. 2　Evolution curve of f_2

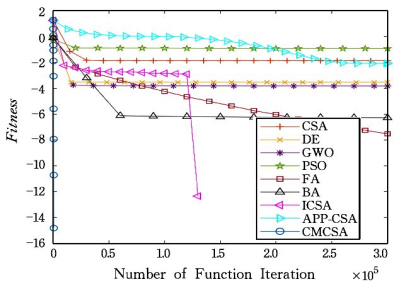

图 3　f_3 寻优进化曲线

Fig. 3　Evolution curve of f_3

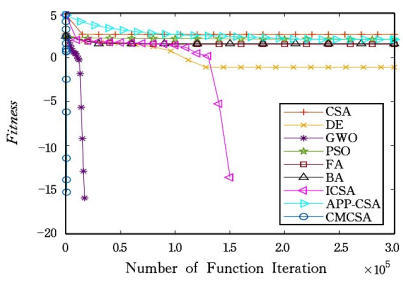

图 4　f_4 寻优进化曲线

Fig. 4　Evolution curve of f_4

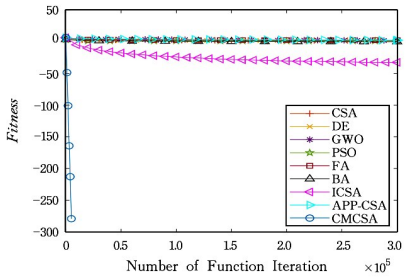

图 5　f_5 寻优进化曲线

Fig. 5　Evolution curve of f_5

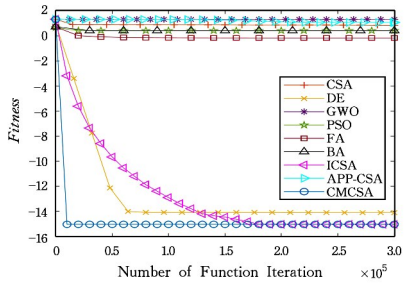

图 6　f_6 寻优进化曲线

Fig. 6　Evolution curve of f_6

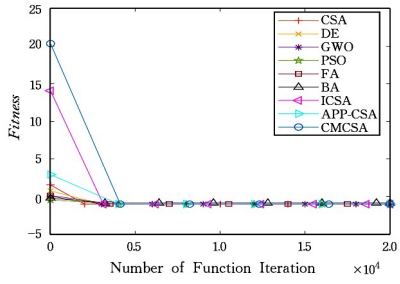

图 7　f_7 寻优进化曲线

Fig. 7　Evolution curve of f_7

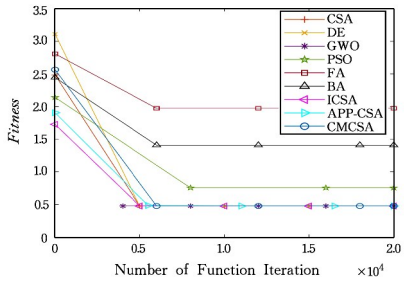

图 8　f_8 寻优进化曲线

Fig. 8　Evolution curve of f_8

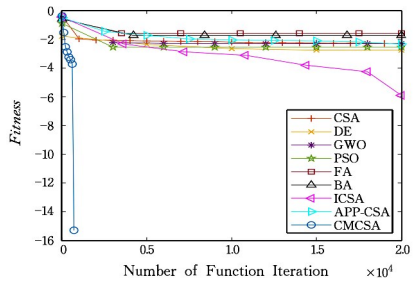

图 9　f_9 寻优进化曲线

Fig. 9　Evolution curve of f_9

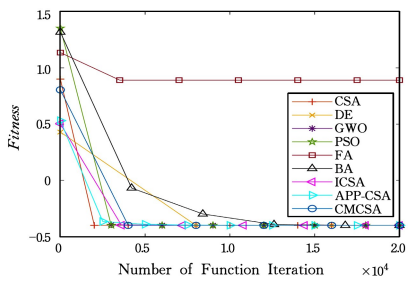

图 10　f_{10} 寻优进化曲线

Fig. 10　Evolution curve of f_{10}

由图1、图3—图5、图9可知，在优化 f_1，f_3，f_4，f_5，f_9 这5个函数的过程中，CMCSA能以较快的收敛速度和较少的评估次数寻找函数最优值。在寻优后期，除CMCSA和GWO外其余算法均陷入局部最优，未找到函数最优解。在寻优结束时，根据表2中 f_1，f_3，f_4，f_5，f_9 的 avg 值可知，CMCSA寻找到的最优平均值不亚于其他8种算法，且对于 f_5，仅有CMCSA算法能找到最优值，证明其收敛速度和收敛精度均优于其他算法。

由图2、图6可知，在优化 f_2 和 f_6 的过程中，在寻优前期，当迭代次数相同时，CMCSA的收敛速度和收敛精度均优于其他算法；在寻优后期，5种算法均陷入局部最优。在寻优结束时，由表2中 f_2 的 avg 结果可知，CMCSA寻找到的平均值劣于FA，BA，DE，但优于其余几种算法。就 f_6 而言，CMCSA寻找到的平均值不亚于其他几种算法。

由图7、图8、图10可知，在寻优前期，9种算法的收敛速度和收敛精度不相上下，在寻优后期，所有算法均陷入局部最优，在寻优结束时，根据表2中 f_7，f_8，f_{10} 的 avg 可知，CMCSA寻找到的最优平均值不亚于其他8种算法。表3列出了CMCSA和其他算法的对比情况，就9种算法寻找到的平均值 avg 而言，提出的CMCSA算法的性能总体最优，除 f_2 之外，在其余8个基准函数上均排名第一，平均排名也第一。

综上，CMCSA算法的总体收敛性能优于其他8种算法。

表3 9种算法寻优平均值排名

Table 3 Ranking of average values found by nine algorithms

Algorithm	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	avg
CMCSA	1	4	1	1	1	1	1	1	1	1	**1.3**
CSA	5	8	8	9	6	7	1	1	7	1	5.3
ICSA	4	6	1	1	2	1	7	6	2	7	3.7
AAP-CSA	6	9	7	7	8	8	1	1	5	1	5.3
BA	8	2	4	6	3	5	9	8	8	1	5.4
DE	3	3	6	4	5	3	1	1	3	1	3
GWO	1	5	5	1	7	9	1	5	6	8	4.8
PSO	9	7	9	8	4	6	1	7	4	1	5.6
FA	7	1	3	5	9	4	8	9	9	9	6.4

4.4 寻优成功率分析

由表4可知，9种算法对 f_2 和 f_{10} 的寻优成功率均为0，均没有找到最优理论值，CMCSA对其余8个基准函数的寻优成功率均为100%，ICSA对5个基准函数的寻优成功率为100%，DE算法对4个基准函数的寻优成功率为100%，CSA，APP-CSA，GWO对3个基准函数的寻优成功率为100%，FA对2个基准函数的寻优成功率为100%，而BA和PSO仅对1个基准函数的寻优成功率为100%。因此，CMCSA算法的总体寻优成功率最好。

表4 9种算法的寻优成功率

Table 4 Succeed rate of nine algorithms for f_1-f_{10}

（单位：%）

Algorithm	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
CMCSA	100	0	100	100	100	100	100	100	100	0
CSA	100	0	42	0	0	0	100	100	46	0
ICSA	100	0	100	100	100	100	80	0	98	0
AAP-CSA	100	0	48	0	0	0	100	100	28	0
BA	36	0	100	0	0	0	88	76	26	0
DE	100	0	96	92	0	100	100	100	76	0
GWO	100	0	98	100	0	0	100	52	46	0
PSO	0	0	0	0	0	0	100	90	70	0
FA	100	0	100	0	0	0	0	0	72	0

4.5 鲁棒性分析

表5列出了标准偏差std的统计结果排名，std反映了算法的鲁棒性。由表2中的 std 值和表5可知，对于 f_1-f_6 以及 f_9 基准函数而言，CMCSA的鲁棒性不亚于其他8种算法，且均排名第一，所得 std 均为0，表现出了较好的稳定性；对 f_7，f_8，f_{10} 寻优结束后，CMCSA的鲁棒性较劣，f_7 和 f_8 排名第四，f_{10} 排名第六，但其平均排名仍为最优，由此可知CMCSA的鲁棒性较好。

表5 9种算法的鲁棒性排名

Table 5 Robustness ranking of nine algorithms

Algorithm	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	avg
CMCSA	1	1	1	1	1	1	4	4	1	6	**2.1**
CSA	5	8	8	9	6	8	3	1	7	1	5.6
ICSA	4	2	1	1	2	1	8	6	2	8	3.5
AAP-CSA	7	9	7	8	7	9	5	3	5	5	6.5
BA	8	4	4	6	3	6	9	9	8	7	6.4
DE	3	5	6	4	4	3	1	2	3	1	3.2
GWO	1	3	5	1	8	4	7	5	6	9	4.9
PSO	9	7	9	7	5	5	2	7	4	1	5.6
FA	6	6	3	5	9	7	6	8	8	4	6.2

结束语 本文引入柯西变异算子对标准乌鸦算法进行改进，提出了一种自适应步长的柯西乌鸦算法(CMCSA)。该算法对全局最优个体进行柯西变异操作，帮助算法跳出局部最优，引导乌鸦进行高效的空间搜索；引入判别概率，其决定了乌鸦在被引导者发现的情况下的位置更新策略，避免了算法随机选择的盲目性；利用当前位置和引导者之间的距离来自适应地调整步长，从而改善寻优效果。

由10个基准函数测试的实验可知，相对其他8种经典的和最新提出的智能算法，改进的自适应柯西变异乌鸦算法具有相对较高的收敛精度和较快的收敛速度，在平均适应度和稳定性方面均优于其他算法，在函数优化领域具有显著优势。在以后的工作中，将对CMCSA的性能进行更深入的研究，并将其应用于更复杂的科学和实际工程问题。

参 考 文 献

[1] ASKARZADEH A. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm[J]. Computers & Structures, 2016, 169(Jun.): 1-12.

[2] SATPATHY A, ADDYA S K, TURUK A K, et al. A Resource Aware VM Placement Strategy in Cloud Data Centers Based on Crow Search Algorithm[C]// International Conference on Advanced Computing & Communication Systems. 2017.

[3] FARID M, HAMDIA. A Modified Crow Search Algorithm (MCSA) for Solving Economic Load Dispatch Problem[J]. Applied Soft Computing, 2018, 71: 51-65.

[4] COELHO L D S, KLEIN C E, MARIANI V C, et al. Electromagnetic Optimization Based on Gaussian Crow Search Approach[C]// 2018 International Symposium on Power Electronics, Electrical Drives, Automation and Motion. 2018: 1107-1112.

[5] PRATIWI A B. A hybrid cat swarm optimization - crow search algorithm for vehicle routing problem with time windows[C]// 2017 2nd International conferences on Information Technology,

Information Systems and Electrical Engineering. 2017.

[6] ASKARZADEH A. Capacitor placement in distribution systems for power loss reduction and voltage improvement: A new methodology[J]. IET Generation Transmission & Distribution, 2016, 10(14): 3631-3638.

[7] MANDALA J, RAO M V P C S. Privacy preservation of data using crow search with adaptive awareness probability[J]. Journal of Information Security and Applications, 2019, 44(FEB.): 157-169.

[8] LIU X J, HE Y C, WU C C, et al. Chaotic binary crow search algorithm for 0-1 knapsack problem[J]. Computer Engineering and Applications, 2018, 54(10): 173-179.

[9] ZHAO S J, GAO L F, YU D M, et al. Improved Crow Search Algorithm Based on Variable-Factor Weighted Learning and Adjacent-Generations Dimension Crossover Strategy[J]. Acta Electronica Sinica, 2019, 47(1): 40-48.

[10] SHI Z, LI Q, ZHANG S, et al. Improved Crow Search Algorithm with Inertia Weight Factor and Roulette Wheel Selection Scheme[C]// 2017 10th International Symposium on Computational Intelligence and Design (ISCID). 2017.

[11] DÍAZ P, PÉREZ-CISNEROS M, ERIK C, et al. An Improved Crow Search Algorithm Applied to Energy Problems[J]. Energies, 2018, 11(3): 571.

[12] ARORA S, SINGH H, SHARMA M, et al. A New Hybrid Algorithm based on Grey Wolf Optimization and Crow Search Algorithm for unconstrained function optimization and feature selection[J]. IEEE Access, 2019, 7: 26343-26361.

[13] RAMGOUDA P, CHANDRAPRAKASH V. Constraints handling in combinatorial interaction testing using multi-objective crow search and fruitfly optimization[J]. Soft Computing-A Fusion of Foundations, Methodologies and Applications, 2019, 23(8): 2713-2726.

[14] MOGHADDAM S, BIGDELI M, MORADLOU M, et al. Designing of standalone hybrid PV/wind/battery system using improved crow search algorithm considering reliability index[J]. International Journal of Energy and Environmental Engineering, 2019, 10(4): 429-449.

[15] STORN R, PRICE K. Differential Evolution-A Simple and Efficient Heuristic for global Optimization over Continuous Spaces [J]. Journal of Global Optimization, 1997, 11(4): 341-359.

[16] KENNEDY J, EBERHART R. Particle swarm optimization [C]// Proceedings of ICNN'95-International Conference on Neural Networks. IEEE, 1995.

[17] YANG X S. A New Metaheuristic Bat-Inspired Algorithm[J]. Computer Knowledge & Technology, 2010, 284: 65-74.

[18] FISTER I, FISTER I, YANGX S, et al. A comprehensive review of firefly algorithms[J]. Swarm and Evolutionary Computation, 2013, 13(Complete): 34-46.

[19] MIRJALILI S, MIRJALILI S M, LEWIS A. Grey Wolf Optimizer[J]. Advances in Engineering Software, 2014, 69(3): 46-61.

HUO Lin, born in 1964, professor, Ph.D supervisor, is a member of China Computer Federation. Her main research interests include information security, parallel distributed computing, artificial intelligence, and applied economics.