文章编号: 1001-0920 (2016) 06-0969-07 **DOI:** 10.13195/j.kzyjc.2015.0829

基于种群特征反馈的布谷鸟搜索算法

贾云璐, 刘 胜, 宋颖慧

(哈尔滨工程大学 自动化学院，哈尔滨 150001)

摘 要: 布谷鸟搜索 (CS) 算法是一种新型的生物启发式算法. 为了提高算法对不同优化问题的适应能力, 根据反馈控制原理提出一种基于种群特征反馈的布谷鸟搜索 (SFFCS) 算法, 将年龄结构、变异成功率等种群特征作为反馈信息引入算法框架, 动态调节算法参数, 同时引入双进化策略机制和策略选择概率, 加强算法对局部搜索和全局搜索的平衡能力. 对标准测试函数和电力系统最优潮流问题进行数值实验, 实验结果表明, SFFCS 算法具有较好的收敛性能和适应能力, 验证了所提出算法的有效性和工程应用价值.

关键词: 布谷鸟搜索算法；种群特征反馈；动态调节；双进化策略

中图分类号: TP399 **文献标志码:** A

Cuckoo search algorithm based on swarm feature feedback

JIA Yun-lu, *LIU Sheng*, *SONG Ying-hui*

(College of Automation，Harbin Engineering University，Harbin 150001，China．Correspondent：LIU Sheng，E-mail: liu.sch@163.com)

Abstract: Cuckoo search(CS) is a novel nature-inspired algorithm. For the sake of adaptation to various optimization problems, an improved CS algorithm is proposed, named swarm feature feedback cuckoo search(SFFCS) algorithm, on the basis of the feedback control principle. Swarm features such as age structure and success rate of mutation are introduced as feedback information for adjusting the parameters dynamically. Double evolutionary strategies and strategy selection probability are also introduced to balance the capability between local and global search. Numerical experiments on benchmark functions and the optimal power flow problem of the electrical system are conducted. The results indicate that the SFFCS algorithm behaves strong performance on convergence and adaptation, and show the effectiveness and practical engineering value of the proposed algorithm.

Keywords: cuckoo search；swarm feature feedback；dynamical adjustment；double evolutionary strategies

0 引 言

布谷鸟搜索算法 (CS) 是由剑桥大学的 Xin 等[1] 于 2009 年提出的一种新兴的生物启发式算法, 其思想源于某些布谷鸟寄生育雏习性, 并通过结合 Lévy Flight 模拟鸟类或果蝇觅食行为的方式加强了算法探索解空间的性能. CS 算法具有简单、参数少、易于实现等优点, 由于引入了 Lévy Flight 搜索方式, 该算法极易跳出局部极值点. 研究表明, 与典型群智能算法, 如粒子群算法 (PSO)、差分演化算法 (DE) 以及人工蜂群算法 (ABC) 相比, CS 算法具有更高的效率[2]. 目前, CS 算法已经被应用于多种工程优化问题.

CS 算法是一种全局最优搜索算法, 能在足够计算次数中达到全局最优, 但其搜索过程完全依赖于随机游走这一机制, 导致算法的快速收敛能力比较差, 众多学者针对这一问题进行了研究并提出了相应的改进版本. 文献 [3] 将 PSO 算法引入标准 CS 算法, 提出了混合 CS/PSO 算法, 实验结果表明, 混合算法的性能具有一定的竞争力; 文献 [4] 将 DE 算法与 CS 算法结合, 用于解决无人机三维路径规划问题, 结果表明, 在该问题上改进的算法具有较好的性能; 文献 [5] 在研究了算法参数与收敛性之间的关系后, 对标准 CS 算法的两个参数进行了改进, 提高了算法的收敛性能, 并成功应用于 PID 控制器设计中; 文献 [6] 提出令步长控制因子随种群迭代代数的增加而变

收稿日期: 2015-07-01；**修回日期:** 2015-09-23.

基金项目: 国家自然科学基金项目(51279036, 51307026, 51079033)；中央高校基本科研业务费专项资金项目 (HEUCFX41305).

作者简介: 贾云璐(1987−), 女, 博士生, 从事智能控制算法的研究；刘胜(1957−), 男, 教授, 博士生导师, 从事船舶运动姿态控制、先进控制理论等研究.

小, 并引入最优个体组合和黄金分割比替代偏好随机游走机制产生新解, 提高了算法在求解高维优化问题时的收敛速度; 文献[7]借鉴了迭代改进策略的思想, 提出了逐维改进的CS算法(DDICS), 在偏好随机游走过程中对候选解的每一维独立更新并评价, 避免了维间信息互相干扰, 实验结果表明, 对于大部分标准测试函数, 改进策略能够有效地提高算法的收敛速度和收敛精度; 文献[8]认为, 现有的CS算法及其变体从控制理论的角度而言, 其进化过程都是开环的, 对问题中的不确定性适应能力较差, 因此将种群改善率作为反馈参数建立了CS算法参数的闭环控制框架, 提出了动态适应布谷鸟算法(DACS), 以Rechenberg的1/5法则作为基准, 引入学习因子调节算法参数, 数值实验结果表明, DACS算法的收敛速度和收敛精度均超过标准CS算法和决定性动态CS算法, 但该算法引入了两个新的参数, 即步长控制因子和发现概率的学习因子, 这两个参数的选取也将影响算法的性能, 破坏CS算法参数少、简单的优点.

本文借鉴文献[8]的参数闭环控制思想, 将年龄结构、变异成功率等种群特征作为反馈信息引入CS算法, 提出种群特征反馈的布谷鸟搜索算法(SFFCS), 将反馈信息处理后直接作为学习因子调节算法参数, 并引入文献[7]中逐维改进思想, 与偏好随机游走形成双进化策略, 通过两种策略成功进化的比例调节个体选择进化策略的概率, 所提出的算法没有引入需要设置或调整的额外参数, 可保持CS算法参数少的特点. 数值实验结果表明, 对于大部分标准测试函数, SFFCS具有较快的收敛速度和较高的收敛精度.

1 布谷鸟搜索算法

Yang等[1,9]根据布谷鸟寄生育雏方式繁衍后代的行为提出了布谷鸟搜索算法. 布谷鸟搜索算法基于3条理想规则假设: 1) 每只布谷鸟每次只产1只卵, 并随机选择一个鸟巢存放; 2) 位置最好的鸟巢将会保留至下一代; 3) 可用鸟巢的数量n是固定的, 且宿主发现鸟巢中寄生卵的概率是$p_a \in [0,1]$. 通常情况下, 宿主发现鸟巢中的寄生卵后会选择将其推出巢外或者另建新巢, 为方便起见, 假设宿主选择另建新巢.

基于上述的基本假设，Yang采用D维向量$\boldsymbol{x} = [x_1, x_2, \cdots, x_D]$表示一个鸟巢位置, 即搜索空间中的一个候选解, 每个鸟巢位置都对应于一个适应度值$f(\boldsymbol{x})$, $f(\boldsymbol{x})$表示当前鸟巢位置的优劣. 布谷鸟首先通过Lévy Flight方式寻找下一个鸟巢位置, 即

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{\alpha} \oplus \boldsymbol{L}(\beta),\ i = 1, 2, \cdots, n. \tag{1}$$

其中: 符号"$\oplus$"表示点乘, $\boldsymbol{x}_i^{t+1}$和$\boldsymbol{x}_i^t$分别为第i个鸟巢在第t代和第$t+1$代中的位置, $\boldsymbol{L}(\beta)$符合D维莱维分布, $\boldsymbol{\alpha}$为步长控制因子. 为了能从当前最优解中获得更有用的步长信息, 采用下式计算步长控制因子:

$$\boldsymbol{\alpha} = \alpha_0(\boldsymbol{x}_i^t - \boldsymbol{x}_{\text{best}}). \tag{2}$$

其中: α_0为一个较小的常数, 通常取0.01; $\boldsymbol{x}_{\text{best}}$为当前最优解.

在第1步按照Lévy Flight进行位置更新后, 按概率p_a丢弃部分解, 算法采用偏好随机游走重新生成相同数量的新解, 即

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{r}(\boldsymbol{x}_j^t - \boldsymbol{x}_k^t). \tag{3}$$

其中: $\boldsymbol{r}$为符合(0,1)区间均匀分布的D维向量, $\boldsymbol{x}_j^t$和$\boldsymbol{x}_k^t$为第t代中两个随机解.

上述过程中产生的新解都采用贪心算法选择, 即得到新解后, 计算新解的适应度值并与原解的适应度值比较, 若新解优于旧解, 则替换旧解, 否则保留旧解.

2 基于种群特征反馈的CS算法

2.1 CS算法闭环控制

本文总结文献[8]的参数闭环控制思想, 将CS算法中的种群概念与经典闭环控制系统中的控制概念一一对应, 给出CS算法闭环控制框架, 如图1所示.

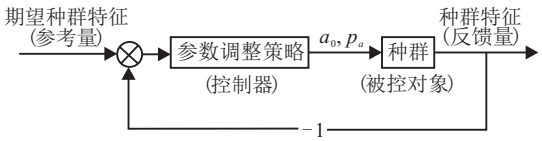

图1 CS算法闭环控制框架

从图1中可以看出, 该控制框架的核心在于选取合适易得的种群特征以及设计参数调整策略.

2.2 种群特征选取

在种群理论中, 种群年龄结构是衡量种群发展趋势的重要特征. 种群年龄结构是指种群中各年龄期个体数在种群中所占的比例. 当幼年个体数明显高于中老年个体数时, 种群的生产性高, 两者比值越大则生产性越高; 反之, 当幼年个体数低于中老年个体数之和时, 生产性低, 两者比值越小则生产性也越低[10]. 此外, 就遗传学角度而言, 种群个体的变异是种群多样性的来源, 根据"优胜劣汰"的自然选择规则, 变异后存活个体数越多, 表明此变异越有利于种群的进化, 变异成功个体数与变异未成功个体数比值越大, 对种群进化越有利.

将标准CS算法中种群个体通过Lévy Flight和宿主发现获得新解的过程分别视为遗传和变异过程. 若个体在自己位置的基础上, 通过Lévy Flight获得更好的新解, 则视为幼年个体, 否则为老年个体; 若个体脱离当代最优解, 通过宿主发现探索新领域获得更好的新解, 则视为变异成功个体, 否则为变异失败个体. 显然, Lévy Flight和宿主发现分别与参数α_0和p_a有关, 结合种群年龄结构及种群变异率对种群进化的影响, 可以采用幼年个体数与中老年个体数的比值R_a作为

反馈调整参数 α_0, 采用变异成功个体数与变异失败个体数比值 R_m 作为反馈调整参数 p_a.

2.3 双进化策略

标准CS算法在宿主发现的过程中采用偏好随机游走策略, 这种进化策略虽然能够提高种群的多样性, 从而提高算法的全局搜索能力, 但其高随机特性不利于算法的局部搜索. 文献[7]提出的DDICS算法在偏好随机游走过程中对候选解的每一维进行独立更新并评价, 即保持个体其余维上数值不变, 仅对其中一维按照下式进行更新:

$$x_{i,j}^{t+1} = x_{i,j}^{t} + r(x_{k,j}^{t} - x_{i,j}^{t}). \tag{4}$$

其中: r 取 $(-1, 1)$ 区间均匀分布的随机数, 变量下角标 j 表示对个体第 j 维进行更新, $x_{k,j}^{t}$ 为第 t 代中一个随机解的第 j 维. 若生成的新解优于旧解, 则保留, 否则丢弃, 而后进行下一维的更新及评价.

逐维改进策略对于变量相互独立的问题具有较好的局部搜索能力, 但在解决变量相互关联的问题上, 其能力却逊色于偏好随机游走策略. 因此, 本文同时采用偏好随机游走和逐维改进两种进化策略, 引入策略选择概率 p_c, 每个通过参与宿主发现过程生成新解的个体按概率选择不同的进化策略. 对于选择偏好随机游走策略的个体, 采用式(3)生成新解, 而选择逐维改进策略的个体则采用式(4)生成新解.

2.4 参数动态调整策略

1) 步长控制因子 α_0 调整策略.

步长控制因子 α_0 采用幼年个体数与中老年个体数的比值 R_a 调整. 当 $R_a > 1$ 时, 表明在当前步长控制因子 α_0 下, 算法通过Lévy Flight能以较大概率找到更优秀的新解, R_a 越大, 表明找到优秀新解的概率越大, 空间局部搜索性能越好, 为了提高搜索效率, 应该适当加大步长以提高种群的探索能力; 反之, 当 $R_a < 1$ 时, 表明在当前步长下, 算法找到优秀新解的概率较低, R_a 越小, 空间局部搜索性能越差, 应该适当缩小步长以提高空间的局部搜索能力. 因此, 采用下式调整参数 α_0:

$$\alpha_0^{t+1} = \alpha_0^{t} \times R_a. \tag{5}$$

注意, 在极端情况下, $R_a = 1$ 时, 表明幼年个体数与中老年个体数相同, 种群在当前步长下得到优秀解的概率为50%, 不适于根据当前的反馈信息调整参数 α_0.

2) 发现概率 p_a 调整策略.

发现概率 p_a 采用变异成功个体数与变异未成功个体数比值 R_m 调整. 当 $R_m > 1$ 时, 表明探索新领域的个体能够大概率找到更优秀新解, 新领域中优秀的解存在的可能性更大, 应当适当提高发现概率 p_a, 使更多个体探索新领域; 当 $R_m < 1$ 时, 表明新领域中存在优秀解的可能性较小, 为了提高效率, 可以降低发现概率 p_a, 节省目标函数的计算次数. 采用下式调整参数 p_a:

$$p_a^{t+1} = p_a^{t} \times R_m. \tag{6}$$

同理, 在极端情况下, $R_m = 1$ 时, 表明探索新领域的个体成功的概率为50%, 因此这种情况下改变参数 p_a 是不合适的. 此外, 需要注意 α_0 和 p_a 的范围以避免参数过调的情况.

3) 进化策略选择概率 p_c.

p_c 用于平衡两种进化策略, 初始值选择0.5, 记录分别参与偏好随机游走和逐维改进两种进化策略的个体变异成功率, 算法迭代过程中采用两种进化策略的个体成功率比值 R_c 更新 p_c, 即

$$p_c^{t+1} = p_c^{t} \times R_c. \tag{7}$$

当 $R_c > 1$ 时, 表明参与偏好随机游走进化策略的个体变异成功率高于参加逐维改进进化策略的个体变异成功率, 变异个体应该更多地参与到前者的进化策略中, 反之亦然. 为了避免种群陷入单一策略, 令 $0.01 < p_c < 0.99$.

2.5 SFFCS算法步骤

在算法运行开始前, 注意确定各参数的上下限以防止参数过调, 步骤如下.

Step 1: 初始化种群, 设置算法终止条件.

Step 2: 对于每个个体, 按照Lévy Flight, 即式(1)和(2)得到新解, 如果新解被保留下来, 则幼年个体数+1, 反之老年个体数+1, 计算 R_a.

Step 3: 生成[0,1]之间的随机数, 与 p_a 比较, 按照概率决定探索新领域的个体数, 记录每个个体变异状态, 1表示该个体参加变异.

Step 4: 生成[0,1]之间的随机数, 与 p_c 比较, 按照概率决定参与变异的个体遵循哪种进化策略, 按照式(3)或(4)生成新解, 如果新解被保留下来, 则成功个体数+1, 反之失败个体数+1, 分别计算两种策略的个体变异成功率后, 计算 R_c. 无论个体遵循哪种进化策略, 只要该个体有新解被保留下来, 即视为变异成功, 记录变异状态为1, 最后计算 R_m.

Step 5: 分别根据式(5)～(7)确定下一代种群的 α_0、p_a 和 p_c.

Step 6: 记录全局最优解, 如果不满足终止条件, 则重复Step 2～Step 5.

3 与其他改进CS算法的性能比较

3.1 标准测试函数

为了观察并验证所提出的改进算法在多维搜索空间中寻找目标函数最优值的能力, 采用一组测试函数进行数值实验, 测试函数细节如表1所示.

表 1 标准测试函数

函数描述	搜索范围	最优值
$f_1(x)=\sum_{i=1}^{D}x_i^2$	$[-100,100]$	0
$f_2(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}x_j\right)^2$	$[-100,100]$	0
$f_3(x)=\sum_{i=1}^{D-1}[100(x_{i+1}-x_i^2)^2+(1-x_i^2)]$	$[-30,30]$	0
$f_4(x)=-20\exp\left(-0.2\sqrt{\sum_{i=1}^{D}\frac{x^i}{D}}\right)-\exp\left(\sum_{i=1}^{D}\frac{\cos(2\pi x_i)}{n}\right)+20+\mathrm{e}$	$[-32,32]$	0
$f_5(x)=\frac{1}{4\,000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$[-600,600]$	0
$f_6(x)=\sum_{i=1}^{D}x_i^2-10\cos(2\pi x(i))+10$	$[-5.12,5.12]$	0
$f_7(x)=418.989\,2D-\sum_{i=1}^{D}(x_i\cdot\sin\sqrt{\lvert x_i\rvert})$	$[-500,500]$	3.818 e-04
$f_8(x)=\frac{\sin^2\sqrt{x_1^2+x_2^2}-0.5}{[1+0.001\times(x_1^2+x_2^2)]^2}+0.5$	$[-100,100]$	0

在表 1 中: f_1 为 Sphere 函数, f_2 为 Schwefel 1.2 函数, 这两个函数是单峰函数, 用于测试算法的收敛精度; f_3 为 Rosenbrock 函数, 当维数大于 3 时是多峰函数, 其全局极值位于陡峭的峡谷内; f_4 为 Ackley 函数, f_5 为 Griewank 函数, f_6 为 Rastrigin 函数, 这 3 个函数是具有多个局部极值点的多峰函数, 用于测试算法的全局搜索能力; f_7 为 Schwefel 2.6 函数, 也具有多个局部极值点, 但其全局极值点不位于零点, 能够避免算法可能存在的零点吸引子对性能的影响; f_8 为 Schaffer 函数, 震荡强烈, 很难找到全局最优值.

将所提出的 SFFCS 算法与基本 CS 算法、DACS 算法以及 DDICS 算法进行比较, 由于不同的算法在迭代过程中每一代种群生成新个体的个数不同, 以迭代次数为标准比较算法的收敛性能是欠公平的, 这里采用适应度计算次数 FC 作为算法进化的标准. 在本次实验中, 目标测试函数除 f_8 外均为 30 维, SFFCS 的参数为: 种群规模 $N=D=30$, 发现概率初值 $p_a=0.25$, $p_a\in[0,1]$, 步长控制因子初值 α_0、最小值 $\alpha_{0,\min}$ 和最大值 $\alpha_{0,\max}$ 分别为搜索范围的 1/200, 1/4 000 和 1/100. 其他算法的参数设置参见文献 [1,7-8].

分别以最大适应度计算次数 $\mathrm{FC}_{\max}=3\times10^5$ 和收敛误差 $\varepsilon=10^{-5}$ 作为算法运行的终止条件, 对 $f_1\sim f_8$ 分别独立运行 4 种算法 100 次.

3.2 解的质量分析

表 2 给出了 4 种算法在适应度计算次数 $\mathrm{FC}_{\max}=3\times10^5$ 时运算得到最优解的统计值.

在表 2 中, 黑色加粗部分表示对于每个测试函数, 4 种算法取得的质量最好的解的平均值 μ 及标准差 σ.

从表 2 数据可以看出, 对于大部分测试函数而言, DACS、DDICS 和 SFFCS 算法与标准 CS 算法相比, 都能取得较高精度的解: DACS 算法仅在 f_7 函数上比 CS 算法求得的解略差, 原因是 f_7 在较大搜索范围内分散分布了众多极值, 在迭代过程中绝大多数时间均以最小步长进行搜索, 没有发挥出算法的优势, 同时 DACS 算法的步长控制因子为 0.25, 大于 CS 算法的步长控制因子 0.01, 因此在搜索精度上不如 CS 算法; DDICS 算法对于变量相对独立的测试函数如 f_1、$f_4\sim f_7$ 都能取得相对较好质量的解, 这是因为逐维改进策略避免了个体维间的信息干扰, 加强了算法的局部搜索能力, 但是对于 f_2、f_3 和 f_8 这样变量之间相互关联的函数而言, 其解的质量较差, 这是由于 f_2、f_3 和 f_8 维间耦合关系令单维信息不能进行有效的搜索, 在每次迭代中消耗了较多的适应度计算次数却没有得到有效的进化; SFFCS 算法除了在 f_2 函数上精度略差于 DACS 算法外, 对于其余函数取得的解的质量都好于其他 3 种算法, 其中在 f_1 函数上 SFFCS 算法取得解的质量比 CS、DACS 和 DDICS 算法分别高出 47、13 和 2 个数量级, 并且在 $f_5\sim f_8$ 函数上取得了全局最优解.

表 2 适应度计算次数相同时最优解统计值

函数	统计值	算法			
		CS	DACS	DDICS	SFFCS
f_1	μ	9.025 e-29	3.527 e-63	8.592 e-74	**1.721 e-76**
	σ	7.014 e-29	2.007 e-63	6.508 e-74	**2.202 e-76**
f_2	μ	4.926 e-01	**9.523 e-14**	2.581 e+03	1.353 e-11
	σ	2.323 e-01	**7.574 e-14**	4.827 e+02	1.662 e-11
f_3	μ	8.795 e+00	6.609 e-07	6.547 e-02	**2.836 e-12**
	σ	2.015 e+00	3.820 e-07	3.602 e-02	**2.780 e-12**
f_4	μ	1.328 e+00	3.227 e-14	3.582 e-14	**2.516 e-14**
	σ	5.059 e+01	5.426 e-15	7.395 e-15	**4.102 e-15**
f_5	μ	1.211 e-15	**0**	**0**	**0**
	σ	2.098 e-15	**0**	**0**	**0**
f_6	μ	4.352 e+01	3.505 e+01	**0**	**0**
	σ	6.535 e+00	3.885 e+00	**0**	**0**
f_7	μ	1.942 e+03	3.036 e+03	**3.818 e-04**	**3.818 e-04**
	σ	7.935 e+02	5.190 e+01	**0**	**0**
f_8	μ	**0**	**0**	7.818 e-08	**0**
	σ	**0**	**0**	9.133 e-08	**0**

3.3 收敛速度分析

表 3 给出了 4 种算法在收敛误差为 $\varepsilon = 10^{-5}$ 时的适应度计算次数的统计值.

表 3　收敛误差相同时适应度计算次数的统计值

函数	统计值	算法			
		CS	DACS	DDICS	SFFCS
f_1	μ	7.846 e+04	4.270 e+04	4.568 e+04	**3.856 e+04**
	σ	9.910 e+02	5.206 e+02	3.920 e+02	**1.092 e+02**
f_2	μ	3.000 e+05	**1.502 e+05**	3.000 e+05	1.671 e+05
	σ	0	8.437 e+03	0	**1.109 e+03**
f_3	μ	3.000 e+05	2.750 e+05	3.000 e+05	**2.395 e+05**
	σ	0	**6.415 e+03**	0	1.691 e+04
f_4	μ	3.000 e+05	7.554 e+04	6.713 e+04	**6.219 e+04**
	σ	0	2.281 e+03	1.103 e+03	**6.194 e+02**
f_5	μ	1.469 e+05	8.021 e+04	**4.731 e+04**	4.981 e+04
	σ	1.346 e+04	4.049 e+04	4.179 e+03	**1.899 e+03**
f_6	μ	3.000 e+05	3.000 e+05	**7.916 e+04**	8.819 e+04
	σ	0	0	1.169 e+04	**8.523 e+03**
f_7	μ	3.000 e+05	3.000 e+05	1.331 e+05	**1.185 e+05**
	σ	0	0	9.665 e+03	**6.032 e+03**
f_8	μ	5.103 e+04	6.822 e+04	7.079 e+04	**2.252 e+04**
	σ	9.288 e+03	2.252 e+04	1.378 e+04	**6.820 e+03**

在表 3 中: $\mu = 3 \times 10^5$ 表示在最大适应度计算次数 $\mathrm{FC}_{\max}$ 内未收敛到指定误差; 黑色加粗部分表示对于每个测试函数, 4 种算法中通过最少的适应度计算次数得到的平均值 μ 及标准差 σ.

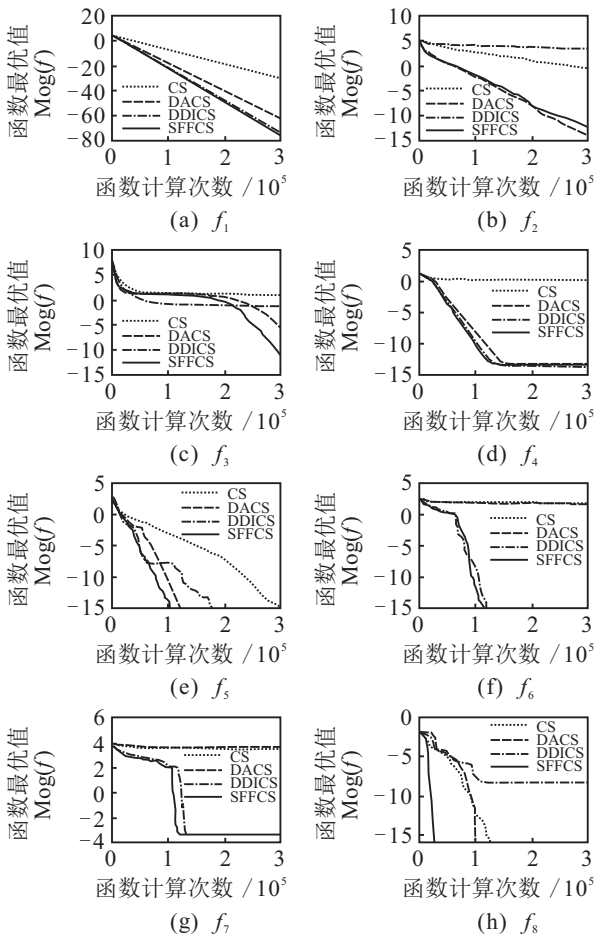

图 2　各算法求解 $f_1 \sim f_8$ 函数的收敛曲线

由表 3 可以看出, CS 算法在 $f_2 \sim f_4$、f_6 以及 f_7 函数上不能收敛到指定误差范围内, DACS 在 f_6 和 f_7 函数上不能收敛到指定误差范围内, DDICS 在 f_2 和 f_3 函数上不能收敛到指定误差范围内, SFFCS 在所有函数上均能收敛到指定误差范围内, 虽然在 f_2 函数上所用适应度计算次数略多于 DACS 算法以及在 f_5 和 f_6 函数上略多于 DDICS 算法, 但是 SFFCS 算法的标准差都小于相应的算法, 表明 SFFCS 算法比其他算法更稳定. 此外, 因为需要经历一次完整迭代后才能比较收敛误差, 统计适应度计算次数, 而每种算法一次迭代中计算目标函数次数不同, 所以也会产生误差. 总体而言, SFFCS 算法的收敛性能要优于参与测试的其他算法.

为了进一步说明 4 种算法的收敛速度, 并验证表 2 和表 3 的结果, 图 2 图形化展示了 4 种算法对于 8 个标准测试函数的收敛过程.

由图 2 可以看出, SFFCS 算法明显具有较好的收敛曲线. 虽然在 f_2 函数上 SFFCS 算法的性能略差于 DACS 算法, 但总体而言, SFFCS 算法优于其他算法, 具有较强的竞争力.

3.4 算法参数适应性分析

以函数 f_5 为例, 观察在搜索过程中, SFFCS 算法各参数的动态变化过程, 并与 DACS 算法进行比较. 图 3 和图 4 分别为 DACS 和 SFFCS 算法的 α_0 和 p_a 的动态变化, 可以看出两种算法参数动态变化趋势较为一致, 都能在函数最优值更新停滞后将 α_0 和 p_a 调节至最小值. 图 3 还给出了两种算法 α_0 在前 1 000 次适应度计算中的变化, 可以看出, SFFCS 算法调节参数 α_0 的快速性和稳定性要好于 DACS 算法.

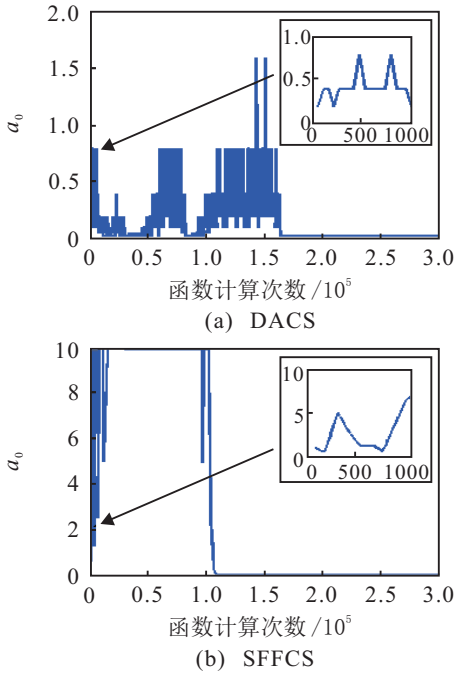

图 3　DACS 和 SFFCS 算法的 α_0 动态变化

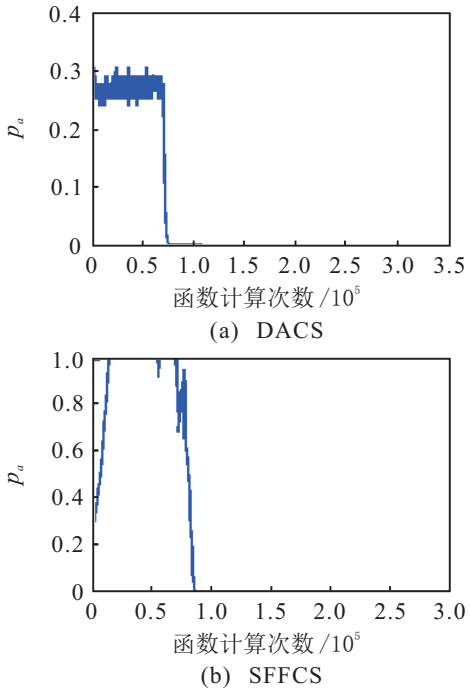

图 4 DACS 和 SFFCS 算法的 p_a 动态变化

图 5 为策略选择概率 p_c 的变化曲线.

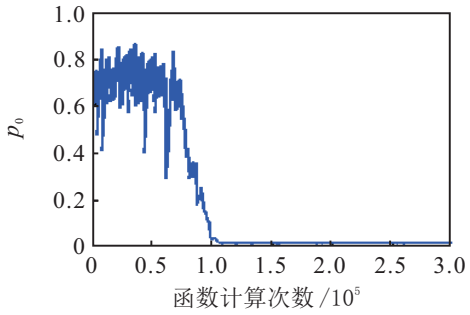

图 5 策略选择概率的 p_c 变化

结合图 2 (e)中 f_5 的 4 种算法的收敛曲线, 在约前 1.5 万次适应度的计算次数内, DACS 和 DDICS 算法的收敛速度相近, 该阶段的 p_c 在 0.5 左右变化, 表明选择偏好随机游走和逐维进化策略的粒子数目相近; 在 1.5～6 万次适应度的计算次数内, DDICS 算法的收敛性能好于 DACS 算法, 该阶段的 p_c 大于 0.5, 表明选择逐维进化策略的粒子数多于选择随机游走; 在 6～10 万次适应度计算次数内, DACS 算法的收敛性能好于 DDICS 算法, 该阶段的 p_c 变小, 表明越来越多的粒子选择偏好随机游走进化策略; 在 10 万次适应度计算次数后, 算法进入停滞期, p_c 也保持最小值进入停滞期.

4 与其他改进演化算法的性能比较

为了进一步说明 SFFCS 算法的优化性能, 本文将 SFFCS 算法与 v-PSO[11]、DP-DE[12]和 IGHS[13]三种改进演化算法进行比较, 限于篇幅, 仅选用 3.1 节标准测试函数中的 $f_2 \sim f_5$ 和 f_7 进行测试, 算法运行终止条件以及 SFFCS 算法的参数与 3.1 节相同, v-PSO、DP-DE 和 IGHS 三种算法的参数分别参见原文献. 分别独立运行 4 种算法 100 次, 表 4 给出了以适应度计算次数 $\mathrm{FC}_{\max} = 3 \times 10^5$ 为终止条件时各算法得到最优解的统计值, 表 5 给出了收敛误差为 $\varepsilon = 10^{-5}$ 时所需适应度计算次数的统计值.

表 4 适应度计算次数相同时最优解统计值

函数	统计值	算法			
		v-PSO	DP-DE	IGHS	SFFCS
f_2	μ	6.052 e-04	3.765 e-02	1.549 e+01	**1.353 e-11**
	σ	3.324 e-04	2.504 e-02	1.932 e+01	**1.662 e-11**
f_3	μ	1.266 e+00	2.956 e+06	6.547 e-02	**2.836 e-12**
	σ	1.408 e+00	1.488 e+04	3.602 e-02	**2.780 e-12**
f_4	μ	4.146 e-14	4.752 e-14	5.051 e-09	**2.516 e-14**
	σ	1.774 e-14	3.294 e-14	4.892 e-09	**4.102 e-15**
f_5	μ	4.102 e-04	1.235 e-04	5.518 e-02	**0**
	σ	3.269 e-03	7.367 e-03	4.935 e-02	**0**
f_7	μ	2.140 e+03	1.762 e+01	**3.818 e-04**	**3.818 e-04**
	σ	1.033 e+01	1.233 e+01	7.400 e-13	**0**

表 5 收敛误差相同时适应度计算次数的统计值

函数	统计值	算法			
		v-PSO	DP-DE	IGHS	SFFCS
f_2	μ	3.000 e+05	3.000 e+05	3.000 e+05	**1.671 e+05**
	σ	0	0	0	**1.109 e+03**
f_3	μ	3.000 e+05	3.000 e+05	3.000 e+05	**2.395 e+05**
	σ	0	0	0	**1.691 e+04**
f_4	μ	6.941 e+04	7.241 e+04	1.239 e+05	**6.219 e+04**
	σ	**5.596 e+02**	1.152 e+03	2.940 e+03	6.194 e+02
f_5	μ	3.000 e+05	3.000 e+05	3.000 e+05	**4.981 e+04**
	σ	0	0	0	**1.899 e+03**
f_7	μ	3.000 e+05	3.000 e+05	2.181 e+05	**1.185 e+05**
	σ	0	0	2.025 e+04	**6.032 e+03**

在表 4 和表 5 中: 当 $\mu = 3 \times 10^5$时, 表示在最大适应度计算次数 $\mathrm{FC}_{\max}$ 内未收敛到指定误差, 黑色加粗的数字表示 4 种算法中的最优统计值. 从统计结果可以看出, SFFCS 算法的收敛性能优于其他 3 种算法.

5 工程应用实例

SFFCS 算法中人为主观设置的参数少, 比较适合于工程实际应用. 电力系统的最优潮流问题是工程中的常见问题, 本质上是一个在满足电力系统安全运行约束条件下, 以经济运行为目标的复杂非线性优化问题. 以发电费用最小为目标函数, 其通用数学模型为

$$\begin{aligned} &\min\ f(\boldsymbol{x},\boldsymbol{u});\\ &\text{s.t.}\ \ \boldsymbol{g}(\boldsymbol{x},\boldsymbol{u})=\boldsymbol{0},\ \boldsymbol{h}(\boldsymbol{x},\boldsymbol{u})\leqslant\boldsymbol{0}. \end{aligned} \tag{8}$$

其中: $f(\boldsymbol{x},\boldsymbol{u})$ 为目标函数; $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{u})$ 为等式约束条件; $\boldsymbol{h}(\boldsymbol{x},\boldsymbol{u})$ 为不等式约束条件; $\boldsymbol{u}$ 为控制变量向量, $\boldsymbol{x}$ 为状态变量向量, 变量的物理意义参见文献[14].

应用 SFFCS 算法对 IEEE 30 节点系统进行多次最优潮流计算, 并与 DACS 算法以及文献[14]提出的 DE-BBO 算法进行比较. 设置最大适应度计算次数为 $\mathrm{FC}_{\max} = 1 \times 10^4$, DACS 和 DE-BBO 算法参数设置分

别参见文献[8]和文献[14], SFFCS 算法参数设置同 3.1 节, 3 种算法分别独立进行 50 次实验, 得到统计数据如表 6 所示.

表 6　IEEE 30 节点系统 50 次实验结果统计值 /(USD/h)

算法	最大值	最小值	平均值
DE-BBO	806.229	806.110	806.129
DACS	802.596	801.327	801.953
SFFCS	800.235	799.658	799.723

从表 6 结果可以看出, 本文提出的 SFFCS 算法得到的最大值、最小值以及平均值都要优于其他两种算法, 验证了 SFFCS 算法求解实际工程问题的能力.

6　结　论

布谷鸟搜索算法是一种新型群智能算法, 具有简单、易于实现、高效等优点. 本文将年龄结构、变异成功率等种群特征作为反馈信息引入到 CS 算法, 提出了种群特征反馈的布谷鸟搜索算法, 将反馈信息处理后直接作为学习因子动态调节算法参数, 采用偏好随机游走和逐维改进双进化策略, 通过两种策略成功进化的比例调节个体选择进化策略的概率, 以平衡算法局部搜索和全局搜索能力.

对标准函数的测试实验结果表明, 对于不同类型的函数, SFFCS 算法与标准 CS 算法相比, 能显著提高算法的收敛速度和解的质量, 与其他改进 CS 算法和演化算法相比, 具有更强的问题适应能力. 将 SFFCS 算法应用于 IEEE 30 节点系统的最优潮流计算问题, 数值结果表明, SFFCS 算法具有良好的优化性能以及工程实用价值.

所提出的 SFFCS 算法在进化过程中没有引入需要人为设置或调整的额外参数, 引入的额外操作均为简单运算, 对算法计算复杂度的影响有限, 保持了标准 CS 算法参数少、简单的优点, 无论在求解函数优化问题还是工程应用优化问题上均具有较强的竞争力.

参考文献(References)

[1] Xin She Y, Deb S. Cuckoo search via Lévy flights[C]. Proc of the World Congress on Nature & Biologically Inspired Computing. Coimbatore: IEEE Publications, 2009: 210-214.

[2] Civicioglu P, Besdok E. A conceptual comparison of the cuckoo search, particle swarm optimization, differential evolution and artificial bee colony algorithms[J]. Artificial Intelligence Review, 2013, 4(39): 315-346.

[3] Ghodrati A, Lotfi S. A hybrid CS/PSO algorithm for global optimization[C]. Intelligent Information and Database Systems. Heidelberg: Springer, 2012: 89-98.

[4] Wang G, Guo L, Duan H. A hybrid metaheuristic DE/CS algorithm for UCAV three-dimension path planning[J]. The Scientific World J, 2012, 2012(9): 2977-2991.

[5] Jin Q, Qi L, Jiang B, et al. Novel improved cuckoo search for PID controller design[J]. Trans of the Institute of Measurement and Control, 2015, 37(6): 721-731.

[6] Li Xiangtao, Yin Minghao. Modified cuckoo search algorithm with self adaptive parameter method[J]. Information Sciences, 2015, 298: 80-97.

[7] 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.
(Wang L J, Yin Y L, Zhong Y W. Cuckoo search algorithm with dimension by dimension improvement[J]. J of Software, 2013, 24(11): 2687-2698.)

[8] 张永韡, 汪镭, 吴启迪. 动态适应布谷鸟搜索算法[J]. 控制与决策, 2014, 29(4): 617-622.
(Zhang Y W, Wang L, Wu Q D. Dynamic adaptation cuckoo search algorithm[J]. Control and Decision, 2014, 29(4): 617-622.)

[9] Xin She Y, Deb S. Engineering optimization by cuckoo search[J]. Int J of Mathematical Modelling and Numerical Optimisation, 2010, 1(4): 330-343.

[10] Pathak R K, Gopesh A, Dwivedi A C. Age composition, growth rate and age pyramid of an exotic fish species, Cyprinus carpio var. communis from the Ganga river at Allahabad, India[J]. National Academy Science Letters — India, 2011, 34(5/6): 223-228.

[11] Pehlivanoglu Y V. A new particle swarm optimization method enhanced with a periodic mutation strategy and neural networks[J]. IEEE Trans on Evolutionary Computation, 2013, 17(3): 436-452.

[12] Zhong Jing-hui, Shen Meie. A differential evolution algorithm with dual populations for solving periodic railway timetable scheduling problem[J]. IEEE Trans on Evolutionary Computation, 2013, 17(4): 512-527.

[13] Ehsan Valian, Saeed Tavakoli, Shahram Mohanna. An intelligent global harmony search approach to continuous optimization problems[J]. Applied Mathematics and Computation, 2014, 232: 670-684.

[14] 李静文, 赵晋泉, 张勇. 基于改进差分进化——生物地理学优化算法的最优潮流问题[J]. 电网技术, 2012, 32(9): 115-119.
(Li J W, Zhao J Q, Zhang Y. Optimal power flow on basis of combining improved differential evolution algorithm with biogeography — based optimization algorithm[J]. Power System Technology, 2012, 32(9): 115-119.)

(责任编辑：闫　妍)