文章编号: 1001-0920 (2014) 04-0617-06 **DOI:** 10.13195/j.kzyjc.2012.1815

动 态 适 应 布 谷 鸟 搜 索 算 法

张永韡, 汪 镭, 吴启迪

(同济大学 电子与信息工程学院，上海 201804)

摘 要: 介绍一种新的生物启发算法——布谷鸟搜索 (CS) 及其相关的 Lévy 飞行搜索机制. 为了进一步提高算法的适应性, 将反馈引入算法框架, 建立了 CS 算法参数的闭环控制系统. 将 Rechenberg 的 1/5 法则作为进化的评价指标, 引入学习因子平衡种群的多样性和集中性, 提出动态适应布谷鸟算法 (DACS). 最后, 通过数值实验验证了所提出算法的有效性.

关键词: 布谷鸟算法；Lévy飞行；动态适应；反馈控制

中图分类号: TP399 **文献标志码:** A

Dynamic adaptation cuckoo search algorithm

ZHANG Yong-wei, *WANG Lei*, *WU Qi-di*

(School of Electronics and Information Engineering，Tongji University，Shanghai 201804，China．Correspondent: ZHANG Yong-wei，E-mail：yongwzhang@gmail.com)

Abstract: A novel bio-inspired algorithm, cuckoo search(CS), is introduced along with the related Lévy flight mechanism. In order to improve the adaptation of this algorithm, a feedback control scheme of algorithm parameters is adopted in CS. By utilizing Rechenberg's 1/5 criteria to evaluate evolution process, and introducing the learning factor, the diversification and intensification of population are well balanced. The dynamic adaptation cuckoo search(DACS) algorithm is proposed. Finally, numerical experiment results show the effectiveness of the proposed algorithm.

Key words: cuckoo search；Lévy flight；dynamic adaptation；feedback control

0 引 言

布谷鸟搜索 (CS) 是由 Yang 等[1]于 2009 年提出的一种新兴生物启发算法. CS 算法通过模拟某些种属布谷鸟的寄生育雏习性有效地求解最优化问题. 本文在 CS 算法中引入 Lévy 飞行来刻画布谷鸟的觅食动态, 使算法探索解空间的性能更高, 并能灵活地跳出局部极值. CS 算法的结构十分简单, 控制参数较少, 且具有较强的跳出局部极值的能力. 研究表明, CS 算法比遗传 (GA) 算法、人工蜂群 (ABC) 算法和粒子群 (PSO) 算法等典型群体智能算法具有更高的效率, 可在较少的函数求解次数 (FE) 下得到更好的优化结果[1-3]. 目前, CS 算法已经被应用于多种工程优化问题[3-4], 具有潜在的研究价值.

经过几年的发展, 为了进一步提高 CS 算法的性能, 很多变体与改进逐步涌现. 文献 [5] 提出了一种可变参数的改进 CS 算法, 提高了收敛速度, 并将该算法应用于前馈神经网络训练; 文献 [6] 将一种混合 CS 算法应用于流水车间调度问题求解; 文献 [7] 将集成了模糊系统的混合 CS 算法应用于机组组合问题; 文献 [8] 通过对种群分组, 并根据搜索的不同阶段对搜索步长进行预先设置, 提出了改进动态布谷鸟搜索 (MACS) 算法, 提高了 CS 算法的性能. 然而, 现有的针对算法适应性的改进均是在运行前确定的, 并不能动态地反馈搜索过程. 从控制理论的观点看, 这种预先计划算法参数的做法是开环的. 元启发算法的搜索过程因问题而异, 且具有很大的随机性, 显然不论固定参数还是预先计划的参数都不能很好地适应搜索空间中的不确定性. 为了进一步提高算法的适应性, 基于 Rechenberg 的 1/5 法则[9], 将种群改善率作为反馈引入算法框架, 以建立 CS 算法参数的闭环控制系统. 通过引入学习因子来平衡种群的多样性和集中性, 并提出了动态适应布谷鸟算法 (DACS).

收稿日期: 2012-12-06；**修回日期:** 2013-03-01.

基金项目: 教育部博士点基金项目 (20100072110038)；国家自然科学基金项目 (70871091, 61075064, 61034004, 61005090)；教育部新世纪人才计划项目 (NECT-10-0633).

作者简介: 张永韡(1983−), 男, 博士生, 从事自然计算、群体智能的研究；汪镭(1970−), 男, 教授, 博士生导师, 从事群体智能、智能控制等研究.

1 布谷鸟算法

布谷鸟最特殊的习性是寄生育雏[10]. 某些种属的布谷鸟将自己的卵偷偷产入宿主巢穴, 由于布谷鸟后代的孵化时间比宿主的幼雏早, 孵化的幼雏会本能地破坏同一巢穴中其他的卵 (推出巢穴), 并发出比宿主幼雏更响亮的叫声. 很多宿主通过后代的叫声大小判断其健康程度, 而健康后代获得的食物较多, 进而拥有更高的存活率. 在某些情况下, 宿主也会发现巢穴中的陌生卵. 这时, 宿主将遗弃该巢穴, 并选择其他地方重新筑巢. 在与宿主不断的生存竞争中, 布谷鸟的卵和幼雏叫声均朝着模拟宿主的方向发展, 以对抗宿主不断进化的分辨能力[10].

Yang[11]使用 D 维向量 $\boldsymbol{x}=[x_1,x_2,\cdots,x_D]$ 表示每一个卵或布谷鸟, 并使用两种方法产生后代. 第 1 种使用 Lévy 飞行, 有

$$\boldsymbol{x}^{t+1}=\boldsymbol{x}^t+\alpha\cdot\boldsymbol{s}_L\otimes(\boldsymbol{x}_i^t-\boldsymbol{x}_b)\otimes\boldsymbol{r}_n. \tag{1}$$

其中: $\boldsymbol{s}_L\sim L^D(\lambda)$, $L^D(\lambda)$ 为 D 维 Lévy 分布; $\boldsymbol{r}_n\sim N^D(0,1)$, $N^D(0,1)$ 为 D 维正态分布; α 为由待求解问题决定的步长因子; $\boldsymbol{x}_b$ 为历史最优解. Lévy 飞行是一种随机运动, 随着迭代次数的增加, 其方差与迭代次数存在如下关系:

$$\sigma^2(t)\sim t^{3-\lambda},\ 1<\lambda<3. \tag{2}$$

布朗运动的方差与迭代次数成线性关系, 即 $\sigma^2(t)\sim t$[12]. 通过将以上两种关系进行比较可知, 在探索大范围空间时, Lévy 飞行比布朗运动具有更高的效率.

由于分布是幂函数, Lévy 分布具有无限方差[13]且增量服从重尾分布. 从表面上看, Lévy 飞行会在看似布朗运动的状态下突然进行一次远距离的飞行. 或者说, Lévy 飞行由频繁的短跳跃与偶然出现的长跳跃组成. 与布朗运动不同的是, Lévy 飞行可减少过采样, 使个体在食物稀缺的环境中高效地觅食[14]. 二维情况 Lévy 飞行是由 Mandelbrot[15]首先描述的. 文献 [16] 证明了许多动物与昆虫的觅食特性表现出了 Lévy 飞行性质, 而文献 [17] 证明了鲨鱼的觅食习性也服从 Lévy 分布. 2 万步的二维 Lévy 飞行与布朗运动的对比如图 1 所示, 其中局部放大部分为 Lévy 飞行在 1.7 万至 2 万步时的情况.

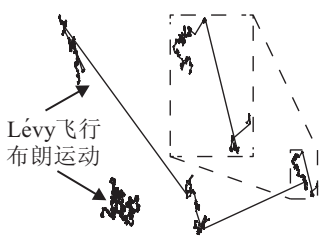

图 1 Lévy 飞行与布朗运动

另一种产生新解的方法是使用种群之间的相似性及发现概率 p_a, 其产生公式如下:

$$\boldsymbol{x}_i^{t+1}=\boldsymbol{x}_i^t+r_u\cdot(\boldsymbol{x}_j^t-\boldsymbol{x}_k^t)\otimes H(p_a-\boldsymbol{r}_u). \tag{3}$$

其中: $r_u\sim U(0,1)$; $\boldsymbol{r}_u\sim U^D(0,1)$, $U^D(0,1)$ 为定义在 [0,1] 上的均匀分布; $\boldsymbol{x}_i$, $\boldsymbol{x}_j$ 和 $\boldsymbol{x}_k$ 为随机选择且互不相同的解; $H(\cdot)$ 为 Heaviside 函数. 产生的新解采用贪心法选择, CS 算法的流程如下.

Step 1: 初始化种群, 每个 D 维向量代表一个巢穴或卵, 计算所有个体的适应度.

Step 2: 对每个巢穴, 按照式 (1) 产生新解, 如果新解优于旧解, 则替换旧解.

Step 3: 对每个巢穴, 任选其他两个与之不同的巢穴, 对向量中的每个元素按照式 (3) 进行组合. 若新解优于旧解, 则替换旧解.

Step 4: 记录最优解, 若不满足终止条件则返回 Step 2.

2 动态适应布谷鸟算法

2.1 种群特征的闭环控制策略

在 CS 算法中, 控制算法动态的是式 (1) 和 (3) 描述的新解生成机制. 尽管下一代解可反映种群结构及问题形态, 但这种反映是被动和机械的. 为了主动地适应算法动态, 有必要引入一些控制种群特征的方法. 动态算法对搜索过程的适应可看作是算法动态或种群特征的闭环控制过程. 其中以环境信息或种群特征为反馈量, 期望种群特征为参考量, 算法调整策略为控制策略, 而受控对象则是由算法的固有结构和搜索空间共同决定的动力学系统.

制定种群特征的闭环控制策略, 首先需要明确以下几点: 1) 选择什么种群特征作为反馈; 2) 期望的种群特征; 3) 种群特征的控制量; 4) 控制量的调整策略. 此外, 种群特征及控制量的计算代价应该尽可能小, 且控制量应该是易于实施的对象. Rechenberg 在研究进化计算的参数自动整定时提出了 1/5 原则[9], 原则认为“所有变异的成功比例应该保持在 1/5”. 也就是说, 算法的控制参数应该随着新解成功的比例 (改善率) 动态调整, 使改善率保持在 1/5. 由 CS 解的生成机制可以看出, 有两个参数在控制后代的特性: 步长因子 α 和发现概率 p_a. 综上所述, 可选择改善率作为反馈, 0.2 为改善率的期望值 (参考量), 步长因子 α 和发现概率 p_a 作为控制量. 控制量的调整策略将在下一节详细讨论.

2.2 算法控制量调整策略

文献 [18] 总结了进化算法中的 3 种动态参数类型. 1) 决定性动态. 这种模式下算法参数的变化在运行前就已经确定, 例如文献 [8] 对原始 CS 算法的改进以及 PSO 算法中的惯性权重[19]等. 2) 适应性动态. 这

种模式下算法参数根据种群的反馈作出调整. 3) 自适应动态. 这种模式下的算法参数直接编码在种群中, 与种群一同进化. 使用算法参数控制种群改善率的闭环控制过程实际上是参数的适应性动态调整过程.

对于步长因子 α, 按照 Rechenberg 原则, 确定了 3 种情况的调整策略: 1) 改善率大于 0.2. 表明搜索空间相对平滑和单调, 在当前步长影响下, 算法以较大的概率找到更优秀的解; 同时表明, 当前步长对搜索空间局部区域的开发程度较高. 为了提高算法的搜索效率, 减少目标函数的计算次数, 应该增加种群的探索性和搜索范围, 此时可适当增大步长. 2) 改善率小于 0.2. 此时, 搜索空间相对于步长较为复杂, 找到更好解的概率降低. 对此, 应增加种群对搜索空间的开发性, 提高算法的搜索精度, 适当减小步长. 3) 改善率等于 0.2. 表明当前步长恰好使种群的改善率维持在最佳值, 不需要进行调整. 然而, 改善率恰好为 0.2 的情况在实际搜索过程中并不多见, 这使得待整定参数频繁在较大范围内改变. 为了参数自动整定的稳定性, 在原 1/5 原则的基础上, 将参数维持不变的范围从 0.2 扩展到 [0.2, 0.3] 区间, 基于修正的 Rechenberg 原则的步长因子整定策略如下:

$$\alpha^{t+1}=\begin{cases}\alpha^t\cdot f_\alpha,\ R>0.3;\\ \alpha^t,\ 0.2\leqslant R\leqslant 0.3;\\ \alpha^t/f_\alpha,\ R<0.2.\end{cases}\tag{4}$$

其中: R 为新解改善的比例, f_α 为步长因子的学习因子, 用来控制算法适应环境的主动程度. 类似地, 发现概率也使用修正的 Rechenberg 原则进行修正.

$$p_a^{t+1}=\begin{cases}p_a^t\cdot f_p,\ R>0.3;\\ p_a^t,\ 0.2\leqslant R\leqslant 0.3;\\ p_a^t/f_p,\ R<0.2.\end{cases}\tag{5}$$

其中 f_p 为发现概率的学习因子. 注意运行开始前应该确定 α 与 p_a 的上下限以防止参数的过调. 动态适应布谷鸟搜索(DACS)算法的流程如下.

Step 1: 初始化种群, 计算所有个体的适应度.

Step 2: 对每个个体, 按照式 (1) 产生新解, 若新解优于旧解, 则替换旧解, 同时改善次数 +1.

Step 3: 对每个个体, 按照相似度和发现概率产生新解, 若新解较优, 则替换旧解, 同时改善次数 +1, 已经改善的解不再重复计数.

Step 4: 计算种群中改善个体的比例 R, 并按照式 (4) 和 (5) 确定下一代种群的步长因子和发现概率.

Step 5: 记录最优解. 若终止条件不满足, 则重复 Step 2 ~ Step 4.

2.3 算法时间复杂度分析

按照基本 CS 算法流程, 在初始化阶段, 假设产生均匀分布随机数的执行时间为 c_1, 计算给定解的目标函数的执行时间是变量数 n 的函数 $f(n)$, 则时间复杂度为 $O(N(c_1n+f(n)))=O(n+f(n))$, 其中 N 为种群规模. 在第 1 个新解生成阶段, 式 (1) 生成新解中一个变量的执行时间为 c_2, 比较新解与旧解和替换一个旧解变量的执行时间分别为 c_3 和 c_4, 时间复杂度为 $O(N(c_2n+f(n)+c_3+c_4n))=O(n+f(n))$. 第 2 个新解生成阶段每个变量的执行时间为 c_5, 比较新解与旧解和替换旧解的执行时间与第 1 个新解生成阶段一样, 时间复杂度为 $O(N(c_5n+f(n)+c_3+c_4n))=O(n+f(n))$. 比较一个解与最优解的执行时间是 c_3, 替换一个最优解变量的执行时间是 c_4, 则记录最优解的时间复杂度为 $O(N(c_3+c_4n))=O(n)$. 对于每一代, 总的时间复杂度为

$$T(n)=3O(n+f(n))+O(n)=O(n+f(n)).\tag{6}$$

由此可见, 基本 CS 算法的时间复杂度取决于计算目标函数的时间复杂度 $f(n)$, 当 $f(n)$ 为比 n 高阶的运算时, 算法执行一代的复杂度为 $O(f(n))$, 当 $f(n)$ 与 n 同阶或低阶时, 复杂度为 $O(n)$. 若以最大目标函数计算次数 $\mathrm{FE}_{\max}$ 为终止条件, 则算法的总体时间复杂度最大为

$$T(n)=O(L(n+f(n)))=O(n+f(n)),\tag{7}$$

其中 $L=(\mathrm{FE}_{\max}-N)/2N$ 为总的进化代数. 同理, 考察 DACS 算法流程, 设改善次数 +1 的执行时间为 c_6, Step 2 的时间复杂度为

$$O(N(c_2n+f(n)+c_3+c_4n+c_6))=O(n+f(n)),$$

Step 3 的时间复杂度为

$$O(N(c_5n+f(n)+c_3+c_4n+c_6))=O(n+f(n)),$$

可以看出 Step 1 ~ Step 3 的时间复杂度与基本 CS 算法一致, 改善次数 +1 增加的时间复杂度对最终的复杂度分析无影响. 对于 Step 4, 假设计算改善率的执行时间为 c_7, 式 (4) 和 (5) 的执行时间总和为 c_8, 则 Step 4 的时间复杂度为 $O(c_7+c_8)$, 最终的时间复杂度为 $O(n+f(n))+O(c_7+c_8)=O(n+f(n))$, 与基本 CS 算法一致.

3 仿真与比较

3.1 测试函数

为了验证本文算法对于不同问题的性能, 使用表 1 所示的标准测试函数进行数值实验. 其中: f_1 为 Schwefel 1.2 函数, f_2 为 Rosenbrock 函数, 这两个函数都是单模不可分离函数, f_2 的全局极值位于陡峭的峡谷, 大多数搜索算法很难在峡谷内获取正确的搜索方

向; f_3 为 Schwefel 2.6 函数, f_4 为 Griewank 函数, f_5 为 Ackley 函数, 3 者均为多模函数, 具有多个局部极值, f_4 和 f_5 的变量之间相互关联, 使优化算法很难搜索到全局最优解. 为了避免算法中可能存在的零点吸引子对性能的影响, 除了全局最优解不位于零点的 Schwefel2.6 函数, 其他函数均在自变量每一维增加一个偏移量. 这些函数的表达式、搜索上下限和偏移量如表 1 所示, 测试函数均为 16 维.

表 1 测试函数

函 数	搜索范围	偏移
$f_1=\sum_{i=1}^{D}\Big(\sum_{j=1}^{i}x_j\Big)^2$	$[-64,64]$	16
$f_2=\sum_{i=1}^{D-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	$[-30,30]$	0.512
$f_3=\sum_{i=1}^{D}(418.982\,9-x_i\sin(\sqrt{\lvert x_i\rvert}))$	$[-500,500]$	0
$f_4=\frac{1}{4\,000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\Big(\frac{x_i}{\sqrt{i}}\Big)+1$	$[-600,600]$	150
$f_5=-20\exp\Big(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\Big)-\exp\Big(\frac{1}{D}\sum_{i-1}^{D}\cos(2\pi x_i)\Big)-20+e$	$[-32,32]$	8

3.2 与基本 CS 算法和 MACS 算法的比较

由于不同的算法采用不同的种群规模, 在每一代中探索可行空间的次数不同, 如果以进化代数描绘收敛轨迹, 则无法进行横向比较. 为了公平起见, 实验中使用目标函数计算次数 FE 来衡量进化过程. 对于基本 CS 算法, 使用文献 [1] 中建议的参数: $N=25$, $p_a=0.25$, 并使用下式确定 Lévy 飞行步长系数:

$$\alpha=(b_u-b_l)/200. \tag{8}$$

对于 MACS 算法, 子群数量 $N=4$, 子群规模 $N_e=20$. 对于 DACS 算法, 种群规模 $N=25$, 发现概率 p_a 初始值为 0.25, 学习因子 $f_p=1.05$, $p_{\max}=1$, $p_{\min}=0$. 步长 α 的初始值为搜索空间大小的 1/1 000, 学习因子 $f_\alpha=2$, $\alpha_{\max}$ 与 $\alpha_{\min}$ 由下式决定:

$$\alpha_{\max}=(b_u-b_l)/100, \tag{9a}$$

$$\alpha_{\min}=(b_u-b_l)/4\,000, \tag{9b}$$

其中 b_u 和 b_l 为求解问题的可行区间上限和下限. 注意, 为了保证发现概率的有效性, 本文约定在使用式 (3) 生成新解时, 即使 $p_a=0$, 也保证至少有一个元素发生改变.

按照上述设置进行实验, 对 $f_1\sim f_5$ 进行 100 次运行, $\mathrm{FE}_{\max}=2\times10^5$, 以获得统计上有效的数据, 记录到达 $\mathrm{FE}_{\max}$ 时所有运行的最佳值及最佳值的标准差. 实验结果如表 2 和表 3 所示.

表 2 100 次运行中的全局最优解

算法	f_1	f_2	f_3	f_4	f_5
CS	4.317 0e-6	3.490 6	1.717 4e-3	5.781 6e-7	15.977 9
MACS	2.030 7e-7	0.489 5	1.383 3e-3	3.0335e-8	16.556 8
DACS	**0.00**	**0.001 8**	**2.036 5e-4**	**0.00**	**1.065 8e-14**

表 3 100 次运行中的解的标准差

算法	f_1	f_2	f_3	f_4	f_5
CS	1.030 7e-5	82.476 8	504.001 5	0.215 5	0.521 2
MACS	3.9154e-7	104.550 5	518.567 7	0.149 9	0.479 2
DACS	**0.00**	**0.672 2**	**56.907 4**	**0.001 8**	**3.065 2e-15**

由表 1 和表 2 可以看出, DACS 算法在所有测试函数中的表现均远优于 CS 算法和 MACS 算法. 其中: f_1 和 f_4 均找到了全局最优解 0; 对于 f_2, f_3 和 f_5, DACS 算法所找到的最优解相对于原始 CS 算法降低了 4, 7 和 15 个数量级. 不论对于单峰函数 f_1 和 f_2, 还是对于多峰函数 $f_3\sim f_5$, DACS 算法找到的最优解均优于其他参与测试的算法. 各算法运行的成功率如表 4 所示, 其中成功率使用下式判定:

$$\mathrm{Src}=\frac{N_{\mathrm{success}}}{N_{\mathrm{trails}}}, \tag{10a}$$

$$|\mathrm{best}_g-f_*|<\epsilon. \tag{10b}$$

当一次运行结束时, 若全局最优 best_g 满足 (10b), 则视为一次成功. 为了每个算法对每个测试问题都显示有意义且有区别的成功率 (而非全为 100% 或者 0), 针对不同的测试问题选择了适当的 ϵ 值. 过小的 ϵ 值将导致所有算法的成功率均为 0, 反之, 过大的 ϵ 值则会导致所有算法的成功率均为 100%. 由表 4 可以看出, 尽管 DACS 算法的成功率总为 100%, 但所有对比都是在同等实验参数下进行的, 因此实验结果仍可以有效地反映 DACS 算法优于其他测试算法的性能.

表 4 100 次运行后各算法的成功率及采用的 ϵ 值

算法	f_1	f_2	f_3	f_4	f_5,
CS/%	74.0	66.0	73.0	93.0	62.0
MACS/%	100.0	78.0	79.0	97.0	76.0
DACS/%	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
ϵ	3.00e-5	10.00	3 000.00	0.50	19.0

图 2 为 3 种算法在各测试函数下最好函数值的中位数, 其中: 实线表示 DACS 算法, 虚线表示 MACS 算法, 点线表示 CS 算法, DACS 算法标注了相应的分位数图 (0.025, 0.25, 0.5, 0.75, 0.975), 以显示运行的稳定性.

由图 2 可以看出, DACS 算法的平均表现远优于 MACS 算法和 CS 算法. 在搜索的所有阶段, 即使最差的表现也优于对比算法的平均性能. 表 5 为基本 CS 和 DACS 算法对各个问题的单次平均运行时间对比. 由表 5 可以看出, 尽管在 DACS 算法中添加了种群改善率反馈和算法参数调整策略, 除了 f_1 和 f_3 在运行时间上略微增加外, 其他 3 个问题的运行时间反而有了小幅下降. 这是因为当改善率低于 0.2 时, 发现概率 p_a 逐步减小, 使得需要进行变异操作的参数变少, 总体上降低了操作时间. 而基本 CS 算法没有参数

调整策略, 需要变异的概率保持不变, 因此在平均改善率较低的情况下运行时间略长于DACS算法.

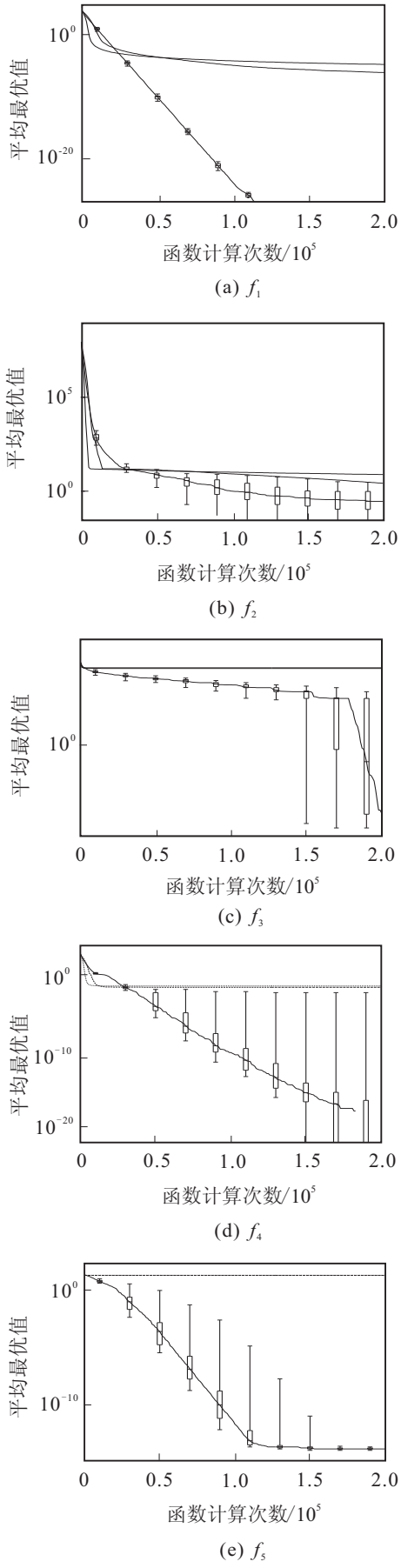

图 2　各算法在 $f_1 \sim f_5$ 上的收敛曲线

表 5　DACS 算法与 CS 算法的单次平均运行时间　s

算法	f_1	f_2	f_3	f_4	f_5
CS	29.966 5	16.645 5	15.289 2	26.166 2	16.604 3
DACS	34.821 3	15.915 0	15.463 6	25.948 1	16.370 0

3.3　发现概率 p_a 与步长 α 的适应性分析

图 3 为发现概率 p_a 与步长 α 在迭代过程中的变化曲线, 测试函数为 30 维 Ackley 函数. 由图 3 可以看出, 两个参数均具有 3 个显著的变化阶段: 探索期、收敛期和停滞期.

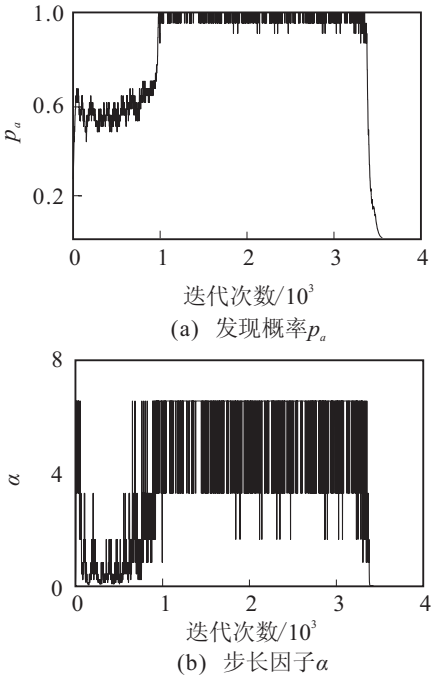

图 3　发现概率与步长因子变化曲线

探索期为前 1 000 次迭代, 对应前 5 万次函数计算. 在这个阶段, DACS 算法自适应地选择中等水平的发现概率和较小的搜索步长. 这与前期使用较大步长和较大发现概率的一般经验不符, 体现了算法对环境的适应性以及对搜索过程的学习能力. 对比图 2(e) 的误差条可以发现, 这一阶段函数值下降较为稳定.

收敛期为大约 1 000 ~ 3 500 次迭代, 对应 5 万 ~ 17.5 万次函数计算. 在这个阶段, DACS 算法自适应地选择较大的发现概率, 而且步长不断在两个值之间切换. 这种表现也同样与一般经验中的参数整定原则不符.

停滞期为 3 500 次迭代之后, 对应后 2.5 万次函数计算. 在这个阶段, 因为种群改善的比例长时间低于 1/5, 发现概率和步长均自适应地降低到最小值. 对比收敛曲线可以看出, 这个阶段函数值不再有明显的变化, 且误差条收缩到很短, 表明所有实验均收敛到相近的数值, 具有较高的稳定性.

4　结　论

适应性是生物在进化过程中与自然界相互作

用的结果. 在基于生物习性的生物启发算法研究中, 适应性也得到了越来越多的重视. 为了进一步改善算法在搜索过程中的适应性, 在决定性动态布谷鸟(MACS)算法的基础上, 引入步长学习因子和发现概率学习因子, 使用Rechenberg的1/5原则作为算法适应性的评价指标, 利用反馈控制原理在搜索过程中动态适应的调整算法参数, 提出了动态适应布谷鸟(DACS)算法.

实验证明, DACS算法的收敛速度和收敛精度均超过原始CS算法和决定性动态CS算法. 对发现概率P_a和搜索步长α的动态分析体现了算法在搜索过程中学习和适应的能力. 相对于决定性动态调整算法参数, 适应性动态更加灵活, 其实现不增加算法复杂度, 且具有通用性, 是一种高效简便的参数调整策略.

参考文献(References)

[1] Yang X, Deb S. Cuckoo search via Lévy flights[C]. World Congress on Nature & Biologically Inspired Computing. Piscataway: 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, 2011, 39(4): 315-346.

[3] Yang X S, Deb S. Engineering optimization by cuckoo search[J]. Int J of Mathematical Modelling and Numerical Optimization, 2010, 1(4): 330-343.

[4] Gandomi A, Yang X, Alavi A. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems[J]. Engineering with Computers, 2013, 29(29): 17-35.

[5] Valian E, Mohanna S, Tavakoli S. Improved cuckoo search algorithm for feed forward neural network training[J]. Int J of Artificial Intelligence & Applications, 2011, 2(3): 36-43.

[6] Marichelvam M K. An improved hybrid cuckoo search(IHCS) metaheuristics algorithm for permutation flow shop scheduling problems[J]. Int J of Bio-Inspired Computation, 2012, 4(4): 200-205.

[7] Chandrasekaran K, Simon S P. Multi-objective scheduling problem: Hybrid approach using fuzzy assisted cuckoo search algorithm[J]. Swarm and Evolutionary Computation, 2012, 5(9): 1-16.

[8] Zhang Y, Wang L, Wu Q. Modified adaptive cuckoo search algorithm and formal description for global optimization[J]. Int J of Computer Applications in Technology, 2012, 44(2): 73-79.

[9] Bäck T. Evolutionary algorithms in theory and practice: Evolution strategies, evolutionary programming, genetic algorithms[M]. Oxford: Oxford University Press, 1996: 161-164.

[10] Winfree R. Cuckoos, cowbirds and the persistence of brood parasitism[J]. Trends in Ecology & Evolution, 1999, 14(9): 338-343.

[11] Yang X. Cuckoo search for inverse problems and simulated-driven shape optimization[J]. J of Computational Methods in Sciences and Engineering, 2012, 12(1): 129-137.

[12] Yang X. Engineering optimization: An introduction with metaheuristic applications[M]. USA: John Wiley & Sons, 2010: 153-169.

[13] Pavlyukevich I. Lévy flights, non-local search and simulated annealing[J]. J Of Computational Physics, 2007, 226(2): 1830-1844.

[14] Viswanathan G M. Fish in Lévy-flight foraging[J]. Nature, 2010, 465(7301): 1018-1019.

[15] Mandelbrot B B. The fractal geometry of nature[M]. New York: W H Freeman and Co, 1983: 232-243.

[16] Brown C, Liebovitch L S, Glendon R. Lévy flights in Dobe Ju/'hoansi foraging patterns[J]. Human Ecology, 2007, 35(1): 129-138.

[17] Humphries N E, Queiroz N, Dyer J R M, et al. Environmental context explains Lévy and Brownian movement patterns of marine predators[J]. Nature, 2010, 465(7301): 1066-1069.

[18] Hinterding R, Michalewicz Z, Eiben A E. Adaptation in evolutionary computation: A survey[C]. Proc of IEEE Int Conf on Evolutionary Computation. Piscataway: IEEE Press, 1997: 65-69.

[19] Shi Y, Eberhart R. A modified particle swarm optimizer[C]. Proc of IEEE Int Conf on Evolutionary Computation. New York: IEEE Press, 1998: 69-73.

(责任编辑: 滕 蓉)