软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2021,32(5):1480−1494 [doi: 10.13328/j.cnki.jos.006056]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

具有万有引力加速机理的布谷鸟搜索算法*

傅文渊[1,2,3]

[1](华侨大学 信息科学与工程学院,福建 厦门 361021)

[2](厦门市专用集成电路系统重点实验室(华侨大学),福建 厦门 361008)

[3](福建省电机控制与系统优化调度工程技术研究中心,福建 厦门 361002)

通讯作者: 傅文渊, E-mail: fwy@hqu.edu.cn

摘 要: 为了解决布谷鸟搜索算法收敛速度较低、全局收敛效率不高的问题,提出了具有万有引力加速机理的布谷鸟算法.该算法基于万有引力搜索无需学习外部环境因素的变化亦能感知全局最优的特点,将布谷鸟巢穴等价为不同质量的个体,使其在优化过程中不仅遵循 Levy 飞行规律,而且遵循万有引力定律.不仅利用布谷鸟巢穴间存在的万有引力进行加速搜索,而且提出了一种概率变异的方法,增大了种群多样性,有效地平衡了算法的全局搜索能力和局部开采能力,提高了算法的全局搜索效率和收敛精度.通过算法的数学机理分析和26个基准测试函数实验结果表明,所提出的算法与其他改进智能优化算法比较,具有更优的性能.

关键词: 布谷鸟搜索算法;加速度;万有引力;发现概率

中图法分类号: TP18

中文引用格式: 傅文渊.具有万有引力加速机理的布谷鸟搜索算法.软件学报,2021,32(5):1480−1494. http://www.jos.org.cn/1000-9825/6056.htm

英文引用格式: Fu WY. Cuckoo search algorithm with gravitational acceleration mechanism. Ruan Jian Xue Bao/Journal of Software, 2021,32(5):1480−1494 (in Chinese). http://www.jos.org.cn/1000-9825/6056.htm

Cuckoo Search Algorithm with Gravitational Acceleration Mechanism

FU Wen-Yuan[1,2,3]

[1](College of Information Science and Engineering, Huaqiao Univesity, Xiamen 361021, China)

[2](Xiamen Key Laboratory of ASIC System (Huaqiao University), Xiamen 361008, China)

[3](Fujian Engineering Research Center of Motor Control and System Optimal Schedule, Xiamen 361002, China)

Abstract: In this paper, a new cuckoo search algorithm with gravitational acceleration search mechanism is presented to address low convergence rate and deteriorated search precision. The algorithm is fundamentally inspired by the fact that gravitational search can also get the global optimal without perceiving the change on the driving effect of external environment. Each of the cuckoo nests exerted on different quality not only follows the Levy flight law but also abides the law of universal gravitation during the process of optimization, which accelerates the convergence significantly due to the intrinsic gravitational attraction between individuals within the cuckoo nests. Furthermore, a new probability mutation approach is formally given to achieve a balance between the global and local search for the proposed algorithm. Consequently, the global convergence efficiency and search precision of the algorithm are significantly enhanced. Via mathematical analysis and 26 benchmark test functions, the proposed agorithm is competitive for the convergence rate and search precision in a comparison with other variants of intelligent optimization algorithm.

Key words: cuckoo search algorithm; acceleration; gravitation; discovery probability

* 基金项目: 国家自然科学基金(61204122); 福建省中青年教师教育科研项目(JA15037); 福建省自然科学基金(2015J1263)

Foundation item: National Natural Science Foundation of China (61204122); Mid-Aged and Young Teachers Education Research Project of Fujian Province (JA15037); Natural Science Foundation of Fujian Province (2015J1263)

收稿时间: 2018-07-02; 修改时间: 2019-09-26; 采用时间: 2020-02-04

布谷鸟算法(cuckoo search,简称 CS)是 Yang 等人于 2009 年根据布谷鸟的孵育寄生行为提出的一种新型仿生智能优化算法[1,2],该算法具有通用性强、控制参数少、算法执行速度快等优点,其整体性能相比于粒子群算法(particle swarm optimization,简称 PSO)、遗传算法(genetic algorithm,简称 GA)、蚁群算法(ant colony optimization,简称 ACO)、模拟退火算法(simulated annealing,简称 SA)具有较好的竞争力,已经成功应用于工业界各个领域,例如结构优化[3,4]、光伏系统[5,6]、支持向量机及神经网络训练[7,8]和多目标优化[9−12]等.

然而,布谷鸟算法与其他智能启发式算法类似,也存在搜索效率不高的问题.为此,国内外学者对该算法进行了大量的研究以提高收敛性能.目前的改进算法主要包括两个方面.

- 算法控制参数改进和混合算法策略.文献[13]提出一种自适应搜索步长的 CS 算法,提高了收敛性能.文献[14]在文献[13]的基础上,将搜索步长因子变更为 3 种不同的函数适应度值变化因子,通过动态调节步长因子,增强算法的鲁棒性.文献[15]将搜索步长因子设计为均匀分布随机数,虽然提高了 CS 算法的全局搜索性能,但是该方法没有考虑局部搜索的性能,导致算法执行后期易发生迟滞现象.文献[16]提出一种逐维更新的函数评价策略,将各维逐一更新,同时采用贪婪更新方式接受改善解.该算法具有一定的竞争力,但是逐维更新的评价策略导致函数评价次数急剧增加,算法执行效率较低.文献[17]将发现概率设置为动态变化,增强了算法的收敛性能.但是由于改进算法搜索步长的可容许范围过小,造成算法全局收敛性能较差.
- 另一方面,混合算法策略的主要方法是布谷鸟算法与其他算法融合,获得最佳的优化性能.文献[18]将和声优化算法与 CS 算法结合,在算法执行过程增加和声算法的变异操作,并将变异结果反馈到寻优过程,加快算法的收敛速度.文献[19]借鉴粒子群算法良好的局部优化性能,在 CS 算法中引入粒子群组件,提高了 CS 算法的收敛精度.文献[20,21]在偏好随机扰动中引入一种正交学习机制,以提高 CS 算法的整体搜索性能.文献[22]将 CS 算法与万有引力算法结合,引入一种局部搜索机制,使布谷鸟个体的更新由局部最优解和引力加速度共同作用.该算法虽然提高了收敛速度,但是引入的局部搜索极易使算法陷入局部最优.文献[23]将 CS 算法与模式搜索方法结合,提高了 CS 算法的局部开采能力.

由上所述,该类算法虽然提高了搜索性能,例如种群的多样性、收敛精度等,但是增加了优化函数评价次数及算法复杂度.当处理高维度多峰函数、脊峰函数和奇异函数时,算法的自适应调节能力较差,寻优效果不理想.同时,该类算法没有深入挖掘种群寻优过程的内部机理,算法执行效率较低.因此,研究新的改进方法具有积极的意义.

基于此,本文提出一种具有万有引力加速机理的布谷鸟算法(gravitational acceleration search based cuckoo search,简称 GASCS),提高 CS 算法的收敛速度和全局收敛效率.本文的主要特色及贡献为:(1) 该算法基于万有引力搜索无需学习外部环境因素的变化亦能感知全局最优的特点,将布谷鸟巢穴等价为不同质量的个体,利用 CS 算法优化过程中遵循的牛顿万有引力定律进行加速搜索,并且提出一种概率变异方法增大种群多样性;(2) 该算法提出两种新型的布谷鸟个体更新律;(3) 该算法解决了 CS 算法存在的 3 个局限性,提高了 CS 算法的全局搜索效率和收敛精度.

1 布谷鸟算法(CS)及其存在的局限性

布谷鸟采用一种特殊的寄生宿主巢穴的方式孕育繁殖,它将孵育的蛋置入寄生宿主的巢穴,让寄生宿主孵化布谷鸟蛋.由于布谷鸟幼雏能发出比寄生宿主幼雏更闪亮的叫声,因此获得更多的食物,具有更高的存活率.在某些情形下,寄生宿主发现巢穴内不是宿主幼雏,则会遗弃该巢穴,并重新选择新的巢穴孵育繁殖.Yang X. S. 和 Deb 等人基于上述孵育寄生机理,提出布谷鸟算法.它遵循以下 3 个理想规则[1].

规则 1. 每只布谷鸟 1 次有且仅有孵化 1 个蛋,并随机选择 1 个寄生宿主巢穴存放.

规则 2. 在随机选择的 1 个寄生宿主巢穴中,最优的寄生宿主巢穴将被保留至下一代.

规则 3. 可利用的寄生宿主巢穴数量是固定的,1 个寄生宿主巢穴的宿主发现寄生布谷鸟蛋的概率为 P_a.

基于上述 3 个理想规则,可以得到 1 个宿主巢穴代表 1 个候选解.因此,布谷鸟算法的基本算法流程包含 3

部分:首先,在当前候选解的基础上,以 Levy 飞行随机游动生成新的候选解,评价并保留较好的候选解;其次,按照发现概率 P_a 舍弃部分候选解;最后,按照偏好随机游动方式产生新的候选解并替代舍弃的解,保留较好解后进入下一次进化,直至满足算法的收敛条件.

设布谷鸟算法进化至第 r 代时第 m 个候选解为 $\boldsymbol{X}_{r,m}$, $\boldsymbol{X}_{r,m}=(x_{r,m}^{(1)},x_{r,m}^{(2)},...,x_{r,m}^{(j)},...,x_{r,m}^{(D)})$, $j\in[1,D]$.采用 Levy 飞行随机游动产生新的个体(候选解)$\boldsymbol{X}_{r+1,m}$ 更新律的表达式为

$$\boldsymbol{X}_{r+1,m}=\boldsymbol{X}_{r,m}+(\boldsymbol{X}_{r,m}-\boldsymbol{X}_{r,gb})\cdot(\gamma_0\oplus L(\beta)) \tag{1}$$

其中,$\boldsymbol{X}_{r,gb}$ 为当前搜索到的全局最优解,$L(\beta)$表示 Levy 飞行随机游动路径,γ_0 表示初始搜索步长,⊕表示点对点乘法,t 为飞行时间:

$$L(\beta)\sim\vartheta=t^{-1-\beta},0<\beta\leqslant 2 \tag{2}$$

通过数学代换[1],公式(2)等价于:

$$L(\beta)\sim\frac{\vartheta}{|v|^{1/\beta}}\cdot\left(\frac{\Gamma(1+\beta)\sin(\pi\cdot\beta/2)}{\Gamma\left(\frac{1+\beta}{2}\right)\cdot\beta\cdot 2^{(\beta-1)/2}}\right)^{1/\beta} \tag{3}$$

其中,$\Gamma(\cdot)$为伽玛函数,取β=1.5;ϑ,v 为标准高斯分布随机数.

在偏好随机游动方式中,按照发现概率 P_a 舍弃部分候选解后,生成相同数量的新解:

$$\boldsymbol{X}_{r+1,m}=\boldsymbol{X}_{r,m}+\phi\cdot(\boldsymbol{X}_{r,k}-\boldsymbol{X}_{r,s}) \tag{4}$$

其中,ϕ为算法的控制缩放系数,满足$\phi\in U(0,1)$;$\boldsymbol{X}_{r,k}$ 和 $\boldsymbol{X}_{r,s}$ 分别为第 r 代时第 k 个和第 s 个随机候选解.

基本 CS 算法虽然具有一定的收敛能力,但是存在以下局限性.

(1) 采用 Levy 飞行随机游动产生新候选解的搜索步长设置困难:如果搜索步长较大,则寻优速度增大,但由于 Levy 飞行具有无限跳跃特性,使得算法执行过程中极易跳过全局最优解;如果搜索步长较小,虽然能以较大概率获得全局最优解,但是寻优速度会降低.考虑到 Levy 飞行具有无限大方差并且其增量服从二阶分布,因此,设计合理的搜索步长变得尤为困难.

(2) CS 算法采用发现概率 P_a 舍弃偏好随机游走方式产生新的候选解.这种方法虽然加强了算法的全局搜索性能,但随机处理会舍弃较好的种群个体,而保留适应度较差的种群个体,进而降低算法搜索速度和收敛精度.

(3) CS 算法仅通过两个随机个体的位置向量差获得下次进化的方向,未能充分利用当前宿主巢穴的足够信息,因此在搜索过程中具有盲目性,不能对种群个体进行局部精细化搜索,导致算法具有较差的局部搜索能力.纵然算法具有一定的全局搜索能力,但是仍然存在全局搜索和局部搜索不平衡的问题.

2 万有引力加速机理的布谷鸟算法(GASCS)

2.1 万有引力算法

受万有引力定律启发,伊朗学者 Esmat Rashedi 等人于 2009 年提出了万有引力搜索算法(gravitational search algorithm,简称 GSA)[24].GSA 算法的原理是:物质个体之间由于引力作用相互吸引,并且引力沿质量较大的个体方向移动.物质个体的质量决定其万有引力的大小,质量越大的个体所受的引力越大.在外有引力的作用下,物质个体位置不断变化,最终,整个群体都聚集在物质个体质量最大的周围,即逼近优化问题的全局最优解.

在万有引力算法中,借鉴理论物理学中质量定义的 3 种属性描述函数适应度值,即主动引力质量 M_a、被动引力质量 M_p 和惯性质量 M_i.每个物质个体的位置对应优化问题解,其万有引力和惯性质量 M_i 共同决定相应的函数适应度数值.事实上,GSA 算法通过调节 M_a,M_p 和 M_i 来控制算法进化,促使群体中的物质个体聚集在万有引力最大的物质个体附近,进而搜索到最优解.

GSA 算法模型包含物质个体的质量计算、引力计算、加速度计算和速度更新与位置更新.算法首先对解空间内的物质个体位置和速度初始化.如文献[24,25]所示,设 D 维空间维度中进化至第 r 代时第 m 个物质个体的

位置和速度分别为

$$\boldsymbol{X}_{r,m}=(x_{r,m}^{(1)},x_{r,m}^{(2)},...,x_{r,m}^{(j)},...,x_{r,m}^{(D)}),\ \boldsymbol{V}_{m}=(v_{r,m}^{(1)},v_{r,m}^{(2)},...,v_{r,m}^{(j)},...,v_{r,m}^{(D)}),$$

其中,j=1,2,3,…,D, $x_{r,m}^{(j)}$ 和 $v_{r,m}^{(j)}$ 分别表示第 m 个物质个体在第 j 维的位置分量和速度分量.通过物质个体位置对应的适应度数值,确定物质个体的质量 M_i 和受到的万有引力.根据牛顿第二定律计算物质个体的加速度,并更新物质个体位置 $\boldsymbol{X}_{r,m}$ 和速度 $\boldsymbol{V}_{r,m}$.GSA 算法主要包括以下 4 个步骤.

(1) 物质个体质量计算

根据文献[24–26],物质个体 m 的质量定义为

$$q_{r,m}=\frac{f_{r,m}-f_{r,worst}}{f_{r,best}-f_{r,worst}} \tag{5}$$

$$M_{r,m}=\frac{q_{r,m}}{\sum_{m=1}^{N}q_{r,m}} \tag{6}$$

其中,$f_{r,m}$ 和 $M_{r,m}$ 分别表示 GSA 算法第 r 次进化时,物质个体 m 的函数适应度值和相应的质量.假设所求解的为最小化问题,$f_{r,best}$ 和 $f_{r,worst}$ 分别表示第 r 次进化时物质个体最优的适应度数值和最差适应度数值,其数学表达式如下:

$$f_{r,best}=\min_{m\in\{1,2,...,N\}}f_{r,m} \tag{7}$$

$$f_{r,worst}=\max_{m\in\{1,2,...,N\}}f_{r,m} \tag{8}$$

(2) 物质个体万有引力计算

根据万有引力定律,在维度 j 上,物质个体 m 对物质个体 k 的万有引力定义如下:

$$F_{r,mk}^{(j)}=G_r\cdot\frac{M_{r,pm}\cdot M_{r,ak}}{R_{r,mk}+\varepsilon}\cdot(x_{r,m}^{(j)}-x_{r,k}^{(j)}) \tag{9}$$

其中,ε为一个无穷小的常数;$M_{r,pm}$ 表示被作用于物质个体 m 第 r 次进化时的惯性质量,$M_{r,ak}$ 表示为作用于物质个体 m 第 r 次进化时的惯性质量,并且 $M_{r,pm}$=$M_{r,ak}$=$M_{r,m}$;$R_{r,mk}$ 为物质个体 m 和物质个体 k 的欧氏空间距离,即 $R_{r,mk}$=$\|\boldsymbol{X}_{r,m}$–$\boldsymbol{X}_{r,k}\|_2$;G_r 表示第 r 次进化时物质个体的万有引力常数,其表达式如公式(10)所示:

$$G_r=G(G_0,r)=G_0\cdot e^{-\theta r/W} \tag{10}$$

其中,G_0 为进化初始时物质个体的万有引力系数;θ为算法控制参数,一般取θ=20;W 为算法进化代数的最大值.

在维度 j 上,物质个体 m 所受的万有引力合力为 $F_{r,m}^{(j)}$:

$$F_{r,m}^{(j)}=\sum_{k\in b,k\neq m}^{N}d_j\cdot F_{r,mk}^{(j)} \tag{11}$$

其中,d_j 为区间(0,1)内均匀分布的一个随机数,b 为物质个体数目.

(3) 物质个体加速度计算

由牛顿第二定律,物质个体 m 在维度 j 上第 r 次进化时的加速度定义如下:

$$a_{r,m}^{(j)}=\frac{F_{r,m}^{(j)}}{M_{r,mm}} \tag{12}$$

其中,$M_{r,mm}$=$M_{r,pm}$=$M_{r,ak}$=$M_{r,m}$.

(4) 物质个体更新速度和位置

算法在每次进化过程中,物质个体按照公式(13)和公式(14)分别更新速度 $v_{r,m}^{(j)}$ 和位置 $x_{r,m}^{(j)}$,其具体的数学表达式分别为

$$v_{r+1,m}^{(j)}=d_j\cdot v_{r,m}^{(j)}+a_{r,m}^{(j)} \tag{13}$$

$$x_{r+1,m}^{(j)}=x_{r,m}^{(j)}+v_{r+1,m}^{(j)} \tag{14}$$

2.2 GASCS数学机理分析

传统的 CS 算法是基于 Levy 飞行随机游动方式和偏好随机游动方式两种搜索机理,虽然具有一定的收敛性能,但仍然存在第 1 节所阐述的 3 点局限性.为此,本文提出具有万有引力加速机理的布谷鸟算法.它基于万有引力搜索无需学习外部环境因素的变化亦能感知全局最优信息的特点,将布谷鸟巢穴赋予不同的个体质量,其在优化过程中不仅遵循 Levy 飞行规律,而且遵循万有引力定律.

GASCS 算法的详细机理如图 1 所示,设定 $nest(m)$,$nest(s)$,$nest(k)$分别为种群进化的宿主巢穴位置,$nest\_g$ 为最优巢穴位置,并且假设宿主巢穴的质量满足 $nest(s)>nest(m)>nest(k)$.因此,最优巢穴位置 $nest\_g$ 的质量满足如下数学关系:$nest\_g>nest(s)>nest(m)>nest(k)$.为便于分析,设定 $\boldsymbol{X}_{r,m}$ 等价于 $nest(m)$,$\boldsymbol{X}_{r,s}$ 等价于 $nest(s)$,$\boldsymbol{X}_{r,k}$ 等价于 $nest(k)$以及 $\boldsymbol{X}_{r,gb}$ 等价于 $nest\_g$.$\boldsymbol{F}_{r,ms}$,$\boldsymbol{F}_{r,mk}$ 和 $\boldsymbol{F}_{r,mg}$ 是分别作用于 $nest(m)$到 $nest(s)$、$nest(m)$到 $nest(k)$以及 $nest(m)$到 $nest\_g$ 的万有引力.在万有引力作用下产生的加速度分别为 $\boldsymbol{a}_{ms}$,$\boldsymbol{a}_{mk}$ 和 $\boldsymbol{a}_{mg}$.$\boldsymbol{F}_{r,m}$ 是作用于 $nest(m)$和 $nest\_g$ 产生加速度 $\boldsymbol{a}_{mg}$ 的所有合力;同时,$\boldsymbol{F}_{r,t}$ 是作用于 $\boldsymbol{F}_{r,m}$ 和 $nest(k)$产生加速度 $\boldsymbol{a}_r$ 的合外力.因此,在万有引力的作用下,宿主巢穴逼近质量最大的巢穴 $nest\_g$.根据牛顿第二定律,宿主巢穴在进化过程中所受的加速度由巢穴所受的合外力及自身质量共同决定.基于此,将布谷鸟算法的个体更新律(1)和更新律(4)改进为公式(15)和公式(16):

$$\boldsymbol{X}_{r+1,m}=\boldsymbol{a}_r-(\gamma_0\oplus L(\beta))\cdot(\boldsymbol{a}_r-\boldsymbol{X}_{r,gb}) \tag{15}$$

$$\boldsymbol{X}_{r+1,m}=\boldsymbol{X}_{r,gb}+p\cdot(\boldsymbol{X}_{r,k}-\boldsymbol{X}_{r,s}-\boldsymbol{a}_r) \tag{16}$$

其中,公式(15)和公式(16)分别表示 Levy 飞行随机游动和偏好随机游动方式的个体位置,$p\in(0,+\infty)$,$\boldsymbol{a}_r$ 为 $\boldsymbol{F}_{r,t}$ 产生的合加速度.由公式(15)可知:与传统的 CS 算法宿主巢穴只是单纯的按照 Levy 飞行随机游动产生候选解不同,GASCS 算法将每个宿主巢穴视为不同质量的物质个体,它遵循万有引力定律和牛顿第二定律.如图 1 所示,宿主巢穴 $nest(m)$,$nest(s)$,$nest(k)$沿加速度 $\boldsymbol{a}_r$ 的方向移动,并且公式(15)的步长修正项为 $-(\gamma_0\oplus L(\beta))\cdot(\boldsymbol{a}_r-\boldsymbol{X}_{r,gb})$.由向量运算法则得到 $\boldsymbol{a}_g=\boldsymbol{a}_r-(\gamma_0\oplus L(\beta))\cdot(\boldsymbol{a}_r-\boldsymbol{X}_{r,gb})$,因此,算法能较好地逼近全局最优解 $nest\_g$.

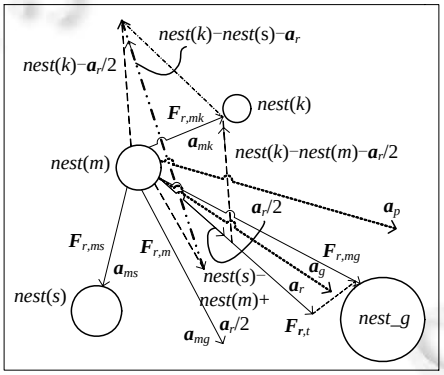


Fig.1 Mechinism of gravitational attraction acceleration

图 1 万有引力加速机理

综上所述,由于 Levy 飞行随机游动和偏好随机游动方式的个体位置更新引入了合加速度 $\boldsymbol{a}_r$,并且 $\boldsymbol{a}_r$ 随算法进化而自适应变化的,因此 GASCS 算法能够平衡算法的搜索进程,抵消搜索步长设置不合理引发的算法性能恶化.本算法在进化中不仅利用 Levy 飞行随机游动方式的无限跳跃特性易于跳出局部最优解的特点,而且利用了万有引力加速度快速逼近全局最优解的特性,克服了第 1 节 CS 算法的第 1 点局限性.

其次,针对第 2 点局限性中随机处理会舍弃较好的种群个体引起算法寻优速度和收敛精度下降的问题,本文算法的改进思路如图 1 所示.作用于 $nest(m)$到 $nest(k)$的外力与 $\boldsymbol{F}_{r,t}/2$ 产生的合加速度 $\boldsymbol{a}_r/2$ 的向量差为 $nest(k)-nest(m)-\boldsymbol{a}_r/2$.同理,作用于 $nest(m)$到 $nest(s)$的合力与 $\boldsymbol{F}_{r,t}/2$ 产生的合加速度 $\boldsymbol{a}_r/2$ 的向量和为 $nest(s)-nest(m)+\boldsymbol{a}_r/2$.根据向量运算法则,$\boldsymbol{X}_{rd}=nest(k)-nest(m)-\boldsymbol{a}_r/2-(nest(s)-nest(m)+\boldsymbol{a}_r/2)=nest(k)-nest(s)-\boldsymbol{a}_r\Leftrightarrow\boldsymbol{X}_{r,k}-\boldsymbol{X}_{r,s}-\boldsymbol{a}_r$.

由公式(16)可知,$\boldsymbol{X}_{r,gb}$ 与 $p\cdot\boldsymbol{X}_{rd}$ 组合的结果为图 1 所示的加速度 $\boldsymbol{a}_p$.通过 Levy 飞行随机游动方式产生的加速

度 $\boldsymbol{a}_g$ 以及偏好随机游动方式产生的加速度 $\boldsymbol{a}_p$ 均能快速聚集在全局最优解 *nest_g*.同时,由于向量 *nest*(*k*)–*nest*(*s*)–$\boldsymbol{a}_r$ 偏离全局最优解方向,因此在偏好随机游动方式中,能有效预防种群进化后期出现的迟滞现象.注意到公式(16)中 $p\in(0,+\infty)$,且为确定的数,不同于传统 CS 算法中的控制缩放系数 q 为随机数.这种非随机处理方式能够保留较好的种群个体,舍弃进化过程中出现的较差种群个体,提高算法寻优速度和收敛精度.

最后,针对第 3 点局限性中存在全局搜索和局部搜索不平衡的问题,GASCS 算法提出一种概率变异方法增大种群进化的多样性,避免算法陷入局部搜索以及算法执行后期出现的迟滞现象.

假设 $\boldsymbol{X}_{r,m}=(x_{r,m}^{(1)},x_{r,m}^{(2)},...,x_{r,m}^{(j)},...,x_{r,m}^{(D)})$,并且 $x_{r,m}^{(j)}\in[l^{(j)},u^{(j)}]$.执行概率变异操作时,首先从 $\boldsymbol{X}_{r,m}$ 中以概率 1/*D* 选取种群个体 $x_{r,m}^{(j)}$,$j\in[1,D]$;其次,利用公式(17)的 $\tilde{x}_{r,m}^{(j)}$ 取代 $x_{r,m}^{(j)}$,其中,k 为[1,*D*]内的随机整数,ξ为(0,1)内的均匀分布随机数,$l^{(j)}$和 $u^{(j)}$分别为 $x_{r,m}^{(j)}$ 的上界和下界;最后,通过概率变异将候选解 $\boldsymbol{X}_{r,m}$ 更替为 $\tilde{\boldsymbol{X}}_{r,m}$,并进入算法执行过程,其中,$\tilde{\boldsymbol{X}}_{r,m}=(\tilde{x}_{r,m}^{(1)},\tilde{x}_{r,m}^{(2)},...,\tilde{x}_{r,m}^{(j)},...,\tilde{x}_{r,m}^{(D)})$.经过概率变异后的种群进化获得更大的多样性,避免算法寻优过程单一化而造成的全局搜索效率较低的问题.通过两个随机个体的位置向量差、当前搜索到的最优解及群体中宿主巢穴所受到的万有引力加速度,获得下一次进化的方向,充分利用宿主巢穴及群体内部进化过程出现的有效信息:

$$\tilde{x}_{r,m}^{(j)}=\begin{cases}(u^{(j)}+l^{(j)})/2+(1-2\cdot\xi)\cdot(u^{(j)}-l^{(j)})/2, & j=k\\ x_{r,m}^{(j)}, & j\neq k\end{cases} \tag{17}$$

GASCS 算法在进化过程中不仅利用了 Levy 飞行随机游动方式的无限跳跃特性易于跳出局部最优解的特点,而且采用概率变异增大算法的多样性,确保算法能够跳离局部最优解.因此,GASCS 算法较好地解决了 CS 算法存在全局搜索和局部搜索不平衡的问题.

2.3 GASCS算法实现

基于 CS 算法存在的 3 点局限性,本文提出了具有万有引力加速机理的 GASCS 算法.该算法通过当前搜索到的最优解、Levy 飞行随机游动和偏好随机游动方式产生的个体更新位置及宿主巢穴所受到的万有引力加速度共同作用,获得下一次进化的方向及最佳巢穴位置,并采用概率变异方法,引导宿主巢穴沿全局最优解方向移动,进而搜索到全局最优解.

GASCS 算法的执行步骤如下所示.

步骤 1. 算法初始化,种群为 N 个宿主巢穴,优化问题的空间维度 D,最大进化代数 W,发现概率 P_a,初始时刻万有引力系数 G_0,算法控制参数 θ,宿主巢穴的初始移动速度 $V_{r,m}$,其中,进化代数 r=1.

步骤 2. 计算随机化宿主巢穴位置 $X_{r,m}$(m=1,2,…,N)对应的适应度函数值 $f(X_{r,m})$.

步骤 3. 由公式(10)计算 G_r,同时,由公式(7)和公式(8)计算当前最优值 $f_{r,best}$ 和最差值 $f_{r+1,worst}$,以及对应的最优解 $X_{r,gb}$.

步骤 4. 根据公式(5)和公式(6)计算宿主巢穴的质量 $q_{r,m}$,$M_{r,m}$.

步骤 5. 分别根据公式(11)和公式(12)计算当前进化代数的宿主巢穴所受到的万有引力合力 $F_{r,m}^{(j)}$ 和加速度 $a_{r,m}^{(j)}$.

步骤 6. 采用公式(15)的 Levy 飞行随机游动方式产生新的宿主巢穴,按照发现概率 P_a 舍弃候选解 $X_{r+1,m}$.

步骤 7. 采用公式(16)的偏好随机游动方式产生新的宿主巢穴,替换步骤 6 中被舍弃的候选解.

步骤 8. 采用概率变异方法产生候选解 $\tilde{\boldsymbol{X}}_{r+1,m}$ 对应的函数适应度值 $f(\tilde{\boldsymbol{X}}_{r+1,m})$.

步骤 9. 计算步骤 8 中种群产生的候选解对应的函数适应度值 $f(\tilde{\boldsymbol{X}}_{r+1,m})$,同时更新当前最优值 $f_{r+1,best}$ 和最差值 $f_{r+1,worst}$ 以及对应的最优解 $X_{r+1,gb}$.

步骤 10. 若满足算法终止条件,则输出当前进化的最优值及最优解,并停止算法;否则,转向步骤 3 继续执行算法.

3 计算机实验仿真及分析

3.1 测试函数及实验标准

为了全面验证提出的 GASCS 算法的有效性和先进性,本文采用 3 类测试函数进行实验:第 1 类为高维度单模函数,选取文献[24]的 F_1~F_7 共 7 个测试函数;第 2 类为高维度多模函数,选取文献[24]的 F_8~F_{13} 和文献[25]的 f_{18}~f_{20} 共 9 个测试函数;第 3 类为固定维度的复杂模态函数,选取文献[24]的 F_{14}~F_{23} 共 10 个测试函数.为保证函数标号的唯一性,将文献[25]的 f_{18}~f_{20} 这 3 个测试函数命名为 F_{14}~F_{16},而原固定维度的测试函数 F_{14}~F_{23} 改为 F_{17}~F_{26}.固定维度的复杂模态函数具有全局最优解非对称和局部最优解分布不均匀的特点,因此其优化难度极大,是优化算法性能较好的测试算例.

为了检验 GASCS 算法的性能,本节对传统的 CS 算法、改进的 CS 算法以及粒子群算法(particle swarm optimization,简称 PSO)、人工蜂群算法(artificial bee colony,简称 ABC)、万有引力搜索算法(gravitational search algorithm,简称 GSA)等智能算法进行对比.

本节实验的参数设置为种群为 N=20 个宿主巢穴,函数空间维度 D=30,发现概率 Pa=0.2,控制参数 θ=20.为保证算法测试的鲁棒性,每种算法均独立运行 30 次,计算其最差值、最优值、优化平均值和优化标准方差.定义函数适应度误差为 $Fn(u)$,即

$$Fn(u)=f(\boldsymbol{X}_{gb}(u))-f(\boldsymbol{X}_{gb}^{*}) \tag{18}$$

其中,u 为测试函数运行的次数,u=1,2,…,30;$\boldsymbol{X}_{gb}(u)$为第 u 次搜索到的实际最优解; $\boldsymbol{X}_{gb}^{*}$ 为理论的最优解;$f(\boldsymbol{X}_{gb}(u))$ 和 $f(\boldsymbol{X}_{gb}^{*})$ 分别为搜索到的实际最优值和理论最优值.假设 $A=\{|Fn(u)|<\varepsilon\}$,其中,$\varepsilon$为函数适应度绝对误差精度.定义 $\delta_A(u)=\begin{cases}1, & u\in A\\ 0, & u\notin A\end{cases}$,因此,算法进化过程中寻优成功率 Ps 定义为

$$Ps=\frac{1}{30}\sum_{u=1}^{30}\delta_A(u) \tag{19}$$

3.2 测试函数及实验标准

表 1 为 GASCS 和 CS 算法[2]在不同维度空间上独立运行 30 次的优化函数适应度最优值、最差值及平均值等寻优性能比较.表 1 的 F 为优化函数,D 为维度空间,Algorithm 为比较的优化算法,Opt 为优化函数的理论最优值,$Best$ 为算法运行 30 次搜索到的最佳优化值,$Aver$ 为算法独立运行 30 次的平均优化值,$Worst$ 为算法独立运行 30 次的最差优化值,Ps 为公式(19)中定义的寻优成功率,Var 为算法独立运行 30 次的优化值标准方差.测试函数由于函数模态不同,其全局最优值也尽不相同,因此函数适应度的绝对误差精度也不同.故设置 $F1$~$F4$ 的 ε为 1.0E−3,$F5$~$F7$,$F9$~$F26$ 的 ε为 1.0E−2,$F8$ 的 ε为 1.0E+2,并且设置算法运行的最大迭代次数为 400 次.

表 1 显示出,在高维度单模函数求解中,GASCS 算法与 CS 算法比较具有更好的搜索精度.对于 $F1$~$F7$,GASCS 算法只有 $F5$ 和 $F6$ 的寻优成功率没有达到 100%,其余 5 个函数均搜索到全局最优解;与之相对,CS 算法的寻优成功率则为 0.GASCS 算法对 $F5$ 的优化结果远胜于 CS 算法.GASCS 算法对于高维度多模函数 $F9$~$F11$,$F14$~$F16$ 均取得全局最优解;而 $F8$,$F12$,$F13$ 虽然获得较好的结果,但未能获得全局最优解.与此形成对比的是,CS 算法在优化高维度多模函数的求解效果较差,均没有搜索到全局最优值.固定维度的复杂模态函数具有全局最优解非对称、局部最优解分布不均匀的特点,因此,测试函数在优化过程中具有欺骗性,增大了优化算法的设计难度.在固定维度的复杂模态函数求解中,GASCS 算法和 CS 算法均获得较好的优化结果,其中,GASCS 算法寻优性能更好,它均能搜索到全局最优解.不论对于高维度单模函数、高维度多模函数,还是固定维度复杂多模函数,GASCS 算法相比于 CS 算法的求解效率和精度有较大的提高,具有优越的全局优化性能.

Table 1 Comparison of GASCS with CS on convergence performance

表 1 GASCS 与 CS 算法收敛性能比较

F	*D*	Algorithm	*Opt*	*Best*	*Aver*	*Worst*	*Ps*	T_m/s
*F*1	30	CS	0	9.879 2	43.619 3	76.434 7	0	0.541 8
		GASCS		2.2219E−114	**1.8789E−104**	5.4581E−103	100	1.003 4
*F*2	30	CS	0	4.250 1	9.999 8	22.954 8	0	0.539 4
		GASCS		1.2333E−057	**1.1517E−053**	2.9067E−052	100	1.006 3
*F*3	30	CS	0	839.603 8	1.6008E+3	3.1700E+3	0	1.025 9
		GASCS		1.0770E−112	**1.9675E−104**	1.8451E−103	100	1.458 6
*F*4	30	CS	0	9.077 0	15.685 6	28.067 7	0	0.518 4
		GASCS		6.2259E−56	**2.9695E−52**	8.2148E−51	100	0.993
*F*5	30	CS	0	1.0788E+3	4.5282E+3	1.4191E+4	0	0.581 9
		GASCS		28.489 5	**28.683 7**	28.728 5	0	1.068 5
*F*6	30	CS	0	12.903 6	54.165 2	143.027 4	0	0.526 6
		GASCS		2.4294E−002	**1.5086E−001**	4.3134E−001	93.3	0.999 7
*F*7	30	CS	0	0.065 6	0.171 3	0.349 9	0	0.537 6
		GASCS		5.0381E−7	**9.9662E−5**	2.5016E−4	100	1.014 6
*F*8	30	CS	0	−82324E+3	−7.6822E+3	−7.2484E+3	0	0.556 2
		GASCS		−1.1149E+004	**−8.5133E+003**	−6.1503E+003	0	1.027 2
*F*9	30	CS	0	0.341 3	0.871 1	2.131 3	0	0.552 2
		GASCS		0	**0**	0	100	1.011
*F*10	30	CS	0	4.048 4	6.994 1	11.920 0	0	0.610 9
		GASCS		8.8818E−16	**8.8818E−16**	8.8818E−16	100	1.051 7
*F*11	30	CS	0	9.9809E+3	5.8827E+3	8.1894E+3	0	0.669 2
		GASCS		0	**0**	0	100	1.112
*F*12	30	CS	0	2.901 8	7.026 6	15.221	0	0.860 8
		GASCS		9.8604E−004	**8.8134E−003**	2.7736E−002	93.3	1.305 8
*F*13	30	CS	0	13.411 9	29.486 6	79.937 1	0	0.863 4
		GASCS		4.0716E−002	**2.3018E−001**	6.9861E−001	90	1.297 5
*F*14	30	CS	0	8.379 6	10.874 0	15.304 2	0	0.592 5
		GASCS		6.2538E−058	**1.4166E−054**	2.9365E−053	100	1.056 3
*F*15	30	CS	0	3.353 2	11.098 2	37.014 3	0	0.602 9
		GASCS		9.5460E−002	**2.590 5**	7.160 0	43.33	1.068 1
*F*16	30	CS	−464.999 5	−316.895 7	−291.490 4	−273.686 8	0	4.419 8
		GASCS		−464.9995	**−464.999 5**	−464.999 5	100	4.644 8
*F*17	2	CS	1	0.998 0	**0.9980**	0.998 0	100	1.102 1
		GASCS		0.998 0	**0.998 0**	0.998 0	100	1.340 9
*F*18	4	CS	3.0749E−4	3.0750E−4	**3.3400E−4**	5.3027E−4	100	0.541 3
		GASCS		3.0749E−4	4.2650E−4	9.0051E−4	100	0.839 6
*F*19	2	CS	−1.031 6	−1.031 6	**−1.031 6**	−1.031 6	100	0.472 9
		GASCS		−1.031 6	**−1.031 6**	−1.031 6	100	0.756 5
*F*20	2	CS	0.397 9	0.397 9	**0.381 2**	0.350 4	86.67	0.479 3
		GASCS		0.397 9	0.397 9	0.397 9	100	0.746 7
*F*21	2	CS	3.000 0	3.000 0	**3.000 0**	3.000 0	100	0.460 8
		GASCS		3.000 0	**3.000 0**	3.000 0	100	0.750 4
*F*22	3	CS	−3.052 4	−3.052 4	**−3.052 4**	−3.052 4	100	0.547 6
		GASCS		−3.052 4	**−3.052 4**	−3.052 4	100	0.830 1
*F*23	6	CS	−3.322 0	−3.322 0	**−3.322 0**	−3.321 6	100	0.557
		GASCS		−3.322 0	−3.314 1	−3.203 1	93.33	0.868 2
*F*24	4	CS	−10.153 2	−10.153 2	−10.152 9	−10.145 7	100	0.593 2
		GASCS		−10.153 2	**−10.153 2**	−10.153 2	100	0.886 6
*F*25	4	CS	−10.402 9	−10.402 9	−8.523 6	−5.110 3	86.67	0.629 5
		GASCS		−10.402 9	**−10.402 9**	−10.402 9	100	0.920 9
*F*26	4	CS	−10.536 4	−10.536 4	−10.535 7	−10.517 8	96.67	0.689 8
		GASCS		−10.536 4	**−10.536 4**	−10.536 4	100	0.979 2

3.3 与其他改进CS算法及其他智能算法性能比较

为分析GASCS算法与其他改进CS算法及其他智能算法性能,表2和表3分别示出了GASCS算法与HSCS算法[18]、CSPSO算法[27]、PSO算法[28]、OLCS算法[29]、PSCS算法[23]、CS-GSA算法[22]、ACS算法[30]、HeCOS算法[31]、NNCS算法[32]、ABC算法[33,34]以及GSA算法[24]的性能比较结果.由于篇幅有限,本节选取高维度单

模函数、高维度多模函数和固定维度复杂多模函数中的部分函数作为算法比较对象.

表 2~表 4 分别为 D=10,D=30 和 D=50 时,GASCS 算法与其他改进 CS 算法对比结果.从表中看出:与其他改进的 CS 算法相比,GASCS 算法显示了优越的全局优化性能,其求解效率和精度有较大的提高.特别是函数 $F9$ 和 $F11$,当优化函数维度分别为 D=10,D=30 和 D=50 时 GASCS 算法均能收敛到理论全局最优值.而对于函数 $F5$,GASCS 算法搜索的优化均值远优于 CSPSO 算法、OLCS 算法、PSCS 算法和 CS-GSA 算法的优化均值,也优于 ACS 算法、HeCOS 算法和 NNCS 算法.对于函数 $F17$,虽然 CSPSO 算法和 PSCS 算法均能搜索到理论全局最优值,但是 GASCS 算法的寻优速度更快.

Table 2 D=10, comparison of GASCS with state-of-the-art CS algorithms

表 2 D=10,GASCS 算法与其他改进 CS 算法对比

F	Criterian	ACS	HeCOS	NNCS	CSPSO	OLCS	PSCS	CS-GSA	GASCS
$F1$	*Aver*	2.78E−93	4.03E−95	2.31E−98	6.36E−8	5.47E−10	3.23E−20	2.15E−11	**2.02E−117**
	Var	3.15E−92	1.30E−96	2.06E−98	1.57E−8	6.02E−10	2.38E−19	7.75E−11	1.61E−117
$F2$	*Aver*	8.27E−58	8.28E−59	2.11E−48	4.87E−32	1.39E−14	**3.72E−59**	1.16E−42	1.87E−58
	Var	3.12E−58	1.73E−58	8.76E−49	9.97E−33	4.29E−13	3.22E−59	2.48E−43	6.92E−57
$F3$	*Aver*	2.68E−60	3.48E−95	4.55E−70	77.45E+2	1.03E+3	3.89E+2	9.88E+2	**3.62E−117**
	Var	8.39E−59	2.89E−94	2.08E−68	1.21E+2	1.38E+2	7.03E+2	1.02E+2	7.86E−116
$F4$	*Aver*	4.39E−43	3.34E−35	1.28E−55	1.37E−5	6.77E−30	3.22E−48	2.88E−30	**3.82E−60**
	Var	5.38E−42	6.18E−35	7.03E−54	6.67E−5	1.98E−30	5.49E−48	8.66E−29	6.30E−59
$F5$	*Aver*	88.35	75.16	27.38	3.26E+3	1.08E+3	1.16E+2	6.38E+2	**8.1731**
	Var	3.16	7.83	3.19	1.65E+3	9.77E+2	1.02E+2	1.04E+2	2.04E−1
$F6$	*Aver*	2.73E−2	7.13E−15	0.27	103.56	3.11	1.02	3.67E−2	**1.44E−17**
	Var	1.89E−2	3.68E−15	0.15	9.77	0.39	0.33	2.11E−2	5.53E−17
$F7$	*Aver*	4.86E−3	8.27E−5	2.33E−2	7.38E−2	7.39E+2	7.17E−2	8.43E−1	**3.11E−6**
	Var	7.23E−4	3.26E−5	5.26E−2	2.21E−2	5.27E+2	8.11E−2	1.05E−1	1.29E−5
$F8$	*Aver*	−4.17E+3	**−4.18E+3**	−4.10E+3	−3.58E+3	−4.10E+3	−3.84E+3	−4.11E+3	−4.07E+3
	Var	3.76E+2	6.19E+2	1.07E+3	2.18E+2	1.09E+2	2.18E+2	3.81E+2	3.21E+2
$F9$	*Aver*	**0**	1.11E−95	**0**	1.48E−36	3.17E−20	5.35E−45	4.23E−3	**0**
	Var	0	1.28E−96	0	6.07E−36	4.14E−20	4.95E−46	2.24E−3	0
$F10$	*Aver*	**8.88E−16**	**8.88E−16**	**8.88E−16**	1.25E−3	8.95E−6	**8.88E−16**	8.65E−10	**8.88E−16**
	Var	0	0	0	4.29E−2	1.02E−6	0	3.44E−10	0
$F11$	*Aver*	2.48E−20	3.34E−32	**0**	5.54E−15	4.46E−8	**0**	5.37E+3	**0**
	Var	2.15E−20	8.27E−33	0	2.49E−15	1.33E−7	0	1.16E+3	0
$F12$	*Aver*	2.77E−5	3.98E−6	3.66E−3	1.83E−4	3.82E−3	5.28E−4	0.82	**5.92E−19**
	Var	2.16E−5	5.38E−6	1.27E−4	4.83E−5	1.13E−3	7.71E−5	0.14	3.57E−19
$F13$	*Aver*	15.76	9.16	1.06E−11	11.75	47.88	9.11	33.75	**1.85E−18**
	Var	1.09	1.23	1.03E−11	2.11	1.03	1.28	3.41	4.72E−18
$F14$	*Aver*	3.28E−62	**7.47E−63**	4.63E−44	3.28E−15	3.36E−8	2.17E−20	5.53E−6	1.15E−60
	Var	4.22E−63	2.97E−63	2.12E−44	6.64E−15	1.91E−7	9.74E−20	8.27E−7	1.81E−60
$F15$	*Aver*	10.12	10.56	3.54E−8	3.25E−6	2.77E−7	19.32	6.26	**5.55E−14**
	Var	1.65	3.48	4.72E−8	1.77E−5	1.33E−7	3.84	2.83	2.66E−14
$F16$	*Aver*	−46.74	−32.72	−47.38	−47.18	−49.03	−50.21	−39.03	**−5.50E+1**
	Var	3.28	2.19	3.75	3.11	6.15	2.89	1.37	1.80E−1

Table 3 D=30, comparison of GASCS with state-of-the-art CS algorithms

表 3 D=30,GASCS 算法与其他改进 CS 算法对比

F	Criterian	ACS	HeCOS	NNCS	CSPSO	OLCS	PSCS	CS-GSA	GASCS
$F1$	*Aver*	2.41E−82	3.67E−85	6.14E−88	1.34E−3	3.35E−3	1.02	5.12	**3.15E−104**
	Var	1.76E−82	7.32E−86	4.10E−89	2.12E−3	4.11E−3	0.32	1.24	1.21E−104
$F2$	*Aver*	**3.33E−60**	4.86E−46	4.03E−49	1.67E−22	8.71E−11	6.64E−49	6.23E−40	1.1517E−53
	Var	8.27E−60	9.52E−47	1.28E−49	3.82E−23	2.33E−12	8.21E−49	1.89E−39	1.6623E−53
$F3$	*Aver*	3.77E−53	8.12E−100	2.67E−66	1.54E+3	2.76E+4	3.32E+3	6.12E+2	**4.28E−103**
	Var	6.44E−54	4.95E−100	3.36E−67	6.11E+3	3.66E+3	2.43E+3	2.83E+2	**3.12E−103**
$F4$	*Aver*	8.65E−41	6.92E−37	2.81E−50	8.93E−3	3.76E−15	**6.74E−52**	3.82E−20	8.2148E−51
	Var	2.78E−42	2.93E−37	1.84E−50	7.29E−3	2.48E−14	3.27E−51	1.74E−19	3.2287E−50
$F5$	*Aver*	1.09E+2	78.92	28.96	4.56E+3	1.23E+3	8.54E+2	2.36E+3	**28.70**
	Var	5.74	8.05	3.38	6.11E+3	4.52E+2	3.78E+2	1.21E+3	2.39

Table 3 *D*=30, comparison of GASCS with state-of-the-art CS algorithms (Continued)

表 3 *D*=30,GASCS 算法与其他改进 CS 算法对比(续)

F	Criterian	ACS	HeCOS	NNCS	CSPSO	OLCS	PSCS	CS-GSA	GASCS
*F*6	*Aver*	4.62E−2	**3.22E−3**	0.56	115.67	2.68	1.04	124.87	4.3134E−1
	Var	3.22E−2	9.34E−4	0.03	2.33	0.98	0.02	10.09	0.97
*F*7	*Aver*	3.76E−1	6.32E−3	7.82E−2	9.32E+2	9.89E+2	4.32E+1	1.02E−1	**9.96E−5**
	Var	0.11	3.25E−3	1.05E−2	5.87E+2	6.11E+2	3.21E+1	2.81E−1	3.21E−4
*F*8	*Aver*	−1.12E+4	**−1.20E+4**	−1.04E+4	−3.24E+3	−6.78E+3	−7.36E+3	−5.04E+3	−8.5133E+3
	Var	112	231	1.23E+3	363	448	109	518	100.33
*F*9	*Aver*	**0**	2.98E−75	3.76E−98	4.38E−6	7.55E−5	3.11E−6	6.11E−2	**0**
	Var	0	5.34E−76	2.83E−98	3.54E−6	3.02E−5	5.14E−6	3.36E−2	0
*F*10	*Aver*	**8.88E−16**	**8.88E−16**	**8.88E−16**	3.27E−3	6.77E−6	**8.88E−16**	8.27E−10	**8.8818E−16**
	Var	0	0	0	2.97E−1	9.31E−5	0	6.11E−10	0
*F*11	*Aver*	7.12E−20	6.53E−30	**0**	1.98E−9	3.25E−7	**0**	6.37E+3	**0**
	Var	9.03E−19	4.38E−28	0	8.22E−8	2.32E−6	0	2.81E+3	0
*F*12	*Aver*	4.43E−3	5.86E−2	**1.22E−3**	0.36	1.29	26.77	0.88	2.7736E−2
	Var	7.45E−4	3.44E−2	3.21E−4	0.16	0.54	3.86	0.39	0.008
*F*13	*Aver*	14.32	11.88	0.53	20.34	27.87	13.23	12.45	**0.3011**
	Var	1.09	1.23	1.03	2.11	1.03	1.28	3.41	2.12E−2
*F*14	*Aver*	4.25E−50	3.46E−53	1.86E−39	2.01E−5	6.64E−6	8.45E−7	3.54E−1	**2.9365E−53**
	Var	8.81E−49	4.78E−52	5.52E−40	3.31E−5	2.83E−6	1.09E−6	2.27E−2	1.1233E−53
*F*15	*Aver*	8.97	13.11	**1.64E−6**	18.87	20.08	19.32	9.88	2.35E−5
	Var	1.34	2.01	3.44E−5	3.21	2.76	1.98	3.02	0
*F*16	*Aver*	−420.87	−338.21	−411.38	−433.27	−462.33	−114.85	−330.67	**−464.9995**
	Var	4.32	22.13	5.87	3.21	8.67	16.02	13.97	0

Table 4 *D*=50, comparison of GASCS with state-of-the-art CS algorithms

表 4 *D*=50,GASCS 算法与其他改进 CS 算法对比

F	Criterian	ACS	HeCOS	NNCS	CSPSO	OLCS	PSCS	CS-GSA	GASCS
*F*1	*Aver*	8.34E−80	1.12E−85	2.46E−87	4.65E−3	8.44E−2	4.56	11.56	**5.56E−104**
	Var	4.21E−80	5.92E−84	2.88E−87	9.27E−4	1.20E−2	0.87	2.39	2.11E−103
*F*2	*Aver*	**6.38E−55**	7.13E−43	5.11E−47	5.63E−20	9.87E−11	1.53E−48	5.84E−39	2.69E−52
	Var	2.43E−56	1.32E−44	4.58E−47	4.09E−21	3.76E−12	6.36E−48	4.29E−38	1.11E−51
*F*3	*Aver*	6.87E−50	2.09E−99	3.68E−60	3.84E+3	9.12E+5	8.02E+3	3.18E+3	**1.15E−101**
	Var	3.19E−51	1.12E−99	3.98E−61	2.09E+2	8.98E+3	1.01E+3	8.46E+2	6.23E−101
*F*4	*Aver*	1.18E−40	4.82E−34	6.39E−48	3.49E−2	1.48E−10	8.28E−48	3.86E−19	**1.55E−47**
	Var	9.11E−41	6.04E−34	4.62E−47	5.11E−2	4.27E−11	2.37E−49	1.17E−18	3.71E−52
*F*5	*Aver*	8.67E+2	213.11	**31.05**	1.36E+4	5.89E+3	1.08E+3	4.15E+3	48.58
	Var	9.24	12.31	3.51	2.52E+3	5.23E+2	4.53E+2	1.83E+3	**2.07E−2**
*F*6	*Aver*	9.37E−2	**1.46E−2**	5.68	421.25	13.47	2.83	832.91	1.82
	Var	1.63E−2	8.13E−3	0.98	6.39	2.18	0.56	43.21	0.50
*F*7	*Aver*	2.39	8.95E−3	9.15E−2	1.24E+3	9.97E+2	3.08E+2	6.63E−1	**1.48E−4**
	Var	0.88	4.57E−3	3.34E−2	4.10E+2	7.82E+2	8.77E+1	4.06E−1	1.49E−4
*F*8	*Aver*	**−1.28E+4**	−1.22E+4	−1.21E+4	−7.12E+3	−8.43E+3	−9.56E+3	−9.94E+3	−1.18E+4
	Var	393	474	3.98E+3	403	448	632	674	2.33E+3
*F*9	*Aver*	**0**	4.28E−70	1.73E−90	1.98E−4	9.28E−4	1.05E−6	0.48	**0**
	Var	0	1.35E−70	6.05E−91	4.83E−4	4.46E−4	3.28E−6	0.13	0
*F*10	*Aver*	**8.88E−16**	**8.88E−16**	**8.88E−16**	6.35E−2	3.97E−4	**8.88E−16**	3.95E−7	**8.88E−16**
	Var	0	0	0	1.72E−1	5.38E−4	0	3.07E−8	0
*F*11	*Aver*	1.03E−15	3.85E−25	**0**	5.58E−5	1.47E−6	5.75E−25	8.13E+3	**0**
	Var	6.67E−16	6.79E−26	0	1.36E−6	6.08E−6	3.84E−25	3.08E+3	0
*F*12	*Aver*	3.74	8.28E−2	**3.86E−2**	2.18	6.49	87.27	1.14	6.77E−2
	Var	0.83	9.73E−2	1.06E−2	0.88	1.14	4.29	0.69	4.44E−2
*F*13	*Aver*	332.37	24.39	2.19	43.78	136.28	28.18	45.63	**2.11**
	Var	9.16	3.78	0.87	3.29	2.84	2.09	4.05	1.20
*F*14	*Aver*	1.22E−35	5.07E−50	8.49E−35	6.43E−4	1.34E−4	3.33E−6	3.28	**1.88E−53**
	Var	4.97E−37	1.36E−51	2.36E−35	2.98E−4	6.77E−5	5.35E−6	0.34	**9.78E−3**
*F*15	*Aver*	89.73	85.38	4.37E−5	33.48	40.37	57.43	63.28	**21.83**
	Var	15.32	5.83	2.96E−5	3.89	3.97	3.29	5.39	**10.92**
*F*16	*Aver*	−930.18	−623.28	−508.28	−799.48	−895.39	−398.28	−689.39	**−1.25E+3**
	Var	63.94	24.25	11.26	12.04	15.39	37.38	16.73	**36.45**

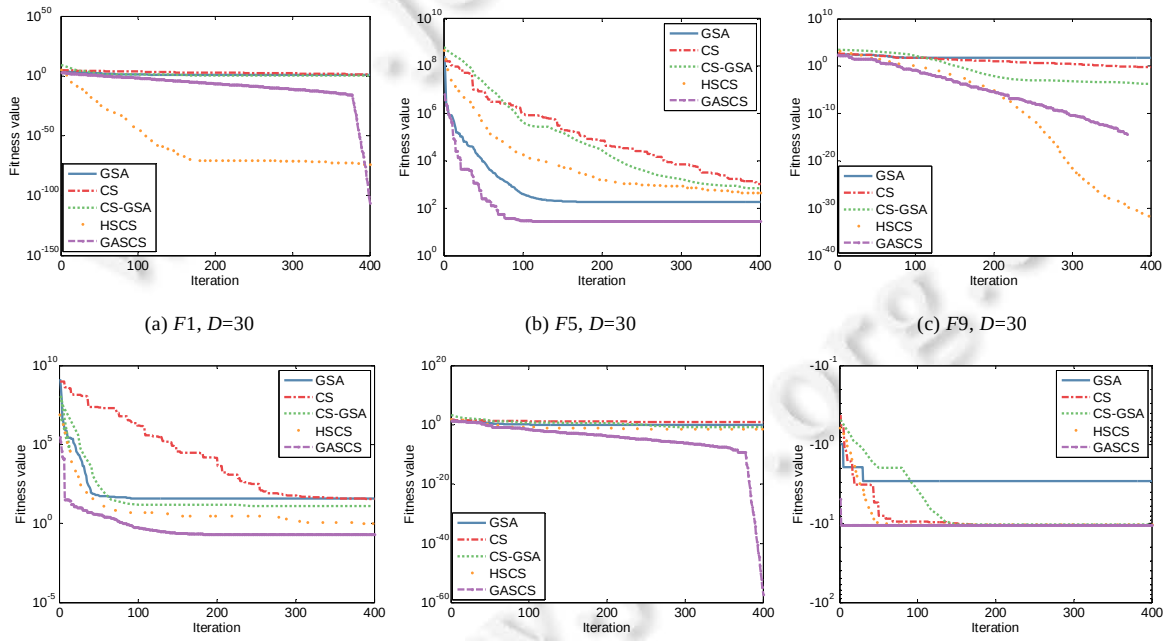
表 5 为 GASCS 算法与其他智能算法比较结果.从表中示出:除了测试函数 *F*13,GASCS 算法均能搜索到全局最优值.对于高维度多模函数 *F*13,ABC 算法和 PSO 算法均能搜索到较高的精度,优于 GASCS 算法搜索性能.从整体上考虑,GASCS 算法除了 *F*5 和 *F*13 外,均能搜索到全局最优解,与 PSO 算法、ABC 算法和 GSA 算法相比,具有更高的搜索速度和求解精度.

Table 5 Comparison of GASCS with state-of-the-art intellient heuristic algorithms

表 5 GASCS 算法与其他智能算法对比

F	PSO		ABC		GSA		HSCS		GASCS	
	Aver	*Var*	*Aver*	*Var*	*Aver*	*Var*	*Aver*	*Var*	*Aver*	*Var*
*F*1	673.42	113.25	2.45E−5	1.66E−5	41.71	11.37	6.82E−72	1.13E−72	**3.15E−104**	1.21E−104
*F*3	3.51E+4	823.17	855.21	369.14	2.01E+3	950.03	4.82E−11	9.02E−11	**4.28E−103**	3.12E−103
*F*5	2.63E+4	956.17	1.85E+3	552.43	411.18	1.19E+3	718.23	5.07E+2	**28.70**	2.39
*F*7	0.72	0.21	6.13E−3	8.66E−3	0.22	0.08	3.28E−4	6.21E−3	**9.96E−5**	3.21E−4
*F*9	198.23	13.25	5.88	11.85	46.10	8.30	7.38E−32	1.12E−32	**0**	0
*F*11	0.98	0.18	5.47E−3	1.52E−3	78.57	15.90	6.54E−23	3.28E−22	**0**	0
*F*13	2.354E−2	4.82E−2	**4.78E−3**	7.13E−3	21.12	9.28	0.79	2.23E−2	0.3011	2.12E−2
*F*15	0.58	0.63	0.19	0.28	0.74	1.96	0.37	0.55	**2.35E−5**	0
*F*17	1.00	0	1.00	0	5.67	4.21	**0.998 0**	0	**0.998 0**	0
*F*26	−8.76	2.74	−10.12	1.58	−10.53	1.56	**−10.536 4**	0	**−10.536 4**	0

为了更直观地比较 GASCS 算法与 CS 算法[2]、GSA 算法[24]、CS-GSA 算法[22]和 HSCS 算法[18]的寻优性能,图 2 示出了 5 种算法的函数适应度值的收敛曲线.为全面测试算法的性能,选择高维度单模函数、高维度多模函数和固定维度复杂多模函数中的 *F*1,*F*5,*F*9,*F*13,*F*14 和 *F*26 作为算法比较对象,同时设置算法运行的最大迭代次数为 400 次.



(a) *F*1, *D*=30 (b) *F*5, *D*=30 (c) *F*9, *D*=30

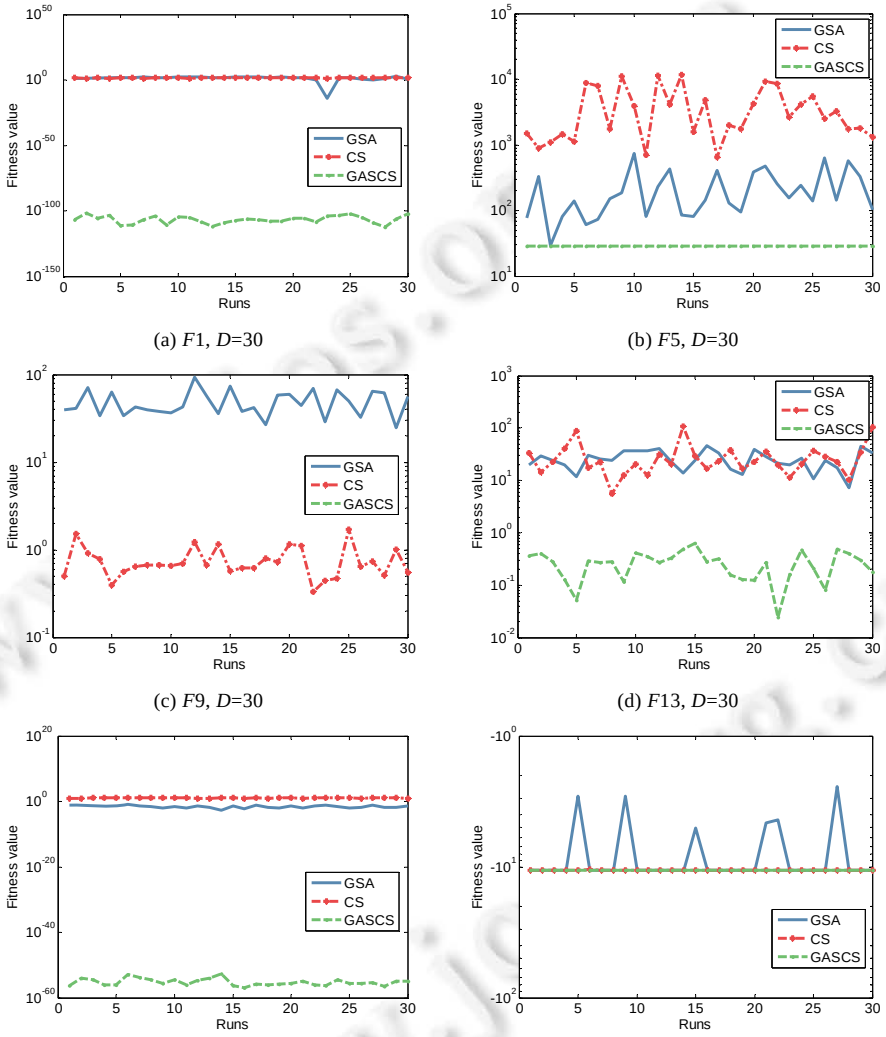
(d) *F*13, *D*=30 (e) *F*14, *D*=30 (f) *F*26, *D*=4

Fig.2 Comparison of GASCS with GSA, CS, CS-GSA, HSCS for convergence performance on fitness value

图 2 GSA、CS、CS-GSA、HSCS 和 GASCS 算法在部分函数的适应度值收敛特性比较

由图 2 示出:GASCS 算法具有更优秀的全局寻优性能,其收敛精度和搜索速度与 CS-GSA 算法[22]、HSCS 算法[18]、CS 算法[2]和 GSA 算法[24]相比较存在明显的优势.特别对于 *F*1、*F*9、*F*14、CS 算法和 GSA 算法在收

敛后期会出现迟滞和陷入局部最优值的现象.与之对比,GASCS 算法只需要较少的迭代次数就能搜索到全局最优解.对于 *F*9,由于 GASCS 算法能搜索到理论最优值 0,而数学上无法对 0 取对数,因此用数学软件作图时无法显示,这也是图 2(c)的迭代次数为 300 后无显示的原因.图 3 为 GASCS 算法、CS 算法[2]与 GSA 算法[24]独立运行 30 次的函数最优适应度值比较特性图.由图 2 和图 3 可知:GASCS 算法能搜索到 6 个测试函数中 5 个测试函数的全局最优值,并且 30 次独立运行的优化结果非常稳定.对于 *F*5,虽然 GASCS 算法未能搜索到全局最优值,但是搜索的优化值远优于 GSA 算法、CS 算法、CS-GSA 算法[22]和 HSCS 算法[18]的优化值.HSCS 算法整体优化性能弱于 GASCS 算法,但优于其他 4 种算法.而 CS 算法的寻优稳定性整体优于 GSA 算法,特别是对于 *F*14 和 *F*26 的优化,达到了较好的稳定度.



(a) *F*1, *D*=30 (b) *F*5, *D*=30

(c) *F*9, *D*=30 (d) *F*13, *D*=30

(e) *F*14, *D*=30 (f) *F*26, *D*=4

Fig. 3 Comparison of GASCS with GSA, CS on optimal fitness value

图 3 GSA、CS 和 GASCS 算法在部分函数的最优适应度值比较

为了进一步验证 GASCS 算法的有效性,考查优化算法的箱线图性能.限于篇幅,选取 *F*5 和 *F*7,并与 ACS, NNCS 和 HeCOS 算法比较,具体结果如图 4 所示.由图 4 可知:GASCS 算法能搜索到最佳的优化解,并且解的稳定性最好.综上所述,GASCS 算法与其他智能算法相比具有更好的全局寻优性能,更高的求解精度及更快的收

敛速度,进一步验证了本文提出算法的有效性和优越性.

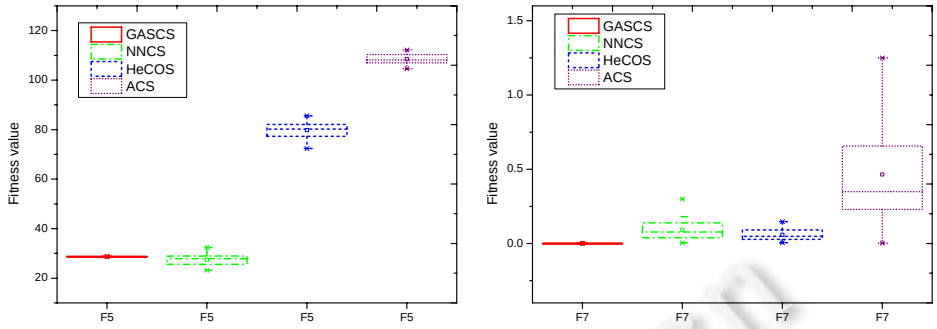


Fig.4 Comparison of GASCS with ACS, NNCS, HeCOS on box-plot

图 4 ACS、NNCS、HeCOS 和 GASCS 算法在部分函数的箱线图比较

3.4 GASCS算法复杂度分析

对于一种给定的优化算法,不仅需要评估其寻优性能,而且需要评估运行效率.本节主要从算法的时间复杂度评估优化算法的运行效率.优化算法的时间复杂度计算主要集中在优化函数的评价阶段,其算法复杂度主要体现为优化函数的评价次数.设置算法最大进化代数为 W,优化函数的种群个数为 N,优化函数的维度为 D.传统 CS 算法的计算复杂度为 $T(CS)=O(W\times(f(D)+N))$.GASCS 算法与传统 CS 算法比较可知,步骤 3~步骤 6 是增加的算法步骤.

- 步骤 3. 计算 G_r 和 $f_{r,best}$ 和最差值 $f_{r,worst}$ 的复杂度为 $T(G_r+f_{r,best}+f_{r,worst})=O(W\times1)$.
- 步骤 4. 计算 $q_{r,m}$,$Q_{r,m}$ 的复杂度为 $T(q_{r,m}+Q_{r,m})=T(q_{r,m})+T(Q_{r,m})=O(W\times(f(D)+N))$.
- 步骤 5. 计算 $F_{r,m}^{(j)}$ 和 $a_{r,m}^{(j)}$ 的复杂度为 $T(F_{r,m}^{(j)}+a_{r,m}^{(j)})=T(F_{r,m}^{(j)})+T(a_{r,m}^{(j)})=O(W\times(f(D)+N))$.

因此,GASCS 算法总体的计算复杂度为

$$T(GASCS)=T(CS)+T(G_r+f_{r,best}+f_{r,worst})+T(q_{r,m}+Q_{r,m})+T(F_{r,m}^{(j)}+a_{r,m}^{(j)})=O(W\times(f(D)+N)).$$

为了比较 GASCS 算法与 CS 算法的计算复杂度,将每种算法独立运行 30 次,计算各种算法运行时间的平均值 T_m,具体结果如表 1 所述.由于 GASCS 算法增加了计算引力合力、计算引力惯性质量、计算引力加速度等操作,因此会适当增加计算复杂度.但是 GASCS 算法引入了万有引力定律和牛顿第二定律,对布谷鸟算法中的 Levy 飞行随机游动和偏好随机游动同时利用优化个体间存在的万有引力进行加速搜索,同时利用概率变异增大种群的多样性,避免算法执行后期陷入局部极值点而出现的迟滞现象,因此能更快搜索到全局最优解.对于高维度单模函数和高维度多模函数,特别是高维度多模函数,例如 F_{16},GASCS 算法的优化效率相比于传统 CS 算法有显著提高.由表 1 和本节计算复杂度理论分析可知:本文提出的 GASCS 算法计算复杂度与传统 CS 算法的复杂度一致,同属于一个数量级,并未增加算法复杂度.

4 结 论

布谷鸟算法是一种新型的智能优化算法,具有通用性强、控制参数少、算法执行速度快等优点,已经成为目前学者研究的热点问题.但是该算法仍然存在全局搜索和局部搜索不平衡导致算法搜索后期陷入局部极值最优、搜索速度较慢、收敛精度较低的局限性.本文将布谷鸟巢穴视为有质量的物质个体,利用万有引力定律和牛顿第二定律,对布谷鸟算法中的 Levy 飞行随机游动和偏好随机游动同时利用优化个体间存在的万有引力进行加速搜索,并且采用概率变异增大种群的多样性,有效地平衡布谷鸟算法的全局搜索能力和局部开采能力,避免算法执行后期陷入局部极值点而出现的迟滞现象,提高了算法的全局搜索效率和收敛精度.通过用优化算法对 26 个基准测试函数进行了性能仿真,结果表明,提出的 GASCS 算法与其他改进智能优化算法相比具有更优秀的全局寻优性能、更好的鲁棒性、更快的搜索速度和更高的收敛精度.

References:

[1] Yang XS, Deb S. Engineering optimisation by cuckoo search. Int'l Journal of Mathematical Modelling & Numerical Optimisation, 2010,1(4):330−343.

[2] Kordestani JK, Firouzjaee HA, Meybodi MR. An adaptive bi-flight cuckoo search with variable nests for continuous dynamic optimization problems. In: Proc. of the Applied Intelligence. 2017. 1−21.

[3] Liang S, Feng T, Sun G. Sidelobe-level suppression for linear and circular antenna arrays via the cuckoo search—Chicken swarm optimisation algorithm. Iet Microwaves Antennas & Propagation, 2017,11(2):209−218.

[4] Gandomi AH, Yang XS, Alavi AH. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. Engineering with Computers, 2013,29:17−35.

[5] Peng BR, Ho KC, Liu YH. A novel and fast MPPT method suitable for both fast changing and partially shaded conditions. IEEE Trans. on Industrial Electronics, 2018,65(4):3240−3251.

[6] Singh SS, Fernandez E. Modeling, size optimization and sensitivity analysis of a remote hybrid renewable energy system. Energy, 2018,143(15):719−731.

[7] Wang MW, Wan YC, Ye ZW, Lai XD. Remote sensing image classification based on the optimal support vector machine and modified binary coded ant colony optimization algorithm. Information Sciences, 2017,402:50−68.

[8] Xiao L, Shao W, Yu M, Ma J, Jin CJ. Research and application of a hybrid wavelet neural network model with the improved cuckoo search algorithm for electrical power system forecasting. Applied Energy, 2017,198(15):203−222.

[9] Zhou J, Yao X. A hybrid approach combining modified artificial bee colony and cuckoo search algorithms for multi-objective cloud manufacturing service composition. Int'l Journal of Production Research, 2017,55(16):1−20.

[10] Wang Z, Li Y. Irreversibility analysis for optimization design of plate fin heat exchangers using a multi-objective cuckoo search algorithm. Energy Conversion & Management, 2015,10(1):126−135.

[11] Zhou J, Yao X. Multi-objective hybrid artificial bee colony algorithm enhanced with Lévy flight and self-adaption for cloud manufacturing service composition. Applied Intelligence, 2017,47(3):721−742.

[12] Balasubbareddy M, Sivanagaraju S, Suresh CV. Multi-objective optimization in the presence of practical constraints using non-dominated sorting hybrid cuckoo search algorithm. Engineering Science & Technology, An Int'l Journal, 2015,18(4):603−615.

[13] Ong P. Adaptive cuckoo search algorithm for unconstrained optimization. The Scientific World Journal, 2014,14(9):1−8.

[14] Mareli M, Twala B. An adaptive cuckoo search algorithm for optimisation. Applied Computing & Informatics, 2017,9(6):1−9.

[15] Wang LJ, Yin YL, Zhong YW. Cuckoo search with varied scaling factor. Frontiers of Computer Science, 2015,9(4):623−635.

[16] Wang LJ, Yin YL, Zhong YW. Cuckoo search algorithm with dimension by dimension improvement. Ruan Jian Xue Bao/Journal of Software, 2013,24(11):2687−2698 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4476.htm [doi: 10.3724/SP.J.1001.2013.04476]

[17] Walton S, Hassan O, Morgan K, Brown MR. Modified cuckoo search: A new gradient free optimisation algorithm. Chaos Solitons & Fractals, 2011,44(9):710−718.

[18] Wang GG, Gandomi AH, Zhao XJ, Chu HC. Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. Soft Computing, 2016,20(1):1−13.

[19] Guo J, Sun Z, Tang H, Jia XJ, Wang S, Yan XH, Ye GL, Wu GH. Hybrid optimization algorithm of particle swarm optimization and cuckoo search for preventive maintenance period optimization. Report, Discrete Dynamics in Nature and Society, 2016. [doi: 10.1155/2016/1516271]

[20] Li XT, Yin MH. Parameter estimation for chaotic systems using the cuckoo search algorithm with an orthogonal learning method. Chinese Physics B, 2012,21(5):113−118.

[21] Li XT, Wang J, Yin MH. Enhancing the performance of cuckoo search algorithm using orthogonal learning method. Neural Computing & Applications, 2014,24(6):1233−1247.

[22] Naik MK, Panda R. A new hybrid CS-GSA algorithm for function optimization. In: Proc. of the Int'l Conf. on Electrical, Electronics, Signals, Communication and Optimization. Visakhapatnam: Andhra Pradesh, 2015. 1−6.

[23] Ma W, Sun ZX. A global Cuckoo optimization algorithm using coarse-to-fine search. Acta Electronica Sinica, 2015,43(12): 2429−2439 (in Chinese with English abstract).

[24] Rashedi E, Nezamabadi-Pour H, Saryazdi S. GSA: A gravitational search algorithm. Intelligent Information Management, 2012, 4(6):390−395.

[25] Yazdani S, Nezamabadi-Pour H, Kamyab S. A gravitational search algorithm for multimodal optimization. Swarm & Evolutionary Computation, 2014,14:1−14.

[26] Zhao FQ, Xue FL, Zhang Y, Ma W, Zhang C, Song HB. A hybrid algorithm based on self-adaptive gravitational search algorithm and differential evolution. Expert Systems with Applications, 2018,113:515−530.

[27] Wang F, He XS, Luo L, Wang Y. Hybrid optimization algorithm of PSO and cuckoo search. In: Proc. of the Int'l Conf. on Artificial Intelligence, Management Science and Electronic Commerce. Dengfeng: IEEE, 2011. 1172−1175.

[28] Borshevsky M. Stability analysis of the particle swarm optimization without stagnation assumption. IEEE Trans. on Evolutionary Computation, 2016,20(5):814−819.

[29] Li X, Wang J, Yin M. Enhancing the performance of cuckoo search algorithm using orthogonal learning method. Neural Computing & Applications, 2014,24(6):1233−1247.

[30] Naik MK, Panda R. A novel adaptive cuckoo search algorithm for intrinsic discriminant analysis based face recognition. Applied Soft Computing, 2016,38:661−675.

[31] Ding XM, Xu ZK, Cheung NJ, Liu XH. Parameter estimation of TakagiSugeno fuzzy system using heterogeneous cuckoo search algorithm. Neurocomputing, 2015,151:1332−1342.

[32] Wang LJ, Zhong YW, Yin YL. Nearest neighbour cuckoo search algorithm with probabilistic mutation. Applied Soft Computing, 2016,49:498−509.

[33] Ozturk C, Hancer E, Karaboga D. A novel binary artificial bee colony algorithm based on genetic operators. Information Sciences, 2015,297:154−170.

[34] Cui LZ, Li GH, Zhu ZX, Lin QZ, Wen ZK, Lu N, Wong KC, Chen JY. A novel artificial bee colony algorithm with an adaptive population size for numerical function optimization. Information Sciences, 2017,414:53−67.

附中文参考文献:

[16] 王李进,尹义龙,钟一文.逐维改进的布谷鸟搜索算法.软件学报,2013,24(11):2687−2698. http://www.jos.org.cn/1000-9825/4476.htm [doi: 10.3724/SP.J.1001.2013.04476]

[23] 马卫,孙正兴.采用搜索趋化策略的布谷鸟全局优化算法.电子学报,2015,43(12):2429−2439.

傅文渊(1982－),男,硕士,主要研究领域为智能信号优化与智能学习控制,电路与系统设计,嵌入式系统设计.