计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.181102165

求解函数优化问题的改进布谷鸟搜索算法

李 煜[1] 尚志勇[2] 刘景森[3]
1 河南大学商学院管理科学与工程研究所 河南 开封 475004
2 河南大学商学院 河南 开封 475004
3 河南大学复杂智能网络系统研究所 河南 开封 475004

摘 要 在工程优化中,大多问题是连续优化问题,即函数优化问题。针对布谷鸟算法求解函数优化问题时存在的收敛速度慢、求解精度不高和易陷入局部最优等问题,文中提出非线性惯性权重对数递减和随机调整发现概率的布谷鸟搜索算法(Cuckoo Search Algorithm with Logarithmic Decline of Nonlinear Inertial Weights and Random Adjustment Discovery Probability,DWCS)。首先,在布谷鸟寻窝的路径和位置更新公式中,设计一种随进化迭代次数非线性递减的惯性权重来改进鸟巢位置的更新方式,以协调布谷鸟算法的探索和开发能力;其次,引入随机调整发现概率代替固定值发现概率,使较大和较小的发现概率随机出现,从而有利于平衡算法的全局探索和局部开发能力,加快算法收敛速度,增加种群多样性;最后,分析对数递减参数和随机调整发现概率,选取对数递减最佳参数组合和随机调整发现概率的最佳取值范围,此时,函数的优化效果最好。与 BA,CS,PSO,ICS 算法相比,所提算法极大地提高了寻优精度,显著地减少了迭代次数,有效地提高了收敛速度和鲁棒性。在 16 个测试函数中,DWCS 均能收敛到全局最优解,证明了 DWCS 在求解连续复杂函数优化问题上具有较强的竞争力。

关键词:布谷鸟搜索算法;参数选取;对数递减;发现概率;函数优化
中图法分类号 TP301.6

Improved Cuckoo Search Algorithm for Function Optimization Problems

LI Yu[1], SHANG Zhi-yong[2] and LIU Jing-sen[3]
1 Institute of Management Science and Engineering, Business School of Henan University, Kaifeng, Henan 475004, China
2 Business School of Henan University, Kaifeng, Henan 475004, China
3 Institute of Complex Intelligent Network System, Henan University, Kaifeng, Henan 475004, China

Abstract In engineering optimization, most problems are continuous optimization problems, that is function optimization problems. Aiming at the problems of slow convergence speed, low precision and easy to fall into local optimization in the later stage of Cuckoo algorithm, this paper proposed an improved Cuckoo search algorithm based on the logarithmic decline of nonlinear inertial weights and the random adjustment discovery probability. Firstly, in the update formula of the path and position of the cuckoo homing nest, a update method that inertia weight decreases nonlinearly with the number of evolutionary iterations is designed to improve the nest location and coordinate the abilities of exploration and exploitation. Secondly, the discovery probability with random adjustment is introduced to replace the discovery probability of fixed value to make the larger and smaller discovery probability appear randomly, which is beneficial to balancing the global exploration and local exploitation of the algorithm, accelerating the convergence speed, and increasing the diversity of the population. Finally, the logarithmic decreasing parameter and stochastic adjustment discovery probability are analyzed and tested, and the optimal parameter combination of logarithmic decrement and the optimal range of stochastic adjustment discovery probability are selected. At this time, the optimization effect of the function is the best. Compared with other evolutionary algorithms (BA, CS, PSO, ICS), DWCS greatly improves the precision of optimization, significantly reduces the number of iterations, and effectively improves the convergence speed and robustness. In 16 test functions, DWCS can converge to the global optimal solution, which proves that DWCS has a strong competitive power in solving

到稿日期:2018-11-23 返修日期:2019-04-01 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:国家自然科学基金(71601071);教育部人文社科青年基金(15YJC630079);河南省重点研发与推广专项(182102310886);河南省科技攻关重点项目(162102110109)
This work was supported by the National Natural Science Foundation of China (71601071), Humanities and Social Sciences Youth Fund of the Ministry of Education (15YJC630079), Special Research and Development and Promotion Project of Henan Province (182102310886) and Key Project of Henan Province Science and Technology (162102110109).
通信作者:李煜(lyhenu@163.com)

the optimization problem of continuous complex functions.

Keywords Cuckoo search algorithm, Parameter selection, Logarithmic decreasing, Discovery probability, Function optimization

1 引言

在工程优化中,大部分问题是连续优化问题,即函数优化问题[1-2]。这些优化问题涉及范围非常广,包含单峰函数到多峰函数、低维函数到高维函数和一些特定情况下的优化问题。因此,对这些问题进行研究有着重要的理论价值和实际意义。虽然连续优化问题的数学描述简单,但是当问题的目标函数拓扑结构比较复杂、决策变量的数目比较多时,其求解难度会显著增加。此外,随着时代的发展,我们所要解决的问题往往是不连续、不可微、非线性、非凸的、多极值的,这进一步增加了问题的求解难度[3-4]。采用传统优化算法(如最速下降法和共轭梯度法等)很难获得理想的优化效果。然而,元启发式算法对于求解此类优化问题非常有效。大多数元启发式算法来源于对生物行为、物理性质、化学过程的模拟,如模拟自然界生物进化的遗传算法[5]、模拟鸟类和鱼类群体运动行为的粒子群算法[6]、模拟固体冷却过程的模拟退火算法[7]、模拟蜜蜂采蜜过程的蜂群算法[8]、模拟即兴音乐创作过程的和声搜索算法[9]、模拟果蝇觅食行为的果蝇算法[10]、模拟萤火虫发光生物特性的萤火虫算法[11]等。这些群智能算法已被广泛应用于求解大量的实际问题。近年来,人们提出了多种模拟生物行为的新颖算法,如布谷鸟算法,这为解决复杂函数的优化问题提供了一种新的途径。

2009 年,剑桥大学的 Yang 和拉曼工程大学的 Deb 模拟布谷鸟的寻窝产卵行为,提出了一种新的智能优化算法——布谷鸟搜索(Cuckoo Search,CS)算法[12-13]。CS 算法由于具有结构简单、控制参数少、搜索路径优、全局寻优能力强等优点,备受学者关注,相关的科研成果也日益倍增。目前,CS 算法已被广泛应用于许多领域,包括函数优化[14-15]、求解旅行商问题[16]、人脸识别[17]、PID 控制器调整[18]、混沌系统参数估计[19]等。

CS 算法虽然较其他智能算法有很多优点,但后期求解函数问题时仍存在收敛速度偏慢、求解精度不高和易陷入局部最优等问题。在求解实际问题的过程中,对于不同的问题,在不同的阶段对算法全局探索和局部搜索能力的要求也不同。所以,如何平衡全局探索和局部搜索之间的关系,使算法在迭代前期能很快地确定全局最优值的大致范围,在迭代后期可以对局部区域进行精细搜索,具有重要意义。针对上述问题,国内外学者对基本布谷鸟搜索算法做了相应的改进。例如,王李进等[20]采用逐维改进的 CS 算法,提高了 CS 算法的收敛速度和寻优精度,该算法在求解连续函数优化问题上具有一定的竞争优势。马卫等[21]采用搜索趋化策略的布谷鸟算法,提升了算法的整体性能,在解决多峰和高维函数优化问题上更具竞争力。Valian 等[22]通过动态调整步长因子与发现概率的方法,提升了算法的收敛速度和寻优精度。Zheng[23]针对布谷搜索算法收敛速度慢的问题,提出了一种基于高斯分布的布谷鸟搜索优化算法,改进后算法的收敛速度和精度均优于基本布谷鸟算法。李荣雨等[24]针对 CS 算法后期存在收敛速度慢、易陷入局部最优等问题,提出了一种新的自适应步长的布谷鸟搜索算法,改进后算法的各方面性能均得到显著提升。Wang 等[25]在莱维飞行的更新公式中加入服从[0,1]均匀分布的压缩因子,使得算法效果优于基本 CS 算法。Jin 等[26]对发现概率和莱维飞行步长进行动态自适应改进,改进后的算法具有更快的收敛速度和更高的求解精度。Prajapati 等[27]提出了一种混合的 CS-PSO 优化算法,对 11 个单峰和多峰非线性基准函数进行测试的结果表明,混合 CS-PSO 算法的性能优于 PSO,DE 和 CS 算法。张子成等[28]提出了一种基于模拟退火的自适应离散型布谷鸟算法来求解旅行商问题,改进后算法的求解精度高,在 TSP 的求解上展现了良好的性能。Zhang 等[29]提出了一种基于动态局部搜索的混合多目标布谷鸟搜索方法。为了提高搜索能力,其引入了动态局部搜索,改进后的算法在收敛性、扩展性和分布性等方面性能更优。傅文渊[30]针对布谷鸟算法存在维度间相互耦合问题,提出了均衡单进化的布谷鸟算法。改进后的算法提升了全局寻优性能、搜索速度和收敛精度。Mareli 等[31]提出 3 种新的基于动态发现概率的布谷鸟搜索算法,并通过 10 个测试函数进行验证,仿真结果表明,具有指数增长发现概率的布谷鸟搜索算法优于其他布谷鸟搜索算法。Salgotra 等[32]为了提高 CS 算法的勘探和开发性能,提出了 3 种改进的 CS 算法,并通过 CEC 2015 基准函数进行测试,结果表明,改进后的算法收敛速度更快,解的质量更高,鲁棒性更强。上述改进策略在一定程度上提高了 CS 算法的收敛速度或解的质量。但是,对于 CS 算法局部搜索性能弱,特别是多模复杂函数全局寻优存在收敛速度慢、求解精度不高等问题,上述改进策略的优化效果仍然有限。为了克服 CS 算法的缺点,提高收敛速度和解的质量,本文提出了一种非线性惯性权重对数递减和随机调整发现概率 pa 的布谷鸟算法,并对对数递减参数和随机调整发现概率 pa 进行了大量测试,选取 $r_{\min}=0$,$r_{\max}=0.1$,$\beta=1$ 作为对数递减的最佳参数组合,选取 $pa\in[0.15,0.55]$作为随机调整发现概率的最佳取值范围,此时,函数的优化效果最好。本文设计了一种随进化迭代次数非线性递减的惯性权重来改进鸟巢位置的更新方式,以协调 CS 算法的探索和开发能力;同时引入随机调整的发现概率,使较大和较小的发现概率随机出现,有利于平衡算法的全局探索和局部开发能力,加快算法收敛速度,增加种群的多样性。实验结果表明,与 BA,CS,PSO,ICS 相比,DWCS 能收敛到理论最优值,极大地提高了寻优精度,显著减少了迭代次数,有效提高了收敛速度和鲁棒性,在求解连续复杂函数优化问题上具有较强的竞争力。

2 基本布谷鸟算法

根据昆虫学家的长期研究发现,一部分布谷鸟以寄生的方式养育幼鸟,它们不筑巢,而是将自己的卵产在其他鸟的巢中(通常为黄莺、云雀等),由其他鸟(义亲)代为孵化和育雏。然而,一旦这些外来鸟蛋被宿主发现,宿主便会将其抛弃或新

筑鸟巢[33]。布谷鸟算法源于对布谷鸟繁育行为的模拟，为了简化自然界中布谷鸟的繁衍习性，Yang 等将布谷鸟的产卵行为假设为 3 种理想状态。

(1)布谷鸟一次只产一个卵，并随机选择鸟窝位置来孵化它。

(2)在随机选择的一组鸟窝中，最好的鸟窝将会被保留到下一代。

(3)可选择的寄生巢的数量是固定的，寄生巢主人发现外来鸟蛋的概率为 pa，其中 $0\leqslant pa\leqslant 1$。

基于这 3 种理想状态，Yang 等采用式(1)对下一代鸟巢位置 $X^{(t+1)}$ 进行更新：

$$X_i^{(t+1)}=X_i^t+\alpha\oplus Levy(\lambda) \tag{1}$$

其中，X_i^t 表示第 $i(i=1,2,\cdots,n)$ 个鸟巢在第 t 代的位置；$\oplus$ 表示点对点乘法；α 表示步长控制量，用于控制步长大小，通常情况下，取 $\alpha=1$。$L(\lambda)$ 为 $Levy$ 随机搜索路径，属于随机行走，采用莱维飞行机制[34]，其行走的步长满足一个重尾的稳定分布。而随机步长为 $Levy$ 分布：

$$Levy:\ \mu=t^{-\lambda},1<\lambda\leqslant 3 \tag{2}$$

基本布谷鸟搜索算法先按照式(1)对下一代的鸟巢位置进行更新，并计算目标函数的适应度值。如果该值优于上一代的目标函数值，则更新鸟巢位置，否则保持原来位置不变。位置更新后，用随机产生的服从 0 到 1 均匀分布的数值 R 与鸟巢主人发现外来鸟蛋的概率 pa 进行比较；若 $R>pa$，则随机改变 $X_i^{(t+1)}$，反之不变。最后保留测试值较好的一组鸟窝位置 $Z_i^{(t+1)}$，记为 $X_i^{(t+1)}$。判断算法是否满足设置的最大迭代次数；若满足，结束迭代寻优，输出全局最优值 $f_{\min}$；否则，继续迭代寻优。

3　改进的布谷鸟算法(DWCS)

在基本 CS 算法中，莱维飞行随机游动和偏好随机游动是两个重要的搜索策略。莱维飞行自身的特征，使得搜索过程具有较好的全局探索能力和随机性。然而，当求解多峰高维函数优化问题时，由于随机游动策略的盲目性，将导致算法无法迅速收敛到全局最优值，且局部搜索能力弱，求解精度不高。对此，本文分别提出非线性惯性权重对数递减的策略和随机调整发现概率的策略来弥补该算法在求解复杂函数优化问题时的局限性，从而协调 CS 算法的探索和开发能力，加快收敛速度，提高求解精度，增强鲁棒性。

惯性权重是 CS 算法最重要的参数之一，其控制历史因素对当前状态的影响程度。在基本 CS 算法中，布谷鸟寻窝的路径和位置是随机的，以父代位置信息为参考进行更新。为提高 CS 算法的性能，在布谷鸟寻窝的路径和位置更新公式中，设计了一种非线性递减的惯性权重来有效平衡 CS 算法的全局探索和局部开发能力，并对对数递减参数进行了大量测试，选取了最佳参数组合 $r_{\min}=0$，$r_{\max}=0.1$，$\beta=1$。在基本的 CS 算法中，发现概率是非常重要的参数，然而固定的发现概率不利于算法全局探索和局部开发性能的平衡以及种群的多样性。为了克服这个弊端，本文引入随机调整的发现概率，使较大、较小的发现概率随机出现，以加快算法收敛速度，增加种群多样性。经过大量测试，选取 $pa\in[0.15,0.55]$ 作为随机调整发现概率的最佳取值范围。

3.1　非线性惯性权重对数递减策略

动态变化惯性权重主要有以下几类。以下公式中，t 为当前迭代次数，$MaxNumber$ 为最大迭代次数，$\alpha_{\max}$ 和 $\alpha_{\min}$ 为 α 的初值和终值。

(1)惯性权重线性递减策略

Shi 等为了协调算法全局探索和局部开发能力之间的平衡关系，将惯性权重引入粒子群算法，并采用惯性权重线性递减策略[35]。递减策略如下：

$$\alpha_{(t)}=\alpha_{\max}-(\alpha_{\max}-\alpha_{\min})\frac{MaxNumber-t}{MaxNumber} \tag{3}$$

(2)模糊调整的自适应惯性权重

Shi 等[36]采用模糊系统提出了具有动态调整特征的自适应性惯性权重。由于该惯性权重设置的参数较多，大大增加了算法的复杂度和实现难度。

(3)随机调整的惯性权重

2001 年，Eberhart 等[37]提出了一种随机调整动态惯性权重的方法，该惯性权重的值是根据每次迭代从高斯分布中随机选取的，较大和较小的惯性权重值随机出现，可以有效平衡算法全局探索和局部搜索之间的关系。

(4)非线性惯性权重递减策略

非线性惯性权重递减策略是指惯性权重从较大值非线性递减到较小值的一种变化过程[38]。其中具有代表性的有呈指数递减、高斯递减和对数递减的惯性权重。

1)指数递减的惯性权重

文献[39]针对基本粒子群算法在后期存在局部搜索能力差、易陷入局部最优解的缺点，提出了基于惯性权重指数递减的粒子群优化算法。其惯性权重的表达式如下：

$$\alpha_{(t)}=\alpha_{\min}(\alpha_{\max}/\alpha_{\min})^{(1-t/MaxNumber)} \tag{4}$$

2)高斯递减的惯性权重

文献[40]为了有效平衡粒子群算法全局探索和局部搜索的能力，提出了一种基于高斯递减惯性权重的粒子群优化算法。仿真结果表明，改进的算法在收敛速度和执行效率方面均有很大的提升。其惯性权重表达式如下：

$$\alpha_{(t)}=\alpha_{\min}+(\alpha_{\max}-\alpha_{\min})\exp\left[-\frac{t^2}{(k\times MaxNumber)^2}\right] \tag{5}$$

其中，k 为常数，改变 k 值就是调整曲线的扩展常数，从而改变曲线的变化率。

3)对数递减的惯性权重

文献[41]针对粒子群算法收敛速度慢和易陷入局部最优的问题，提出了基于惯性权重对数递减的粒子群算法，并引入了对数调整系数。仿真结果表明，改进的算法可以提高全局收敛速度，并且具有一定的鲁棒性。其惯性权重的表达式如下：

$$\alpha_{(t)}=\alpha_{\max}-\eta\times(\alpha_{\max}-\alpha_{\min})\times\log_{MaxNumber}^{t} \tag{6}$$

其中，η 为对数调整系数。

图 1 给出 $t=1:100$，$MaxNumber=100$，$\alpha_{\min}=0.4$，$\alpha_{\max}=0.9$，$\eta=1$ 时，线性、指数、高斯、对数递减策略的迭代次数和惯性权重的关系曲线图。从图 1 可以看出，在迭代前期，对数

递减的惯性权重曲线比线性、指数、高斯递减的惯性权重曲线下降得更快，有利于全局探索、快速找到最优解的大致范围；在迭代后期，对数递减的惯性权重曲线相比另外 3 种惯性权重策略的曲线更平缓，有利于局部精细搜索，从而找到全局最优解。

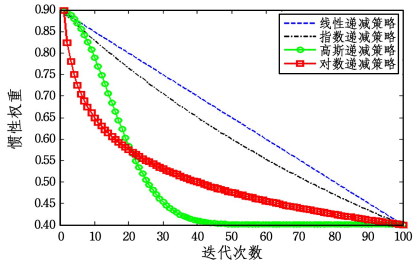


图 1 惯性权重曲线图

Fig. 1 Inertial weight graph

为了提高 CS 算法的性能，在布谷鸟寻窝的路径和位置更新公式中引入惯性权重：

$$X_i^{(t+1)}=w^t X_i^t+\alpha\oplus Levy(\lambda),i=1,2,\cdots,n \tag{7}$$

惯性权重的加入可以有效平衡全局探索和局部搜索之间的关系，使算法在前期有较强的全局探索能力，在后期有较强的局部开发能力，以加快算法的收敛速度和寻优精度。从图 1 中可以看出，4 种递减策略中，对数递减策略更具优势。虽然 4 种递减策略在不同程度上都可以协调算法的探索和开发能力，且会对算法的收敛速度和精度产生一定的影响。但是，如果在寻优初期找不到较理想的位置，那么随着惯性权重的减小，全局探索能力逐渐变弱，局部开发能力逐渐增强，算法极易陷入局部最优。基于以上分析，本文提出了一种非线性惯性权重对数递减的策略，即：

$$w_{(t)}=r_{\max}-\beta\times(r_{\max}-r_{\min})\times\log_{MaxNumber}^{t}-\varphi\times betarnd() \tag{8}$$

其中，t 为当前迭代次数；$MaxNumber$ 为最大迭代次数；$r_{\min}$ 是惯性权重的最小值，$r_{\max}$ 是惯性权重的最大值；β 为对数偏离系数，$0<\beta<1$ 时称为对数压缩系数，$\beta>1$ 时称为对数膨胀系数；φ 为惯性调整系数；$betarnd()$ 是服从贝塔分布产生的 $(0,1)$ 之间非对称分布的随机数。惯性权重 $w_{(t)}$ 表达式中的前两项利用对数递减的特性，使 $w_{(t)}$ 随迭代次数的增加而减小；第三项利用贝塔分布使惯性权重偏离，使 $w_{(t)}$ 的取值更灵活；同时，在 $betarnd()$ 前加 φ 是为了使惯性权重 $w_{(t)}$ 的偏离程度更合理。综上所述，非线性惯性权重对数递减惯性策略不仅可以使布谷鸟算法在前期拥有较强的全局探索能力，在后期拥有较强的局部开发能力，而且能在一定程度上避免算法陷入局部最优。

3.2 动态惯性权重系数的变化范围对函数优化效果的影响

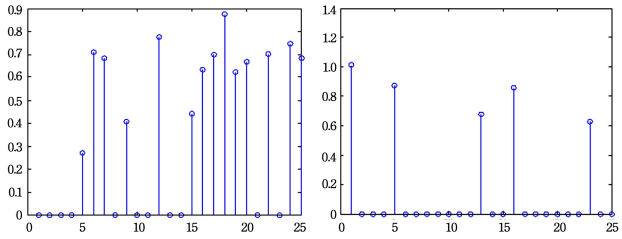
本文对 $w_{(t)}$ 中的系数 $r_{\min}$，$r_{\max}$（$r_{\min}$，$_{\max}\in[0,-1]$，$r_{\min}<r_{\max}$）和 β（$0\leqslant\beta<1.8$）进行两种测试。首先，本文取 $r_{\min}=0.4$，$r_{\max}=0.9$，β 的值从 0 增加到 1.7，由于测试系数 β 的变化步长为 0.1 时取值次数太多，为了方便起见，我们只在表 1 中列出 $0\leqslant\beta<1$ 和 $1\leqslant\beta<1.8$ 两种情况下的最优值，并用函数 $f_1(x)$，$f_2(x)$，$f_3(x)$ 对以上两种情况进行检验。为了确保惯性权重在整体递减下波动较小，φ 的取值要足够小。设置参数：种群大小 $N=25$，最大迭代次数 $MaxNumber=100$，维数 $D=10,50,100$，每个函数评价 30 次。表 1 给出了 β 系数的变化对函数优化效果的影响。

表 1 系数的变化对函数优化效果的影响($D=10$)

Table 1 Effect of change of coefficient on function optimization effect($D=10$)

β	$f_1(x)$		$f_2(x)$		$f_3(x)$	
	M	收敛率	M	收敛率	M	收敛率
$0\leqslant\beta<1$	d	44	d	4	d	0
$1\leqslant\beta<1.8$	c	80	c	20	c	36

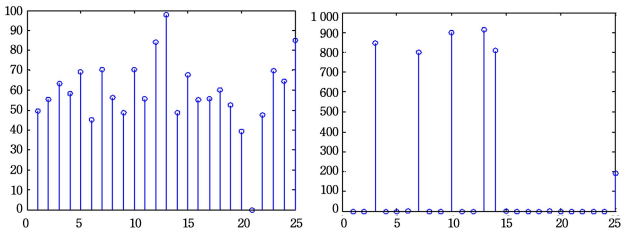
从表 1 可以看出，在 10 维的情况下，当 $1\leqslant\beta<1.8$ 时，函数 $f_1(x)$，$f_2(x)$，$f_3(x)$ 取得理论最优值的最小迭代次数更少（分别在 50，50，100 代以内）、收敛率更高（分别为 80%，20%，36%）；当 $0\leqslant\beta<1$ 时，函数 $f_1(x)$，$f_2(x)$，$f_3(x)$ 取得理论最优值的最小迭代次数分别为 100 代以内、100 代以内和在设置的最大迭代次数内无法收敛到最优，收敛率分别为 44%，4%，0。同样，在 50 维和 100 维的情况下，当 $1\leqslant\beta<1.8$ 时，取得最优值的最小迭代次数和收敛率也比 $0\leqslant\beta<1$ 时更优。同时，从图 2－图 4 中可以看出，当 $1\leqslant\beta<1.8$ 时，函数 $f_1(x)$，$f_2(x)$，$f_3(x)$ 取得理论最优值的次数明显比 $0\leqslant\beta<1$ 时多。



(a)$0\leqslant\beta<1$ (b)$1\leqslant\beta<1.8$

图 2 $f_1(x)$ 函数的收敛图

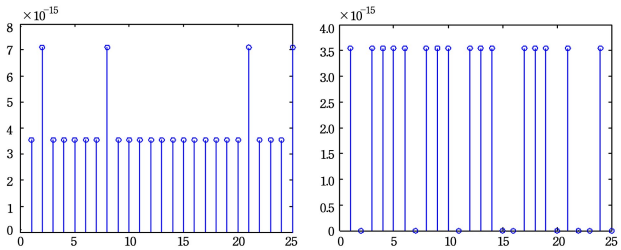
Fig. 2 Convergence diagram of $f_1(x)$ function



(a)$0\leqslant\beta<1$ (b)$1\leqslant\beta<1.8$

图 3 $f_2(x)$ 函数的收敛图

Fig. 3 Convergence diagram of $f_2(x)$ function



(a)$0\leqslant\beta<1$ (b)$1\leqslant\beta<1.8$

图 4 $f_3(x)$ 函数的收敛图

Fig. 4 Convergence diagram of $f_3(x)$ function

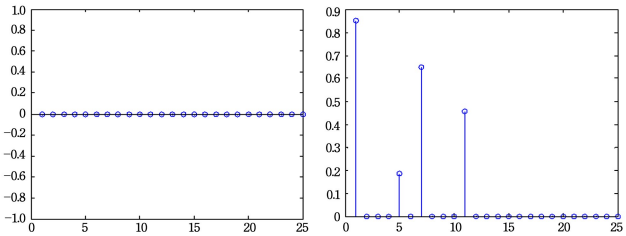
综上，在 10 维、50 维、100 维的情况下，$1\leqslant\beta<1.8$ 比 $0\leqslant\beta<1$ 对函数的优化效果更好。因此，令 $1\leqslant\beta<1.8$，在实验中取 $\beta=1$，$r_{\min}$ 和 $r_{\max}$（$0\leqslant r_{\min}\leqslant r_{\max}\leqslant 1$）从 0 取到 1，用数字 1 表示 $r_{\min}=0$，$0<r_{\max}\leqslant 1$，用数字 2 表示 $r_{\min}=0.1$，$0.1<r_{\max}\leqslant 1$，以此类推，用数字 8 表示 $r_{\min}=0.7$，$0.7<r_{\max}\leqslant 1$。例如，第一次令 $r_{\min=}0$，$r_{\max}$ 从 0.1 取到 1（第一次取 0.1，第二次取 0.2 以此类推至取到 1 结束），第二次令 $r_{\min}=0.1$，$r_{\max}$ 从 0.2 取到 1，以此类推直到第 10 次令 $r_{\min}=0.9$，$r_{\max}=1$ 时结束。由于每一次测试系数的取值次数太多，而本文只取 10 种情况中每一种情况的最优值，并用函数 $f_1(x)$，$f_2(x)$，$f_3(x)$ 对以上情况进行检验。设置参数同上，实验结果如表 2 所列。

表 2　$r_{\min}$ 和 $r_{\max}$ 的变化对函数的影响

Table 2　Effect of changes in $r_{\min}$ and $r_{\max}$ on function

$f(x)$	$f_1(x)$		$f_2(x)$		$f_3(x)$	
取值范围	M	收敛率/%	M	收敛率/%	M	收敛率/%
1	a	100	a	100	b	100
2	b	100	b	100	b	100
3	b	100	b	100	c	100
4	b	96	c	48	c	64
5	c	84	c	28	c	16
6	c	84	c	24	—	0
7	d	52	—	0	—	0
8	d	28	—	0	—	0
9	—	0	—	0	—	0
10	—	0	—	0	—	0

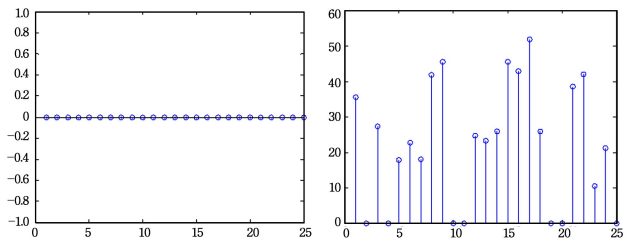
限于篇幅，本文只展示 10 维时，具有代表性的第 1 种和第 5 种情况。由于 50 维、100 维的情况与 10 维类似，此处不再赘述。图 5－图 7 是在 10 维情况下，函数 $f_1(x)$，$f_2(x)$，$f_3(x)$ 在第 1 种和第 5 种情况下的收敛图，其中横轴代表种群规模，纵轴代表适应度值。



（a）第 5 种　（b）第 1 种

图 5　$f_1(x)$ 函数的收敛图

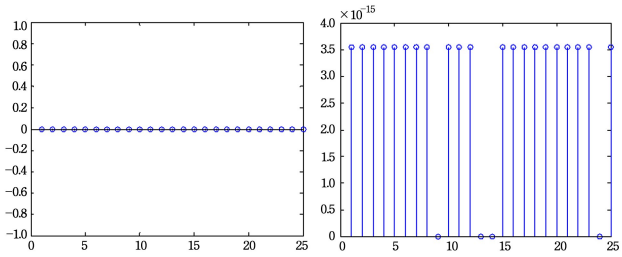
Fig. 5　Convergence diagram of $f_1(x)$ function



（a）第 5 种　（b）第 1 种

图 6　$f_2(x)$ 函数的收敛图

Fig. 6　Convergence diagram of $f_2(x)$ function



（a）第 5 种　（b）第 1 种

图 7　$f_3(x)$ 函数的收敛图

Fig. 7　Convergence diagram of $f_3(x)$ function

从表 2 中可以看出，在 10 维惯性权重系数 $r_{\min}$ 和 $r_{\max}$ 的取值为第 1 种情况时，函数 $f_1(x)$，$f_2(x)$，$f_3(x)$ 取得最优值的最小迭代次数分别在 10，10，25 代以内，收敛率为 100%，此时，函数的优化效果最好。

虽然在 10 维第 2、3 种情况下，函数 $f_1(x)$，$f_2(x)$，$f_3(x)$ 的收敛率也都达到了 100%，但是，取得最优值时的最小迭代次数比第 1 种情况大。从第 3 种情况开始，随着 $r_{\min}$ 和 $r_{\max}$ 取值范围的变）的收敛率从 100%下降为 0。这表明，在 10 维第 1 种情况下，函数 $f_1(x)$，$f_2(x)$，$f_3(x)$ 的优化效果最好。同理，在 50 维、100 维的第 1 种情况下，函数 $f_1(x)$，$f_2(x)$，$f_3(x)$ 的优化效果也是最好的。从图 5－图 7 可以看出，在第 1 种情况下，函数 $f_1(x)$，$f_2(x)$，$f_3(x)$ 取得理论最优值的收敛率为 100%，而第 5 种情况下分别为 84%，28%，16%。

综上，在 10 种情况中，第 1 种情况下函数 $f_1(x)$，$f_2(x)$，$f_3(x)$ 的优化效果最好。基于此，本文选用第 1 种情况，即将 $r_{\min}=0$，$r_{\max}=0.1$，$\beta=1$ 作为惯性权重 $w_{(t)}$ 的系数值。

图 8 给出了 $t=1:100$，$MaxNumber=100$，$\alpha_{\min}=0$，$\alpha_{\max}=0.1$，$\beta=1$ 时，惯性权重线性、指数、高斯、对数递减策略的迭代次数和惯性权重的关系曲线图。从图 8 可以看出，在迭代前期，对数递减的惯性权重曲线比线性、高斯递减的惯性权重曲线下降得更快，有利于全局探索、快速找到全局最优解的大致范围；在迭代后期，指数和高斯递减的惯性权重分别在 30 代和 40 代以后变成 0，不利于局部精细搜索，而对数递减的惯性权重曲线相比线性递减惯性权重的曲线更平缓，有利于局部精细搜索，从而找到全局最优解。

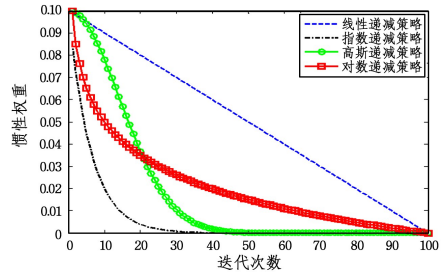


图 8　惯性权重曲线图

Fig. 8　Inertial weight graph

3.3　随机调整的发现概率策略

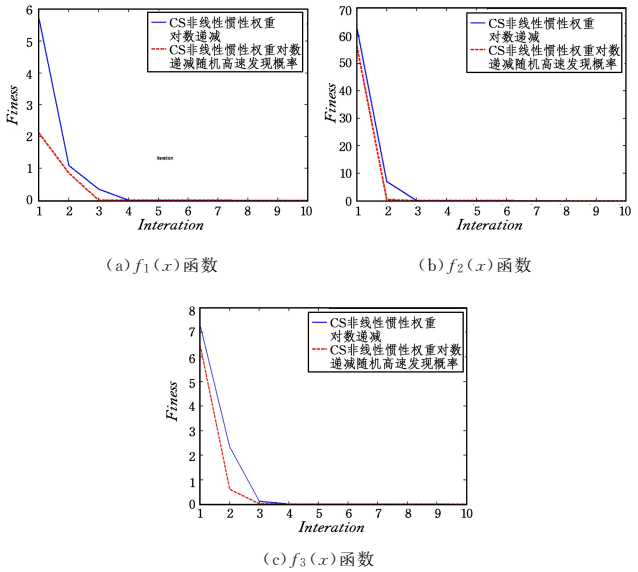
在基本的布谷鸟算法中，pa 是非常重要的参数，合适的 pa 有助于平衡算法的全局探索和局部开发能力，加快算法的

收敛速度，从而找到全局最优解。然而，基本 CS 算法的 pa 取值为固定值 0.25，该值是在初始化阶段确定的，不能改变。显然，这不利于算法全局探索和局部开发的平衡以及种群的多样性。为了克服 pa 取固定值带来的弊端，本文采用随机调整的发现概率 $r\text{-}pa$ 来取代固定值的发现概率 pa，使较大和较小的发现概率 $r\text{-}pa$ 随机出现，从而有利于平衡算法的全局探索和局部开发能力，加快算法的收敛速度，增加种群多样性。$r\text{-}pa$ 的具体表达式如下：

$$r\text{-}pa = pa_{\min} + (pa_{\max} - pa_{\min}) \times rand() - \lambda \times \tau(\alpha,\beta) \quad (9)$$

其中，$r\text{-}pa \in [0.15, 0.55]$，$pa_{\min}$ 和 $pa_{\max}$ 分别为 pa 的最大值和最小值，$rand()$ 是产生在 (0, 1) 之间均匀分布的随机数，$\tau(\alpha,\beta)$ 是服从形状参数为 α、尺度参数为 β 的伽马分布随机数，λ 用于控制惯性权重与期望值之间的偏差程度。

为了证明随机调整发现概率的有效性，我们将非线性惯性权重对数递减随机调整的发现概率与非线性惯性权重对数递减进行对比实验，并用 $f_1(x)$，$f_2(x)$，$f_3(x)$ 3 个函数进行检验。实验环境：CPU 为 Intel(R)Core(TM)i5-7500 CPU @ 3.40GHz，运行内存 8GB，操作系统为 Windows10 家庭中文版，编程环境 Matlab R2014a。其中种群规模为 25，最大迭代次数 10，算法维数 $D=10,50,100$，实验结果如图 9 所示。



(a) $f_1(x)$ 函数　(b) $f_2(x)$ 函数

(c) $f_3(x)$ 函数

图 9　函数的收敛曲线对比图

Fig. 9　Contrast of convergence curve of functions

图 9 是维数 $D=10$，$pa=0.25$，$r\text{-}pa \in [0.15, 0.55]$ 时，$f_1(x)$，$f_2(x)$，$f_3(x)$ 3 个函数的收敛曲线图。可以看出，DWCS 比非线性惯性权重对数递减的布谷鸟算法收敛速度更快且曲线更平滑，鲁棒性更强。当维数 $D=50, 100$ 时，$f_1(x)$，$f_2(x)$，$f_3(x)$ 的收敛曲线与 $D=2$ 时的结果一样，此处不再赘述。

综上，相比于非线性惯性对数递减布谷鸟算法，DWCS 在求解函数优化问题时效果更好。

3.4　DWCS 算法的流程

通过在位置更新公式中引入非线性对数递减的惯性权重，DWCS 算法在迭代前期可以快速搜寻最优解的大致范围，在迭代后期可以进行局部精细搜索，从而平衡算法全局探索和局部开发的能力；同时，将固定值发现概率 pa 改为随机调整发现概率 $r\text{-}pa$，有利于加快算法收敛速度，增加种群的多样性，使算法跳出局部极值，从而搜索到全局最优值。因此，DWCS 算法将 CS 算法位置更新公式(式(1))改为式(7)和式(8)，将发现概率 pa 改为随机调整发现概率 $r\text{-}pa$(式(9))，具体流程如下。

Step 1(初始化)　确定目标函数 $f(X)$，$X=(x_1,\cdots,x_d)^{\mathrm{T}}$，初始化群体，随机产生 n 个鸟窝的初始位置 $X_i(i=1,2,\cdots,n)$。设置算法参数：种群规模 N、维度 D、发现概率 pa、界值大小 L、最大迭代次数 $MaxNumber$、最优鸟窝位置 x_b^0，$b\in\{1,2,\cdots,n\}$ 和最优解 $f_{\min}$。

Step 2(循环体)　计算每个鸟窝的目标函数值 $f(X)$，并记录当前的最优解 $f_{\min}$，保留上代最优鸟窝位置 x_b^{t-1}，并按式(7)、式(8)更新其他鸟窝位置，得到一组新的鸟窝位置；将现有鸟窝与上一代鸟窝位置 $p_{t-1}=[x_1^{t-1},x_2^{t-1},\cdots,x_n^{t-1}]^{\mathrm{T}}$ 进行对比，用适应度值较好的鸟窝位置替换适应度值较差的鸟窝位置：$g_t=[x_1^t,x_2^t,\cdots,x_n^t]^{\mathrm{T}}$。

Step 3(循环体)　将一个随机数 R 作为鸟窝主人发现外来鸟蛋的可能性，将其与鸟被淘汰的概率 $r\text{-}pa$ 进行比较；若 $R>r\text{-}pa$，则随机改变 g_t 中的鸟窝位置，得到一组新的鸟窝位置。然后，计算全部个体的适应度值，将其与 g_t 中每个鸟窝位置的适应度值进行对比，用适应度值较好的鸟窝位置代替适应度值较差的鸟窝位置，得到一组较好的鸟窝位置：$p_t=[x_1^t,x_2^t,\cdots,x_n^t]^{\mathrm{T}}$。

Step 4　判断算法是否满足设置的最大迭代次数，若满足，则结束搜索过程，输出全局最优值 $f_{\min}$；否则，重复 Step2—Step3 进行迭代寻优。

4　仿真实验

本文选取 16 个不同难度的函数优化问题，包括 10 个多峰函数和 6 个单峰函数，在相同的环境下分别独立运行 30 次以进行仿真实验。其中 $f_1(x)-f_{11}(x)$ 为高维函数，维数 $D=10,50,100$，$f_{12}(x)-f_{16}(x)$ 为低维函数，维数 $D=2$。实验将本文方法与 CS 算法、改进的布谷鸟算法(Improved Cuckoo Search for Reliability Optimization Problems, ICS)[25]、粒子群优化算法(Particle Swarm Optimization, PSO)和蝙蝠算法(Bat-inspired Algorithm, BA)进行对比，其中 ICS 算法是布谷鸟搜索算法中的经典改进算法，且此算法也对莱维飞行和发现概率进行了改进，具有可比性；PSO 是最经典的算法之一，具有代表性；BA 算法是近几年出现的算法，具有新颖性。将本文的改进算法与以上算法进行对比，可以全面、有效地测试 DWCS 算法的寻优性能。

4.1　测试函数

(1)Griewank 函数

$$f_1(x)=1+\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1,\ x_i\in[-600,600]$$

该函数在(0,…,0)处取得最小值0。

(2)Rastrigin 函数

$$f_2(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10],x_i\in[-5.12,5.12]$$

该函数在(0,…,0)处取得最小值0。

(3)Ackley 函数

$$f_3(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+20+e,x_i\in[-32.768,32.768]$$

该函数在(0,…,0)处取得最小值0。

(4)Schaffer 函数

$$f_4(x)=\frac{\sin^2\sqrt{\sum_{i=1}^{n}x_i^2}-0.5}{[1+0.001\times\sum_{i=1}^{n}x_i^2]}+0.5,x_i\in[-10,10]$$

该函数在(0,…,0)处取得最小值0。

(5)Sphere 函数

$$f_5(x)=\sum_{i=1}^{n}x_i^2,\ x_i\in[-100,100]$$

该函数在(0,…,0)处取得最小值0。

(6)Sum Squares 函数

$$f_6(x)=\sum_{i=1}^{n}ix_i^2,x_i\in[-10,10]$$

该函数在(0,…,0)处取得最小值0。

(7)Zakharov 函数

$$f_7(x)=\sum_{i=1}^{n}x_i^2+(\frac{1}{2}\sum_{i=1}^{n}ix_i)^2+(\frac{1}{2}\sum_{i=1}^{n}ix_i)^4,x_i\in[-10,10]$$

该函数在(0,…,0)处取得最小值0。

(8)Schwefel's problem 1.2 函数

$$f_8(x)=\sum_{i=1}^{n}(\sum_{j=1}^{i}x_j)^2,\ x_i\in[-100,100]$$

该函数在(0,…,0)处取得最小值0。

(9)Schwefel's problem 2.21 函数

$$f_9(x)=\max_i\{|x_i|,1\leqslant i\leqslant n\},x_i\in[-100,100]$$

该函数在(0,…,0)处取得最小值0。

(10)Schwefel's problem 2.22 函数

$$f_{10}(x)=\sum_{i=1}^{n}|x_i|+\prod_{i=1}^{n}|x_i|,x_i\in[-10,10]$$

该函数在(0,…,0)处取得最小值0。

(11)Alpine 函数

$$f_{11}(x)=\sum_{i=1}^{n}|x_i\sin(x_i)+0.1x_i|\ ,\ x_i\in[-10,10]$$

该函数在(1,…,1)处取得最小值0。

(12)Goldstein-Price 函数

$$f_{12}(x)=[1+(x_1+x_2+1)^2\cdot(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times[30+(2x_1-3x_2^2)^2\cdot(18-32x_1+12x_1^2+48x_2^2-36x_1x_2+27x_2^2)],x_i\in[-2,2]$$

该函数在(0,-1)处取得最小值3。

(13)Branin 函数

$$f_{13}(x)=(x_2-\frac{5.1}{4\pi}x_1^2+\frac{5}{\pi}x_1-6)^2+10(1-\frac{1}{8\pi})\cos(x_i)+10,x_1\in[-5,10],x_2\in[0,15]$$

该函数在$(-\pi,12.275)$,$(\pi,2.275)$,(9.42478,2.475)处取得最小值0.397。

(14)Six-Hump Camel 函数

$$f_{14}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4,\quad x_1\in[-3,3],x_2\in[-2,2]$$

该函数在(-0.0898,0.7126),(0.0898,-0.7126)处取得最小值-1.0316。

(15)Schaffer N.2 函数

$$f_{15}(x)=0.5+\frac{\sin^2(x_1^2-x_2^2)-0.5}{[1+0.001(x_1^2+x_2^2)]^2},x_i\in[-100,100]$$

该函数在(0,…,0)处取得最小值0。

(16)Easom's 函数

$$f_{16}(x)=-\cos(x_1)\cos(x_2)\exp[-((x_1-\pi)^2+(x_1-\pi)^2)],x_i\in[-100,100]$$

该函数在$[\pi,\pi]$处取得最小值-1。

4.2　测试环境和算法参数的确定

实验环境：CPU 为 AMD A6-5350M 2.90GHz，运行内存8GB，操作系统为 Windows7，编程环境 Matlab R2014a。表3中 CS，BA，PSO，ICS，DWCS 5种算法的种群规模为25，迭代次数为1000，算法维数为2，10，50，100，其他参数如表3所列。

表3　各算法参数设置

Table 3　Parameter settings of each algorithm

算法名称	参数设置
CS	$n=25,T=1000,pa=0.25,D=2,10,50,100$
BA	$A=0.25,r=0.5$
PSO	$c_1=c_2=2,w\in[0.4,0.9)$
ICS	$pa_{\min}=0.0005,pa_{\max}=1,\alpha_{\min}=0.05,\alpha_{\max}=0.5$
DWCS	$pa\in[0.15,0.55],w(t)\in(0,0.1]$

4.3　算法比较

基于4.2节对算法参数的设置，本节运用表3中的5种算法对16种不同难度的函数问题在相同的环境下分别独立运行30次以进行仿真实验，表4－表14给出了在不同维数下5种算法对函数 $f_1(x)-f_{16}(x)$ 的求解精度，其中加粗数据代表精度较好的解。

表4　$f_1(x)$：Griewank 函数的仿真结果

Table 4　Simulation results of Griewank function

维数	算法	最优解	最差解	平均值	标准差
10	BA	1.77×10	1.52×10^2	5.65×10	3.26×10
	CS	1.94×10^{-2}	7.03×10^{-2}	4.23×10^{-2}	1.33×10^{-2}
	PSO	1.11×10^{-1}	1.24	6.08	2.50
	ICS	2.59×10^{-4}	6.34×10^{-2}	2.89×10^{-2}	1.58×10^{-2}
	DWCS	**0.00**	**0.00**	**0.00**	**0.00**
50	BA	2.08×10^2	6.19×10^2	3.76×10^2	1.05×10^2
	CS	5.91×10^{-1}	9.98×10^{-1}	8.45×10^{-1}	1.13×10^{-1}
	PSO	1.67×10^{-1}	4.81×10^{-1}	3.27×10^{-1}	9.03×10^{-2}
	ICS	7.70×10^{-3}	1.27×10^{-1}	3.52×10^{-2}	2.62×10^{-2}
	DWCS	**0.00**	**0.00**	**0.00**	**0.00**
100	BA	4.22×10^2	1.57×10^3	7.66×10^2	2.77×10^2
	CS	2.95	6.02	4.34	6.99×10^{-1}
	PSO	8.59×10^{-1}	3.81	1.54	7.28×10^{-1}
	ICS	1.24	1.61	1.42	8.60×10^{-2}
	DWCS	**0.00**	**0.00**	**0.00**	**0.00**

表 5 $f_2(x)$:Rastrigin 函数和 $f_3(x)$:Ackley 函数的仿真结果

Table 5 Simulation results of Rastrigin function and Ackley function

维数		$f_2(x)$				$f_3(x)$			
	算法	最优解	最差解	平均值	标准差	最优解	最差解	平均值	标准差
10	BA	3.98	88.26×10	3.64×10	1.96×10	1.12×10	1.85×10	1.54×10	1.83
	CS	1.94	11.07×10	6.26	1.94	2.89×10^{-7}	3.82×10^{-5}	8.93×10^{-6}	8.84×10^{-6}
	PSO	9.97×10^{-1}	3.03×10	6.22	7.16	1.10×10^{-3}	1.16	9.07×10^{-2}	2.90×10^{-1}
	ICS	2.86×10^{-4}	3.89	1.78	9.58×10^{-1}	3.56×10^{-15}	2.60×10^{-13}	4.97×10^{-14}	5.89×10^{-14}
	DWCS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	BA	9.35×10	2.77×10^{2}	1.75×10^{2}	4.80×10	1.57×10	1.87×10	1.71×10	7.99×10^{-1}
	CS	1.16×10^{2}	1.90×10^{2}	1.53×10^{2}	2.28×10	2.71	5.08	3.77	6.23×10^{-1}
	PSO	7.96×10	2.29×10^{2}	1.62×10^{2}	3.99×10	2.41	4.24	3.33	5.15
	ICS	2.02×10^{2}	3.30×10^{2}	2.69×10^{2}	3.17×10	5.32×10^{-2}	2.16×10^{-1}	1.03×10^{-1}	3.72×10^{-2}
	DWCS	0.00	0.00	0.00	0.000.00	0.00	0.00	0.00	
100	BA	1.81×10^{2}	4.66×10^{2}	2.80	6.07×10	1.53×10	1.81×10	1.70×10	6.50×10^{-1}
	CS	3.34×10^{2}	5.14×10^{2}	4.20×10^{2}	4.24×10	6.07	1.01×10	7.38	8.98
	PSO	3.60×10^{2}	6.41×10^{2}	5.04×10^{2}	6.90×10	3.57	6.48	4.70	7.03×10^{-1}
	ICS	6.66×10^{2}	8.15×10^{2}	7.47×10^{2}	3.79×10	3.60	5.04	4.18	3.22
	DWCS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

表 6 $f_4(x)$:Schaffer 函数和 $f_5(x)$:Sphere 函数的仿真结果

Table 6 Simulation results of Schaffer function and Sphere function

维数		$f_4(x)$				$f_5(x)$			
	算法	最优解	最差解	平均值	标准差	最优解	最差解	平均值	标准差
10	BA	3.72×10^{-2}	1.78×10^{-1}	1.25×10^{-1}	3.85×10^{-2}	1.94×10^{3}	1.19×10^{4}	6.21×10^{3}	2.72×10^{3}
	CS	9.70×10^{-3}	1.39×10^{-2}	9.90×10^{-3}	7.59×10^{-4}	3.63×10^{-16}	1.50×10^{-14}	5.60×10^{-15}	4.13×10^{-15}
	PSO	9.70×10^{-3}	3.72×10^{-2}	1.71×10^{-2}	1.24×10^{-2}	1.66×10^{-8}	4.75×10^{-4}	5.86×10^{-5}	9.77×10^{-5}
	ICS	9.70×10^{-3}	3.72×10^{-2}	1.17×10^{-2}	7.00×10^{-3}	7.31×10^{-32}	6.37×10^{-29}	1.22×10^{-29}	1.53×10^{-29}
	DWCS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	BA	2.28×10^{-1}	4.30×10^{-1}	3.43×10^{-1}	6.04×10^{-2}	1.75×10^{4}	7.30×10^{4}	4.06×10^{4}	1.22×10^{4}
	CS	7.82×10^{-2}	1.27×10^{-1}	1.21×10^{-1}	1.69×10^{-2}	6.68×10^{-1}	4.27	2.16	1.01
	PSO	3.72×10^{-2}	1.27×10^{-1}	8.55×10^{-2}	2.63×10^{-2}	1.77	5.49	3.28	9.45
	ICS	7.82×10^{-2}	1.78×10^{-1}	1.27×10^{-1}	2.27×10^{-2}	2.50×10^{-3}	1.86×10^{-2}	7.10×10^{-3}	4.00×10^{-3}
	DWCS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	BA	3.46×10^{-1}	4.72×10^{-1}	4.13×10^{-1}	3.76×10^{-2}	4.47×10^{4}	1.69×10^{5}	8.07×10^{4}	2.90×10^{4}
	CS	1.78×10^{-1}	2.28×10^{-1}	2.00×10^{-1}	2.46×10^{-2}	2.26×10^{2}	5.24×10^{2}	3.81×10^{2}	8.74×10
	PSO	7.82×10^{-2}	2.28×10^{-1}	1.59×10^{-1}	3.38×10^{-2}	2.33×10	7.90×10	3.79×10	1.07×10
	ICS	1.79×10^{-1}	2.98×10^{-1}	2.43×10^{-1}	2.55×10^{-2}	2.47×10	6.88×10	4.83×10	1.15×10
	DWCS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

表 7 $f_6(x)$:Sum Squares 函数和 $f_7(x)$:Zakharov 函数的仿真结果

Table 7 Simulation results of Zakharov function and Sum Squares function

维数		$f_6(x)$				$f_7(x)$			
	算法	最优解	最差解	平均值	标准差	最优解	最差解	平均值	标准差
10	BA	7.88×10^{-6}	1.16×10^{2}	3.28×10	3.10×10	1.52×10	2.50×10^{3}	1.82×10^{2}	4.51×10^{2}
	CS	1.91×10^{-17}	1.43×10^{-15}	3.08×10^{-16}	3.41×10^{-16}	9×10^{-9}	4.47×10^{-8}	9.29×10^{-9}	7.89×10^{-9}
	PSO	4.83×10^{-7}	6.40×10^{-3}	7.59×10^{-4}	1.40×10^{-3}	4.88×10^{-4}	5.17×10^{-2}	1.46×10^{-2}	1.36×10^{-2}
	ICS	2.41×10^{-32}	6.38×10^{-31}	1.72×10^{-31}	1.68×10^{-31}	1.55×10^{-11}	7.61×10^{-10}	1.93×10^{-10}	1.73×10^{-10}
	DWCS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	BA	2.32×10^{3}	1.59×10^{4}	6.01×10^{3}	2.95×10^{3}	5.26×10	3.92×10^{4}	4.65×10^{3}	8.59×10^{3}
	CS	1.16×10^{-1}	5.70×10^{-1}	3.76×10^{-1}	1.22×10^{-1}	2.60×10^{2}	5.66×10^{2}	4.41×10^{2}	7.78×10
	PSO	9.81	2.23×10^{3}	1.12×10^{2}	4.02×10^{2}	1.17×10^{2}	6.83×10^{2}	3.31×10^{2}	1.43×10^{2}
	ICS	3.61×10^{-4}	2.80×10^{-3}	1.20×10^{-3}	5.08×10^{-4}	1.64×10^{2}	4.49×10^{2}	2.88×10^{2}	6.91×10
	DWCS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	BA	1.39×10^{4}	6.11×10^{4}	2.79×10^{4}	1.07×10^{4}	1.24×10^{3}	3.37×10^{4}	6.16×10^{3}	8.32×10^{3}
	CS	9.13×10	2.95×10^{2}	1.50×10^{2}	4.67×10	1.12×10^{3}	2.25×10^{3}	1.67×10^{3}	3.43×10^{2}
	PSO	1.38×10^{2}	7.52×10^{2}	3.82×10^{2}	1.23×10^{2}	5.97×10^{2}	2.32×10^{3}	1.28×10^{3}	4.15×10^{2}
	ICS	1.06×10	2.57×10	1.57×10	3.55	7.74×10^{2}	1.74×10^{3}	1.34×10^{3}	2.38×10^{2}
	DWCS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

表 8　$f_8(x)$：Schwefel's problem 1.2 函数和 $f_9(x)$：Schwefel's problem 2.21 函数的仿真结果

Table 8　Simulation results of Schwefel's probiem 1.2 function and Schwefel's problem 2.21function

维数		$f_8(x)$				$f_9(x)$			
	算法	最优解	最差解	平均值	标准差	最优解	最差解	平均值	标准差
10	BA	1.13×10^{3}	2.01×10^{4}	8.33×10^{3}	4.91×10^{3}	2.73×10	5.60×10	3.85×10	7.15
	CS	2.93×10^{-8}	5.05×10^{-7}	2.05×10^{-7}	1.05×10^{-7}	5.16×10^{-5}	4.26×10^{-4}	1.92×10^{-4}	1.06×10^{-4}
	PSO	1.30×10^{-3}	3.23×10^{-1}	3.32×10^{-2}	6.07×10^{-2}	8.30×10^{-3}	1.44×10^{-1}	5.05×10^{-2}	3.43×10^{-2}
	ICS	2.25×10^{-9}	8.28×10^{-8}	1.83×10^{-8}	1.98×10^{-8}	2.56×10^{-7}	2.41×10^{-6}	7.41×10^{-7}	5.23×10^{-7}
	DWCS	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
50	BA	3.33×10^{4}	3.50×10^{5}	1.80×10^{5}	9.33×10^{4}	4.33×10	7.86×10	6.14×10	8.90
	CS	1.70×10^{3}	4.96×10^{3}	3.00×10^{3}	8.16×10^{2}	1.15×10	2.48×10	1.70×10	3.51
	PSO	1.15×10^{3}	7.21×10^{3}	3.07×10^{3}	1.34×10^{3}	2.76	1.41×10	7.22	3.22
	ICS	3.10×10^{3}	9.97×10^{3}	5.56×10^{3}	1.40×10^{3}	6.03	1.65×10	1.04×10	2.35×10
	DWCS	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
100	BA	1.22×10^{5}	1.47×10^{6}	7.43×10^{5}	3.14×10^{5}	5.07×10	7.86×10	6.36×10	7.31
	CS	1.93×10^{4}	3.99×10^{4}	2.66×10^{4}	4.81×10^{3}	2.13×10	3.50×10	2.60×10	2.92
	PSO	1.16×10^{4}	5.14×10^{4}	3.11×10^{4}	1.03×10^{4}	9.11	2.98×10	1.56×10	4.71
	ICS	2.44×10^{4}	6.01×10^{4}	3.76×10^{4}	8.67×10^{3}	1.87×10	3.20×10	2.56×10	3.43
	DWCS	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**

表 9　$f_{10}(x)$：Schwefel's problem 2.22 函数和 $f_{11}(x)$：Alpine 函数的仿真结果

Table 9　Simulation results of Schwefel's problem 2.22 function and Alpine function

维数		$f_8(x)$				$f_9(x)$			
	算法	最优解	最差解	平均值	标准差	最优解	最差解	平均值	标准差
10	BA	3.93×10^{-1}	4.71×10	1.87×10	1.15×10	2.42×10^{-1}	7.48	3.27	1.61
	CS	1.09×10^{-7}	9.31×10^{-7}	3.37×10^{-7}	1.76×10^{-7}	7.78×10^{-2}	3.27×10^{-1}	1.91×10^{-1}	6.10×10^{-2}
	PSO	9.80×10^{-3}	2.25×10^{-1}	6.62×10^{-2}	5.62×10^{-2}	1.20×10^{-3}	1.59×10^{-1}	2.31×10^{-2}	3.06×10^{-2}
	ICS	4.66×10^{-18}	1.14×10^{-16}	2.87×10^{-17}	2.38×10^{-17}	3.42×10^{-4}	5.76×10^{-2}	8.70×10^{-3}	1.32×10^{-2}
	DWCS	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
50	BA	5.44×10	1.49×10^{17}	5.03×10^{15}	2.71×10^{16}	1.98×10	5.39×10	3.41×10	8.07
	CS	5.38×10^{-1}	4.27	1.22	6.78×10^{-1}	1.18×10	2.46×10	1.70×10	3.06
	PSO	4.57	3.77×10	1.16×10	8.33	2.23	1.27×10	7.44	2.80×10
	ICS	1.32×10^{-2}	3.85×10^{-2}	2.45×10^{-2}	5.90×10^{-3}	2.53×10	3.78×10	3.14×10	3.82
	DWCS	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
100	BA	1.11×10^{2}	2.50×10^{31}	8.38×10^{29}	4.56×10^{30}	5.28×10	1.10×10^{2}	7.97×10	1.61×10
	CS	1.01×10	1.79×10	1.38×10	2.17	3.45×10	6.05×10	4.31×10	6.67
	PSO	1.50×10	1.23×10^{2}	3.65×10	2.08×10	1.16×10	5.07×10	2.71×10	8.77
	ICS	7.06	3.21×10	1.13×10	4.86	5.67×10	8.91×10	7.57×10	9.54×10
	DWCS	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**

表 10　$f_{12}(x)$：Goldstein-Price 函数的仿真结果

Table 10　Simulation results of Goldstein-Price function

维数	算法	最优解	最差解	平均值	标准差
2	BA	3.00	8.40×10	5.70×10	1.48×10
	CS	3.00	3.00	3.00	2.09×10^{-15}
	PSO	3.00	3.00	3.00	1.24×10^{-15}
	ICS	3.00	3.00	3.00	1.23×10^{-15}
	DWCS	**3.00**	**3.00**	**3.00**	$\mathbf{1.20\times10^{-15}}$

表 11　$f_{13}(x)$：Branin 函数的仿真结果

Table 11　Simulation results of Branin function

维数	算法	最优解	最差解	平均值	标准差
2	BA	0.3979	0.399	0.3979	2.49×10^{-10}
	CS	0.3979	0.3979	0.3979	0.00
	PSO	0.3979	0.3979	0.3979	0.00
	ICS	0.3979	0.3979	0.3979	0.00
	DWCS	**0.3979**	**0.3979**	**0.3979**	**0.00**

表 12　$f_{14}(x)$：Six-Hump Camel 函数的仿真结果

Table 12　Simulation results of Six-Hump Camel function

维数	算法	最优解	最差解	平均值	标准差
2	BA	−1.0316	−0.2155	−1.0044	1.49×10^{-1}
	CS	−1.0316	−1.0316	−1.0316	6.71×10^{-16}
	PSO	−1.0316	−1.0316	−1.0316	6.52×10^{-16}
	ICS	−1.0316	−1.0316	−1.0316	6.78×10^{-16}
	DWCS	**−1.0316**	**−1.0316**	**−1.0316**	$\mathbf{5.83\times10^{-16}}$

表 13　$f_{15}(x)$Schaffer N.2 函数的仿真结果

Table 13　Simulation results of Schaffer N.2 function

维数	算法	最优解	最差解	平均值	标准差
2	BA	2.68×10^{-13}	4.25×10^{-1}	1.25×10^{-1}	1.09×10^{-1}
	CS	0.00	0.00	0.00	0.00
	PSO	0.00	0.00	0.00	0.00
	ICS	0.00	0.00	0.00	0.00
	DWCS	**0.00**	**0.00**	**0.00**	**0.00**

表 14 $f_{16}(x)$:Easom's 函数的仿真结果

Table 14 Simulation results of Easom's function

维数	算法	最优解	最差解	平均值	标准差
2	BA	−1	0.00	-2.33×10^{-1}	4.30×10^{-1}
	CS	−1	−1	−1	0.00
	PSO	−1	−1	−1	0.00
	ICS	−1	−1	−1	0.00
	DWCS	**−1**	**−1**	**−1**	**0.00**

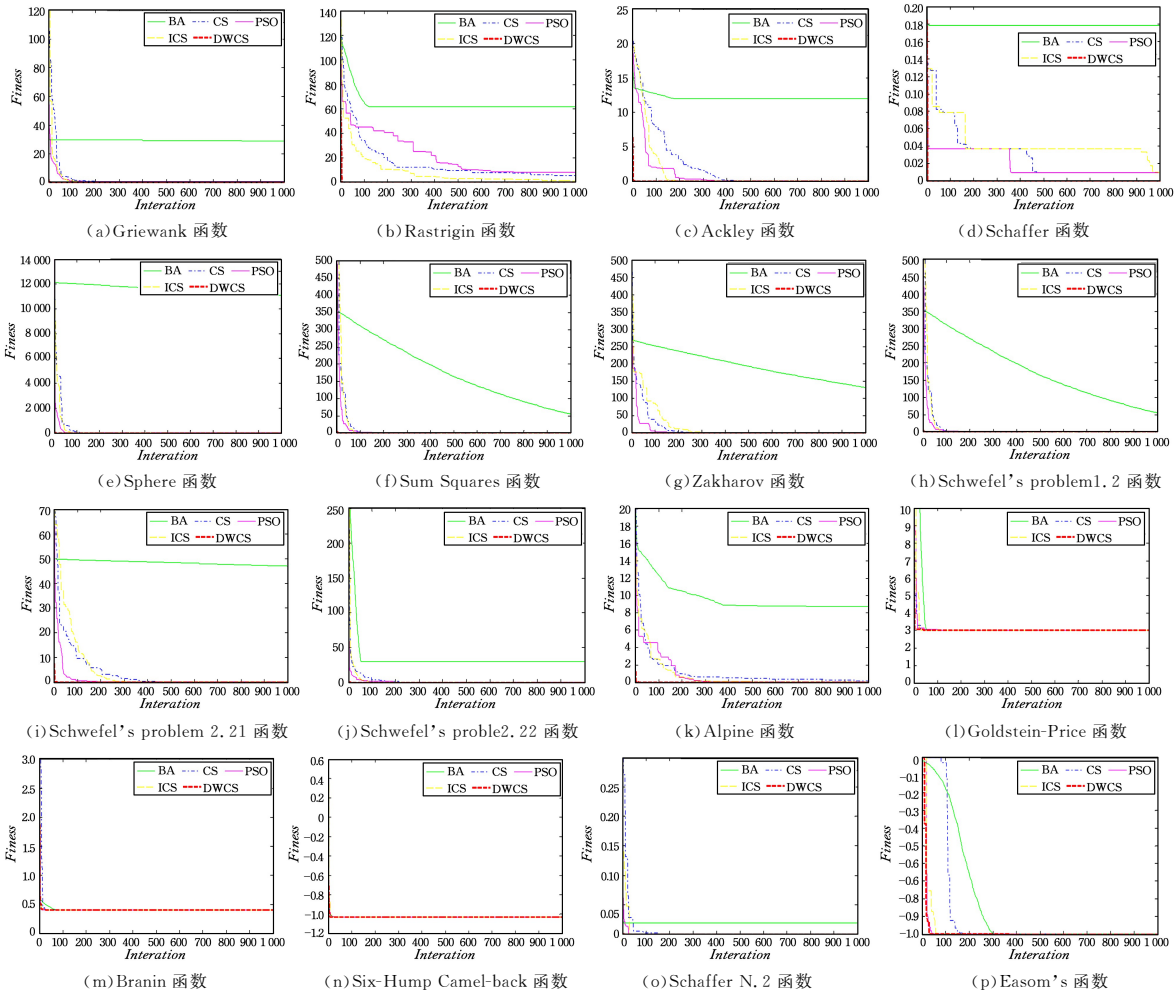
从表 4—表 14 可以看出,在 2 维、10 维、50 维、100 维 4 种情况下,与其他 4 种算法相比,DWCS 算法显著提高了解的质量,具有很高的寻优精度,且无论对于单峰函数 $f_5(x)$, $f_6(x)$, $f_8(x)$, $f_9(x)$, $f_{10}(x)$, $f_{16}(x)$,还是对于拥有众多局部极值点的多峰函数 $f_1(x)$, $f_2(x)$, $f_3(x)$, $f_4(x)$, $f_7(x)$, $f_{11}(x)$, $f_{12}(x)$, $f_{13}(x)$, $f_{14}(x)$, $f_{15}(x)$,DWCS 算法均可以取得全局最优解,而且随着维数的变化算法仍呈现较好的搜索能力和较稳定的收敛性能,这表明改进后的算法求解精度高、鲁棒性强。而其他 4 种算法在 10 维、50 维、100 维 3 种情况下,均无法求得函数 $f_1(x)-f_{11}(x)$的全局最优解。在 10 维情况下,对于函数 $f_5(x)$和 $f_6(x)$,ICS 与 BA,PSO,CS 相比,求解精度分别提高了 32,24,14 个数量级以上;对于函数 $f_3(x)$, $f_7(x)$, $f_8(x)$, $f_{10}(x)$,ICS 与 BA,PSO,CS 相比;求解精度分别提高了 10,6,1 个数量级以上;对于函数 $f_1(x)$, $f_2(x)$, $f_4(x)$,ICS 与 BA,PSO,CS 相比数量级相当。

从以上分析可知,对于高维函数,4 种算法的搜索性能从优到劣依次是 ICS,CS,PSO,BA。当维数从 10 维上升到 50 维或 100 维时,实验结果相差不大,不同的是随着维数的变化,PSO 的收敛精度与 ICS,CS,BA 3 种算法相比变化较小,这表明 PSO 算法有较稳定的收敛性能。对于二维函数 $f_{12}(x)-f_{16}(x)$,CS,PSO,ICS 算法都能求得全局最优解,且标准差较小,而 BA 只能求得函数 $f_{12}(x)$, $f_{13}(x)$, $f_{14}(x)$和 $f_{16}(x)$的全局最优解,且标准差较大。以上结果表明,对于低维函数,4 种算法的搜索性能从优到劣依次是 ICS,CS,PSO,BA。当函数 $f_1(x)-f_{11}(x)$的维数从 10 维上升到 50 维或 100 维时,BA,CS,ICS 算法的收敛精度明显下降,PSO 算法随着维数的升高表现出较稳定的收敛性能,而 DWCS 算法随着维数的变化仍然呈现出较好的搜索能力和稳定的收敛性能。以上表明,在 BA, CS, PSO, ICS, DWCS 5 种算法中,DWCS 算法的鲁棒性最强。

综上,维数为 2,10,50,100 时,对 16 个函数优化问题进行求解的实验结果表明,与 BA, CS, PSO, ICS 算法相比,DWCS 算法的性能最好,无论是求解低维、高维,还是单峰、多峰函数优化问题,该算法求解精度高、鲁棒性强,特别对于 100 维高维函数 $f_1(x)-f_{11}(x)$,其能收敛到全局最优解。

4.4 收敛曲线分析

收敛曲线可以直观地呈现算法的收敛速度和收敛精度,因此本文给出了 5 种算法在 16 个函数下的收敛曲线对比图,如图 10 所示。



(a)Griewank 函数 (b)Rastrigin 函数 (c)Ackley 函数 (d)Schaffer 函数

(e)Sphere 函数 (f)Sum Squares 函数 (g)Zakharov 函数 (h)Schwefel's problem1.2 函数

(i)Schwefel's problem 2.21 函数 (j)Schwefel's proble2.22 函数 (k)Alpine 函数 (l)Goldstein-Price 函数

(m)Branin 函数 (n)Six-Hump Camel-back 函数 (o)Schaffer N.2 函数 (p)Easom's 函数

图 10 函数的收敛曲线

Fig. 10 Convergence curve of functions

图 10 中，$f_5(x)$，$f_6(x)$，$f_8(x)$，$f_9(x)$，$f_{10}(x)$，$f_{16}(x)$为单峰函数，常用于检测算法的收敛速度；$f_1(x)$，$f_2(x)$，$f_3(x)$，$f_4(x)$，$f_7(x)$，$f_{11}(x)$，$f_{12}(x)$，$f_{13}(x)$，$f_{14}(x)$，$f_{15}(x)$，为多峰函数，存在较多局部极小值，可以有效检测算法的全局收敛能力。图 10(a)－图 10(h)中的函数为高维函数，维数 $D=10$，图 10(i)－图 10(p)中的函数为低维函数，维数 $D=2$。

为了直观地展现 DWCS 算法的收敛速度优于 BA，CS，PSO，ICS 算法，图 10 展示了这 5 种算法在 16 个函数下的收敛过程，从图中可以看出，DWCS 算法明显有较好的收敛性能，与其他算法相比，DWCS 算法的收敛曲线前期下降得最快，且几乎在 100 代之内可以收敛到全局最优解，这表明 DWCS 算法有较快的收敛速度和较好的求解能力，从而验证了表 4－表 14 的结果。

从图 10 中函数的收敛曲线也可以看出，在 BA，CS，PSO，ICS 4 种算法中，ICS 算法相比其他算法在前期有较快的收敛速度，在后期有较好的寻优精度；而余下的 3 种算法中 CS 表现较好，PSO 次之，BA 最差。在 16 幅收敛曲线图中，BA 的收敛速度慢，寻优精度低，且易陷入局部最优，找到全局最优解的次数也最少。

综上所述，对这 5 种算法在 16 个函数下的收敛曲线进行分析。与其他 4 种算法相比，DWCS 算法明显有较好的收敛性能，无论是单峰、多峰，还是低维、高维函数，DWCS 算法在收敛速度和求解精度上都有很大的提升。

结束语　为求解复杂函数优化问题，将非线性惯性权重对数递减和随机调整发现概率引入 CS 算法中，并对对数递减参数和随机调整发现概率 pa 进行了大量测试，选取 $r_{\min}=0$，$r_{\max}=0.1$，$\beta=1$ 作为对数递减的最佳参数组合，将 $pa\in[0.15,0.55]$作为随机调整发现概率的最佳取值范围，此时，函数的优化效果最好。本文设计了一种随进化迭代次数非线性递减的惯性权重来改进鸟巢位置的更新方式，协调布谷鸟算法的探索和开发能力。同时，引入随机调整的发现概率 r-pa 代替固定值 pa，使较大和较小的发现概率 r-pa 随机出现，从而有利于平衡算法的全局探索和局部开发能力，加快算法收敛速度，增加种群多样性。对于 16 个函数优化问题，在 100 维的高维求解难度下，改进的布谷鸟算法均能得到全局最优解，与 BA，CS，PSO，ICS 算法相比，其寻优精度大幅提高，收敛速度更快，迭代次数更少，鲁棒性更强。在 16 个测试函数中，本文改进的算法均能收敛到全局最优解，证明了 DWCS 算法的有效性和可行性。

参 考 文 献

[1] WALTON S，HASSAN O，MORGAN K，et al. Modified cuckoo search：a new gradient free optimization algorithm[J]. Chaos，Solitons and Fractals，2011，44(9)：710-718.

[2] VALIAN E，MOHANNA S，TAVAKOLI S. Improved cuckoo search algorithm for global optimization[J]. International Journal of Communications and Information Technology，2011，1(1)：31-44.

[3] LI X T，WANG J A，YIN M H. Enhancing the performance of cuckoo search algorithm using orthogonal learning method[J]. Neural Computing and Applications，2014，24(6)：1233-1247.

[4] LONG W，CAI S H，JIAO J J，et al. Improved whale optimization algorithm for large scale optimization problems [J]. Systems Engineering－Theory & Practice，2017，37(11)：2983-2994.

[5] GOLDBERG D E，HOLLAND J H. Genetic algorithm in search，optimization and machine learning[M]. Boston：Addison-Wesley Longman Publishing Co. Inc. 1989.

[6] EBERHART R，KENNEDY J. A new optimizer using particleswarm theory [C]// Proceedings of the Sixth International Symposium on Micro Machine and Human Science. Nagoya：IEEE，1995：39-43.

[7] KIRKPATRICK S，GELATT C D，VECCHI M P. Optimization by simulated annealing [J]. Science，1983，220：671-680.

[8] KOUDIL M，BENATCHBA K，TARABET A，et al. Using artificial bees to solve partitioning and scheduling problems in code sign [J]. Applied Mathematics and Computation，2007，186(2)：1710-1722.

[9] GEEM Z W，KIM J H，LOGANATHAN G V. A new heuristic optimization algorithm：Harmony search[J]. Simulation，2001，76(2)：60-68.

[10] PAN W T. A new Fruit Fly Optimization Algorithm：Taking the financial distress model as an example [J]. Knowledge-Based Systems (0950-7051)，2012，26(2)：69-74.

[11] KRISHNANAND K N，GHOSE D. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics[C]// Proceedings of IEEE Swarm Intelligence Symposium. Pasadena：IEEE，2005：84-91.

[12] YANG X S，DEB S. Cuckoo search via Levy flight [C]// Proceedings of World Congress on Nature & Bio-logically Inspired Computing. Coimbatore：IEEE，2009：210-214.

[13] YANG X S，DEB S. Engineering optimization by cuckoo search [J]. International Journal of Mathematical Modeling and Numerical Optimization (2040-3607)，2010，1(4)：330-343.

[14] LI Y，MA L. A new metaheuristic cuckoo search algorithm [J]. Systems Engineering，2012，30(8)：64-69.

[15] LI Y，PEI Y H，LIU J S. Bat optimal algorithm combined uniform mutation with gaussian mutation[J]. Control and Decision，2017，32(10)：1775-1781.

[16] OUAARAB A，AHIOD B，YANG X S. Discrete cuckoo search algorithm for the travelling salesman problem[J]. Neural Computing and Applications，2014，24(7/8)：1659-1669.

[17] NAIK M K，PANDA R. A novel adaptive cuckoo search algorithm for intrinsic discriminant analysis based face recognition [J]. Applied Soft Computing (1568-4946)，2016，38(C)：661-675.

[18] SETHI R，PANDA S，SAHOO B P. Cuckoo search algorithm based optimal tuning of PID structured TCSC controller[M]. Odisha：Springer，2015：251-263.

[19] WANG J，ZHOU B. A hybrid adaptive cuckoo search optimization algorithm for the problem of chaotic systems parameter estimation [J]. Neural Computing & Applications (0941-0643)，2016，27(6)：1511-1517.

[20] WANG L J, YIN Y L, ZHONG Y W. Cuckoo search algorithm with dimension by dimension improvement[J]. Journal of Software, 2013, 24(11): 2687-2698.

[21] MA W, SUN Z X. A global cuckoo optimization algorithm using coarse-to-fine search[J]. Acta Electronica Sinica, 2015, 43(12): 2429-2439.

[22] VALIA E, TAVAKOLI S, MOHANNA S. Improved cuckoo search for reliability optimization problems [J]. Computers & Industrial Engineering (0360-8352), 2013, 64(1): 459-468.

[23] ZHENG H Q, ZHOU Y Q. A novel cuckoo search optimization algorithm based on Gauss distribution[J]. Journal of Computational Information Systems, 2012, 8(10): 4193-4200.

[24] LI R Y, DAI R W. Adaptive step-size cuckoo search algorithm [J]. Computer Science, 2017, 44(5): 235-240.

[25] WANG L J, YIN Y L, ZHONG Y W. Cuckoo search with varied scaling factor[J]. Frontiers of Computer Science, 2015, 9(4): 623-635.

[26] JIN Q B, QI L F. Novel improved cuckoo search for PID controller design [J]. Transactions of the Institute of Measurement & Control, 2014, 37(6): 1-11.

[27] PRAJAPATI P P, SHAH M V. Performance Estimation of Differential Evolution, Particle Swarm Optimization and Cuckoo Search Algorithms [J]. I. J. Intelligent Systems and Applications, 2018, 6: 59-67.

[28] ZHANG Z C, HAN W, MAO B. Adaptive discrete cuckoo algorithm based on simulated annealing for solving TSP [J]. Acta Electronica Sinica, 2018, 46(8): 1849-1857.

[29] ZHANG M Q, WANG H, CUI Z H, et al. Hybrid multiob-jective cuckoo search with dynamical local search [J]. Memetic Computing, 2017, 10(4): 1-10.

[30] FU W Y. Equilibrium single evolution based cuckoo search algorithm [J]. Acta Electronica Sinica, 2019, 47(2): 282-288.

[31] MARELI M, TWALA B. An adaptive cuckoo search algorithm for optimization[J]. Applied Computing and Informatics, 2018, 14(2): 107-115.

[32] SALGOTRA R, SINGH U, SAHA S. New cuckoo search algorithms with enhanced exploration and exploitation properties [J]. Expert Systems With Applications, 2018, 95: 384-420.

[33] WANG Z, JIA C X, SUN Y H. Parasitized breeding and nestlings growth in oriental cuckoo[J]. Chinese Journal of Zoology, 2004, 39(1): 103-105.

[34] VISWANATHAN G M, AFANASYEV V, BULDYRE-V S V, et al. Lévy flights in random searches [J]. Physica A: Statistical Mechanics and its Applications, 2000, 282(1/2): 1-12.

[35] SHI Y H, EBERHART R C. Empirical study of particle swarm optimization [C]// Proceedings of the 1999 Congress on Evolutionary Computation. Washington: IEEE, 1999, 3: 1945-1949.

[36] SHI Y H, EBERHART R C. Fuzzy adaptive particle s-warm optimization [C]// Proceedings the 2001 Congress on Evolutionary Computation. Seoul: IEEE, 2001: 101-106.

[37] EBERHART R C, SHI Y H. Tracking and optimizing dynamic systems with particle swarms [C]// Proceedings of the 2001 Congress on Evolutionary Computation. Seoul: IEEE, 2001: 94-100.

[38] PERAM T, VEERAMACHANENI K, MOHAN C K. -Fitness-distance-ratio based particle swarm optimization [C]// Proceedings of the 2003 IEEE Swarm Intelligence Symposium. Indianapolis: IEEE, 2003: 174-181.

[39] FENG J M, LIU S Y. Particle swarm optimization algorithm based on inertia weight exponentially decreasing for solving absolute value equations [J]. Journal of Jilin University (Science Edition), 2016, 54(6): 1265-1269.

[40] ZHANG X, WANG P, XING J C, et al. Particle swarm optimization algorithms with decreasing inertia weight based on gaussian function[J]. Application Research of Computers, 2012, 29(10): 3710-3712, 3724.

[41] DAI W Z, YANG X L. Particle swarm optimization algorithm based on inertia weight logarithmic decreasing. Computer Engineering and Applications, 2015, 51(17): 14-19, 52.

LI Yu, born in 1969, Ph. D, professor. Her main research interests include intelligent algorithm, logistics management and e-commerce.