采用搜索趋化策略的布谷鸟全局优化算法

马 卫[1,2],孙正兴[1]
(1.南京大学计算机软件新技术国家重点实验室,江苏南京 210093; 2.南京旅游职业学院酒店管理学院,江苏南京 211100)

摘 要: 布谷鸟搜索算法是一种基于莱维飞行搜索策略的新型智能优化算法.单一的莱维飞行随机搜索更新策略存在全局搜索性能不足和寻优精度不高等缺陷.为了解决这一问题,本文提出了一种改进的布谷鸟全局优化算法.该算法的主要特点在于以下三个方面:首先,采用全局探测和模式移动交替进行的模式搜索趋化策略,实现了布谷鸟莱维飞行的全局探测与模式搜索的局部优化的有机结合,从而避免盲目搜索,加强算法的局部开采能力;其次,采取自适应竞争机制动态选择最优解数量,实现了迭代过程搜索速度和解的多样性间的有效平衡;最后,采用优势集搜索机制,实现了最优解的有效合作分享,强化了优势经验的学习.对 52 个典型测试函数实验结果表明,本文算法不仅寻优精度和寻优率显著提高,鲁棒性强,且适合于多峰及复杂高维空间全局优化问题.本文算法与最新提出的改进的布谷鸟优化算法以及其它智能优化策略相比,其全局搜索性能与寻优精度更具优势,效果更好.

关键词: 布谷鸟算法; 趋化搜索; Hooke-Jeeves 模式搜索; 合作分享; 自适应竞争; 全局优化
中图分类号: TP18 **文献标识码:** A **文章编号:** 0372-2112 (2015)12-2429-11
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2015.12.013

A Global Cuckoo Optimization Algorithm Using Coarse-to-Fine Search

MA Wei[1,2], SUN Zheng-xing[1]
(1. *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu* 210093, *China*;
2. *School of Hotel Management, Nanjing Institute of Tourism and Hospitality, Nanjing, Jiangsu* 211100, *China*)

Abstract: Cuckoo search (CS) algorithm is a meta-heuristic optimization algorithm based on Lévy flights. We propose an improved cuckoo search algorithm to enhance the accuracy, avoid the local optima and accelerate the convergence speed. The proposed algorithm has three main characteristics. Firstly, pattern search method enhances the exploitation ability of the basic CS algorithm. The proposed algorithm combines the random exploration of the CS algorithm and the exploitation capacity of pattern search method. Secondly, the optimal solution is obtained by self-adaptive competition mechanism. Hence, the proposed algorithm has a trade-off between searching speed and the diversity of solution. Finally, we realize the cooperation of the optimal solution to share, and strengthen the advantage of experience learning in the use of the optimal solution set search mechanism. The experimental results conducted on 52 benchmark functions show that the proposed algorithm is promising in terms of accuracy, success rate and robustness. And it is also suitable for multimodal and high-dimensional numerical optimization problems. Therefore, in terms of the global search ability and solution accuracy, our algorithm performs better than other modified CS algorithms, such as ICS (Improved Cuckoo Search algorithm), CSPSO (Cuckoo Search algorithm and Particle Swarm Optimization), OLCS (Orthogonal Learning Cuckoo Search algorithm), etc.

Key words: cuckoo search (CS) algorithm; coarse-to-fine search; Hooke-Jeeves pattern search; cooperation and sharing; adaptive competitive ranking; global optimization

1 引言

在日常生产生活中的诸多问题都可归结为全局最优化问题,采用传统的方法来解决此类问题效果不太理想,因此许多学者从模拟生物生活的习性角度出发解决此类问题,并受到了较好的效果.其中,布谷鸟搜索算法(Cuckoo Search,CS)则是近年来提出的一种新颖的元启发式全局优化方法[1].该方法模拟布谷鸟的寻窝产卵行为而设计出的一种基于莱维飞行(Lévy flights)机制的全

收稿日期:2014-06-03;修回日期:2014-09-14;责任编辑:覃怀银
基金项目:国家自然科学基金(No.61321491,No.61272219,No.61100110);国家 863 高技术研究发展计划(No.2007AA01Z334);江苏省科技计划(No.BE2011058,No.BY2012190);江苏省高校自然科学研究面上项目(No.15KJB520017);计算机软件新技术国家重点实验室创新基金重点项目(No.ZZKT2013A12);2014 年度江苏省"青蓝工程"优秀青年骨干教师项目基金;南京旅游职业学院科研创新团队建设项目基金

空间的搜索策略.在求解全局优化问题中表现出较好地性能.该算法具有选用参数少,全局搜索能力强,计算速度快和易于实现等优点,与粒子群优化算法和差分演化算法相比具有一定的竞争力[2].并在工程设计[3,4]、神经网络训练[5]、结构优化[6]、多目标优化[7]以及全局最优化[8~17]等领域取得了应用.

然而,CS 算法作为一种新的全局优化方法,搜索性能还有待提高.为此一些学者对该算法的全局寻优性能进行了改进,如 Valian 等学者提出利用参数自适应机制改进搜索步长与发现概率的 ICS(Improved Cuckoo Search algorithm)算法[8],从而提高了函数优化质量.此外,还有一些学者提出改进搜索机制中的步长[9,10]、动态自适应[11,12]、逐维改进机制[13]以及合作协同进化策略[14]等.这类改进算法在一定程度上提高了算法的搜索性能,取得了很好的寻优效果.然而单一的搜索策略在解决复杂的多维空间优化问题时,往往难以兼顾全局搜索与局部寻优的能力.另外,一些学者提出了与其他算法的杂交混合[15~18]的策略,如文献[16]提出了一种 CSPSO(Cuckoo Search algorithm and Particle Swarm Optimization algorithm)算法,利用 PSO 算法与 CS 莱维飞行策略杂交混合,达到一定的搜索性能.文献[17,18]提出了 OLCS(Orthogonal Learning Cuckoo Search algorithm)算法,在莱维飞行随机游动之后结合正交学习机制进行搜索从而增强了算法策略的寻优性能.这类算法加强了算法的搜索机制,可以取得更好的效果,但会增加算法的复杂性,并且在解决复杂问题及高维空间优化时,适应能力与鲁棒性不够,使得搜索效果不够理想等.其原因是目前的进化算法面对欺骗问题、多峰问题和孤立点等因素导致全局优化困难[19].因此,有必要继续探索新的改进方法与求解策略.

虽然 CS 算法全局探测能力优异,但是其局部搜索性能相对不足,特别是多模复杂函数的全局寻优时存在收敛速度慢、求解精度不高等问题,为了克服 CS 算法的缺点,提高其搜索性能,本文提出了一种基于模式搜索策略的布谷鸟搜索算法(strategy-Pattern Search based Cuckoo Search,PSCS).该算法基于模式搜索具有高效的局部趋化能力这一特点,在 CS 算法的框架下,嵌入模式搜索机制加强局部求解能力,利用 CS 算法较强地莱维飞行全局搜索能力和模式搜索的局部寻优性能,两者互为补充,兼顾均衡,从而避免搜索过程陷入局部最优.

2 CS 算法及求解复杂全局优化问题的局限性

标准 CS 算法是模拟布谷鸟寻窝产卵的特点形成理论,从而设计出基于莱维飞行搜索机制的随机优化算法,该算法及其参数见文献[1].

在莱维飞行随机游动搜索策略中,布谷鸟 i 根据式(1)进行寻窝搜索路径和位置的更新,并通过新的搜索位置 $\boldsymbol{X}_i^{t+1}$ 生成适应度值 F_i:

$$\boldsymbol{X}_i^{t+1} = \boldsymbol{X}_i^t + \alpha \oplus \text{Lévy}(\lambda) \tag{1}$$

其中,$i \in \{1,2,\cdots,n\}$,n 为鸟巢数量.$\boldsymbol{X}_i^t$ 和 $\boldsymbol{X}_i^{t+1}$ 表示第 i 个布谷鸟巢穴在第 t 代和 $t+1$ 代的位置向量 $\boldsymbol{X}_i = x_{i1}, x_{i2}, x_{i3,}, \cdots, x_{iD}$,$D$ 为每个巢穴位置的空间维数,α 为步长大小,用于控制随机搜索的范围,$\alpha = \alpha_0(\boldsymbol{X}_i^t - \boldsymbol{X}_b)$,$\alpha_0$ 是常数($\alpha_0 = 0.01$),$\boldsymbol{X}_b$ 表示当前最优解.式(1)本质上是随机行走方程.$\oplus$为点对点乘法,Lévy(λ)为随机搜索路径与时间 t 的关系服从莱维分布,表现为随机幂次形式的概率密度函数.

在偏好随机游动搜索策略中,算法以混合变异和交叉操作的方式重新生成若干个新解,如式(2)所示:

$$\boldsymbol{X}_i^{t+1} = \boldsymbol{X}_i^t + r \cdot (\boldsymbol{X}_j^t - \boldsymbol{X}_k^t) \tag{2}$$

其中,r 是缩放因子,是(0,1)区间内均匀分布的随机数,$\boldsymbol{X}_j^t$ 和 $\boldsymbol{X}_k^t$ 是两个随机解.

在 CS 算法中,莱维飞行搜索机制利用随机游动进行全局探测,根据偏好随机游动搜索策略指导局部寻优.复杂的全局优化问题由于多极值且变量间相互独立等特点,需要算法尽可能搜索到全局较好解的分布范围,扩大精细搜索力度.而基本的 CS 算法全局优化却存在以下局限性:(1)在迭代过程中,布谷鸟在当前位置的基础上以随机游动方式产生新的位置,单一的随机游动策略的搜索方式在搜索过程中具有很强的盲目性,导致难以快速地搜索到全局最优值,开发性能不足,搜索精度不高;(2)搜索到的位置评价后算法总是以贪婪方式选择较好的解,保存全局最优位置,而全局优化问题多极值使得布谷鸟易陷入对先前环境的局部寻优,导致早熟收敛;(3)CS 算法是以概率 P_a 放弃部分解而采用偏好随机游动方式重新生成新解来增加搜索位置的多样性,却忽视了学习与继承种群内优势群体的优良经验,增加了搜索空间的计算量与时间复杂度.

3 PSCS 算法的基本策略

基于莱维飞行的随机游动和偏好随机游动是 CS 算法中两个重要的搜索策略,由于莱维飞行其自身的特性使得搜索性能具有较好的随机性与全局探测能力,但是面对复杂的全局优化问题的求解时,其局限性就显露出来.针对上述 3 点不足,本文分别提出模式搜索趋化策略、自适应竞争排名机制与合作分享策略来弥补该算法在复杂的全局优化问题中的局限性.以期达到全局搜索和局部开发的平衡,使得算法的搜索性能更加优越.

3.1 模式搜索趋化策略

基于模式搜索局部趋化的布谷鸟算法的策略是以

CS 算法为基本框架,将模式搜索方法作为一种局部趋化搜索算子,嵌入到 CS 算法中,以加强求解精度.模式搜索(Pattern Search,PS)也叫 Hooke-Jeeves 算法[20],是由 Hooke 和 Jeeves 提出的一种基于坐标搜索法改进的搜索方法.该方法的原理是若要寻找搜索区域的最低点,可以先确定一条通往区域中心的山谷,然后沿着该山谷线方向前进搜索.探测移动(exploratory move)和模式移动(pattern move)是这种趋化策略的两个重要步骤,在迭代过程中交替进行,最终到达理想的求解精度.其中,探测移动的目的是探寻有利的趋化方向,而模式移动则沿着有利的方向快速搜索.其计算步骤如算法 1 所示.

算法 1　模式搜索趋化策略

Begin

　给定趋化策略的起始位置 x_1,步长 δ,分别设置加速、减速因子 α,β,步长计算精度 ε,k 和 j 为 1;

　确定初始位置 $y_1 = x_k$;

While ($\delta \leqslant \varepsilon$ && $j \leqslant D$)

　采用探测移动:从参考点出发,依次沿坐标轴方向 $d_j(j = 1,2,\cdots,D)$进行 2 个方向的探测;

　　沿正轴方向:若目标函数值 $f(y_j + \delta d_j) < f(y_j)$,设置 $y_{j+1} = y_j + \delta d_j$,否则沿负轴方向探测;

　　沿负轴方向:若目标函数值 $f(y_j - \delta d_j) < f(y_j)$,设置 $y_{j+1} = y_j - \delta d_j$,否则沿正轴方向探测;

　　得到新的位置 y_{j+1},设置 $x_{k+1} = y_{j+1}$;

　进行模式移动:沿着理想的目标函数值下降方向进行加速搜索;

　　若 $f(x_{k+1}) < f(x_k)$,设置 $y_1 = x_{k+1} + \alpha(x_{k+1} - x_k)$,$k = k + 1$;

　　否则,缩短轴向移动步长 $\delta = \delta\beta$;

　保留最好解;

End while

End

PS 趋化策略的本质是通过不断地成功的迭代,实现搜索步长的模式改进,从而加速算法的收敛.通过对当前搜索位置的探测与模式移动,达到趋化于更优值的直接搜索.在迭代过程中若找到相对于当前位置的更优点,则步长递增,并从该点位置进行下一次迭代;否则步长递减,继续搜索于当前位置.

以图 1 为例,若 x_k 迭代成功,则下次迭代从待定位置 $x' = x_k + \alpha(x_k - x_{k-1})$开始探测,其中 $x_k - x_{k-1}$为模式步长,沿着模式步长方向搜寻优于位置 x_k 更好的解.无论是否存在 $f(x') \leqslant f(x_k)$,都将以 x' 为基准位置进行坐标搜索.若 x' 坐标搜索成功,则令 $x_{k+1} = x'$,并从 x_{k+1}位置开始新的迭代搜索;否则,坐标搜索在 x_k 展开.若在位置 x_k 坐标搜索失败,则新一轮的坐标搜索步长减半在 x_{k-1}处展开.若在 x_{k-1}搜索仍然失败,回溯并重复上述过程.

模式搜索趋化策略是在算法的迭代过程中,如果满足 mod(gen, T)的整除条件,gen 表示当前迭代次数,$T = 2 * D$ 为与维数相关的模式搜索参数.这样在搜索的过程中,先由 CS 算法执行全局搜索得到新的群体,采用自适应竞争排名构建优势巢穴集,如果满足模式搜索条件,根据合作分享策略利用优势巢穴集生成新的模式搜索起始位置.从而利用 PS 搜索策略对该位置进行局部趋化,并评价优化后的结果,加强求解精度.

3.2　自适应竞争排名构建机制

为了有效求解复杂多极值全局优化问题,避免算法陷入局部最优,本文提出了一种自适应竞争排名构建方法,该方法根据适应度值进行自适应竞争排名,排在前面的构成优势巢穴集.该方法可使迭代初期强化竞争,减少排名数量,加快搜索;而迭代后期放宽名次数量,优势巢穴集扩大,便于合作分享信息,避免早熟.

构建优势巢穴集 N 的具体实施方法如下:根据排名机制保存多个优质巢穴,这些巢穴对应多个全局最好位置解,用这些解来指导模式搜索及位置更新,然后从更新后的 n 个巢穴中选取排名前 R 的优势巢穴进行保存.R 的定义如式(3)所示:

$$R = \begin{cases} \text{ceil}\left((R_{\max} - (R_{\max} - R_{\min})\dfrac{2(\text{nFE}-1)}{\text{maxNFES}})\cdot n\right), \\ \qquad \text{if nFE} \leqslant \dfrac{1}{2}\text{maxNFES} \\ \text{floor}\left((R_{\min} + (R_{\max} - R_{\min})\dfrac{\text{nFE}}{\text{maxNFES}})\cdot n\right), \\ \qquad \text{if nFE} > \dfrac{1}{2}\text{maxNFES} \end{cases} \tag{3}$$

其中,$R_{\max}$和 $R_{\min}$分别表示最大和最小排名数,ceil(·)与 floor(·)则表示向上和向下取整,nFE 为当前评价次数,maxNFES 是最大评价数.

通过这种自适应竞争排名机制构建优势巢穴集,使得迭代初期优势巢穴集较小,有利于快速搜索到全局较优解并能增强模式搜索的局部趋化能力,加速算法收敛.迭代中后期,该策略利用自适应排名机制的巢穴集,扩大了搜索范围,抑制过快早熟,从而使得算法不易陷入局部最优,保持了种群的多样性.

3.3　合作分享策略

基本的 CS 算法中采用的是偏好随机游动搜索策略,该策略存在启发信息不足,搜索慢的问题,为此,本文提出了一种合作分享策略.该策略利用合作分享优势集搜索机制,代替混合变异和交叉操作方式生成若干新解,有利于强化优势经验的学习.具体实施方法为:布谷鸟在位置更新时,随机选择优势巢穴集中的一个优势巢穴位置供当前模式搜索信息分享,该优势巢穴作为新的局部搜索的起始位置,指导模式搜索趋化

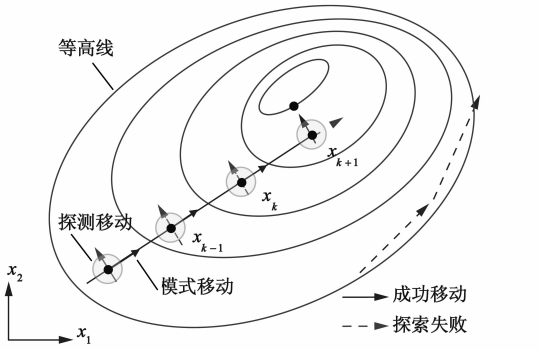


图1 PS算法步长搜索模式

寻优,从而避免过早收敛,防止陷入局部最优.合作分享策略引入线性惯性权重,以加强全局指导能力.利用合作分享策略选择模式搜索的起始位置,其分享策略如式(4)所示:

$$\boldsymbol{V}_i = \boldsymbol{N}_k + \varphi \cdot w \cdot (\boldsymbol{X}_j - \boldsymbol{N}_k) \tag{4}$$

$\boldsymbol{N}_k$ 为从优势巢穴集中随机选择一只布谷鸟 k 的巢穴位置,分享因子 $\varphi = \text{rand}[-1,1]$, w 为线性惯性权重,其计算公式如式(5)所示:

$$w = w_{\min} + \frac{\text{nFE}}{\text{maxNFES}} \cdot (w_{\max} - w_{\min}) \tag{5}$$

对于全局优化问题其目标函数值无限接近0时,对应的适应度值也非常小,当适应度值小于一定数量级时,很难区分适应度值的大小,为了解决这一问题,算法在实施过程中直接采用目标函数值来代替适应度值.

PSCS算法步骤算法2所示.

算法2 PSCS算法

```
Begin
    初始化 n 个布谷鸟巢穴 X_i(i = 1,2,…,n),迭代次数 gen 初始值设为 1;
    计算各个巢穴位置 X_i = (x_i1, x_i2, …, x_iD)^T 的适应度值 F_i = f(X_i);
  While (nFE < maxNFES)或(满足求解精度条件)
    (全局探测阶段)
    采用莱维飞行随机游动机制产生新的巢穴位置 X_i;
    评价新的巢穴位置 X_i 的适应度值 f(X_i);
    随机选择一个候选巢穴位置 X_j;
    If(f(X_i) < f(X_j))
        用新的巢穴位置 X_i 替代候选巢穴 X_j;
    End if
    按一定发现概率 P_a 丢弃差的巢穴;
    (局部开发阶段)
    自适应竞争排名构建机制:利用式(3)选取排名前 R 的优势巢穴进行保存为 N;
    合作分享策略:利用式(4)产生新的巢穴位置 V_k 替代丢弃位置并保留最好解;
    If (mod(gen, T) = = 0)
        模式搜索趋化策略:将巢穴位置 V_k 作为模式搜索的起始位置进行局部趋化于 V_k^*;
        If (f(V_k^*) < f(X_i))
            用新的巢穴位置 V_k^* 替代候选巢穴 X_i;
        End if
      gen = gen + 1 并保存最好解;
    End if
  End while (nFE = = maxNFES)
End
```

4 计算机数值仿真实验结果与算法比较

4.1 测试函数与评价标准

对这类优化算法的测评,有一些经典的测试函数.为了全面验证本文提出的PSCS算法的有效性和先进性,共选用了52个具有代表性的且为不同类型的典型测试函数对算法进行全面测试.测试函数主要分为三类.第一类是典型常用的16个高维测试函数[21],Ackley(AC)、Griewank(GR)、Penalized1(P_1)、Penalized2(P_2)、Quartic Noise(QN)、Rastrigin(RA)、NC_ Rastrigin(NR)、Rosenbrock(RO)、Schwefel1.2(S_{12})、Sphere Model(SM)、Step(ST)、Schwefel2.21(S_{21})、Schwefel2.22(S_{22})、Schwefel2.26(S_{26})、Weierstrass(WE)和Zakharov(ZA).所有函数的理论最优值都为0.其中,对 S_{26} 函数进行修正为求解全局最小值.这些测试函数固定维度为30,求解困难,对于算法的全局优化性能要求较高.以全局优化复杂单模态的高维RO香蕉型函数问题为例,其内部是一个长而狭窄、形如抛物线的平坦山谷地带,变量间相互关联,很难收敛于全局最优.目前已有的算法迭代后期基本停止进化,求解精度不高.第二类选用了26个固定维数的测试函数[21],BOhachevsky1(BO_1)、BOhachevsky2(BO_2)、BRanin(BR)、EaSom(ES)、GoldsteinPrice(GP)、Shekel′s Foxholes(SF)、SixhumpcamelBack(SB)、SHubert(SH)、SChaffer(SC)、Hartman3($H_{3,4}$)、Helical Valley(HV)、COlville(CO)、KOwalik(KO)、PErm(PE)、Power Sum(PS)、Sheke5($S_{4,5}$)、Shekel7($S_{4,7}$)、Shekel10($S_{4,10}$)、Hartman6($H_{6,4}$)、MIchalewicz(MI)、WhItley(WI)、Fletcher Powell(FP)、Modified Langerman(ML)、Modified Shekel′s Foxholes(MS)、POwell(PO)、Expansion F10(EF).其维数为指定的固定值,从2维至25维不等,部分函数搜索难度极高,如FP,$S_{4,10}$和ML等复杂多模态函数.这些函数表现为非对称,局部最优解随机分布,选择这些复杂的函数可以更好地测试本文算法的通用性.第三类为具有扰动的测试函数,以进一步验证PSCS算法求解连续全局优化问题的适应性与鲁棒性.选用了文献[12]中的前10个复杂变换后的测试函数 $F_1 \sim F_{10}$ 以便于与近年来新提出的CS改进算法进行比较.这些复杂的测试函数中包括变换和旋转的单峰和多峰函数,且变量间存在相

互独立与相互关联的特征;所以,这些函数在算法的求解过程中难度较高.

本文采用上述测试函数对 PSCS 算法进行了测试,并与传统的 CS 算法、近年来提出的改进的 CS 算法以及其他智能优化算法进行了实验比较.实验设备为一般笔记本电脑,CPU 为 Intel(R) Core(TM) 2 Duo CPU T6500 2.10GHz,4GB 内存,实验仿真软件是 Matlab 7.0.

为了更好地评估算法的性能,本文采用如下评价准则.

(1)适应度值误差 Error.如式(6)所示:

$$\text{Error} = v^* - v^\alpha \tag{6}$$

其中,$v^* = f(\boldsymbol{X})$表示算法搜索得到的解 $\boldsymbol{X}$ 对应的适应度值,$v^\alpha = f(\boldsymbol{X}^*)$为目标函数理论上的全局最优解 $\boldsymbol{X}^*$ 对应的适应度值.对于式(6)中的 Error 值的意义表现为值越小,求解精度越高.

(2)函数成功运行评价次数 NFEs.当算法在每次运行时,在当前函数评价次数没有达到最大评价次数且最优解的适应度值误差达到指定的求解精度(小于一定阈值)时的函数评价次数.

实验中,本文算法将最大评价次数分别设置为 100000 和 300000,根据式(7)定义的误差容许范围,测试结果是否成功.

$$\text{Error} < \varepsilon_1 \mid v^\alpha \mid + \varepsilon_2 \tag{7}$$

其中,ε_1 和 ε_2 为误差容许范围中精度控制参数 $\varepsilon_1 = \varepsilon_2$.其中前 42 个函数以及 $F_1 \sim F_5$的误差阈值精度设置为 10^{-6},$F_6 \sim F_{10}$的误差阈值为 10^{-2}.

(3)函数寻优成功率 SR.算法独立运行 30 次,达到误差阈值精度累计成功的实验次数与总实验运行 30 次的比值.

(4)算法收敛加速率 AR.为了测试算法的收敛速度,本文使用加速率来比较本文算法与 CS 算法的收敛速度,其公式定义如式(8)所示.其中NFE_{CS}和NFE_{PSCS}分别表示算法 CS 和 PSCS 关于函数的成功评价次数.

$$\text{AR} = \frac{\text{NFE}_{\text{CS}}}{\text{NFE}_{\text{PSCS}}} \tag{8}$$

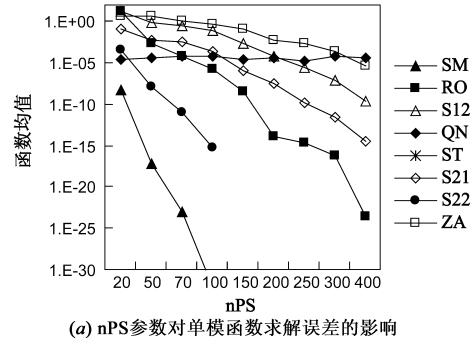
(5)算法的复杂性 AC.为定量评价算法的复杂性,采用式(9)度量算法的复杂性[22]

$$\text{AC} = \frac{\text{mean}(T_2) - T_1}{T_0} \tag{9}$$

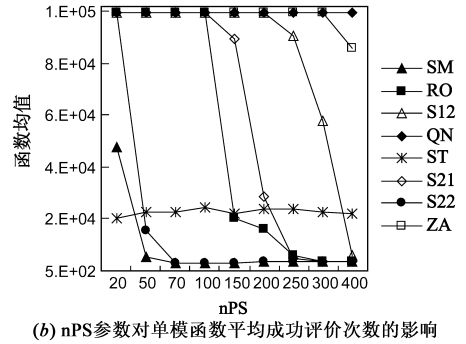
其中,T_0表示执行特定的测试程序[22]所需时间;T_1表示在 200000 次函数评价条件下,算法优化 F_3函数所需时间;mean(T_2)表示在 200000 次函数评价条件下,算法累计 5 次优化 F_3函数所需的平均时间.

4.2 PSCS 算法参数设置

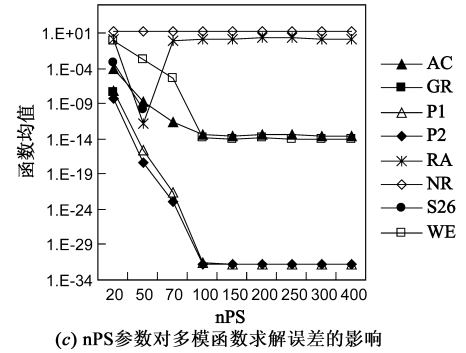
本文算法中基本的参数设置与 CS 算法设置相同,为了便于算法的比较,种群规模数定为 30,固定维数为 30,发现概率 $P_a = 0.25$.实验数据是在指定最大评价次数独立运行 30 次的情况下,取平均值 Mean,最好值 Best,最坏值 Worst,标准方差 SD(Standard Deviation)以及平均成功评价次数 NFEs.其中,平均成功评价次数是在 30 次独立运行下其收敛精度误差值小于指定阈值的平均成功评价次数.$R_{\max}$和 $R_{\min}$分别取 0.5 和 0.05.$w_{\min}$和 $w_{\max}$分别设置为 1 和 0.2.模式搜索中的 $\delta = 0.2$,$\alpha =$



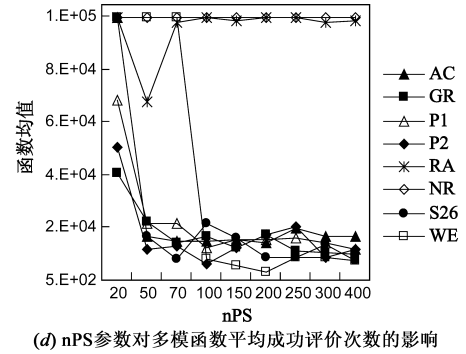
(*a*) nPS参数对单模函数求解误差的影响



(*b*) nPS参数对单模函数平均成功评价次数的影响



(*c*) nPS参数对多模函数求解误差的影响



(*d*) nPS参数对多模函数平均成功评价次数的影响

图2　nPS设置对16个测试函数的实验结果(D=30)

$1.0, \beta = 0.5, \varepsilon = \varepsilon_1 = \varepsilon_2$.

为了测试模式搜索最大迭代次数 nPS 值的不同选择对算法的影响,防止局部信息权重过高,可能会使算法搜索陷入局部最优解.本文选用了 16 个高维复杂测试函数来评测模式搜索次数对算法性能的影响,其中包括 8 个单模和 8 个多模复杂函数.16 个测试函数全局最优值都为 0,30 维最大评价次数为 100000 情况下的测试结果如图 2 所示.

测试方法是固定其余的参数的情况下,变化模式搜索 nPS 的范围取值为[20,400].从图 2(a)的实验结果来看,除了单模函数 QN 外,nPS 参数设置不同会对单模函数的收敛效果产生了明显的影响,随着 nPS 设置数值的增大,实验效果会趋于更优.而对于图 2(c)关于 8 个多模函数的测试结果来看则不同,当 nPS 数值达到 100 次后,取得了较好的求解值,随后,收敛效果趋于稳定.另外,图 2(b)、(d)实验的成功评价次数也进一步验证上述结论.所以,综上所述,为了达到 PSCS 算法全局搜索与局部趋化能力的平衡,nPS 设置值控制在[100,200]之间为宜,在 150 附近取值对算法的整体性能相对较好.所以,实验中 nPS 取为 150,有利于提高算法对不同类型函数优化的求解精度.

4.3 PSCS 与 CS 算法比较

表 1 和表 2 为本文 PSCS 算法与标准 CS 算法优化 52 个函数的适应值平均误差与标准差的实验结果.最大函数评价次数 maxNFES = 100000,其中,"≈"表示 CS 算法与 PSCS 算法的平均误差在 0.05 水平下的双侧 t-检验是不显著的;"☆"和"◎"表示标准 CS 算法与 PSCS 算法的平均误差在 0.05 水平下的双侧 t-检验是显著的,"☆"表示 CS 算法求解质量比 PSCS 算法差,而"◎"则代表求解精度比 PSCS 算法好.表中最好的实验结果为加粗显示.Sy(Symbol)表示函数简称.

表 1 CS 与 PSCS 算法对 26 个固定函数的测试结果

Sy	D	CS						PSCS					AR
		Mean NFEs	SD NFEs	SR	Mean	SD		Mean NFEs	SD NFEs	SR	Mean	SD	
BO_1	2	6660	845	1	0	0	≈	624	523	1	0	0	10.67
BO_2	2	7608	963	1	0	0	≈	624	349	1	0	0	12.19
BR	2	5064	1010	1	0.39789	0	≈	390	0	1	0.39789	0	12.98
ES	2	3984	445	1	−1	0	≈	7152	1220	1	−1	0	0.56
GP	2	5580	935	1	3	9.42E−16	≈	468	174	1	3	9.42E−16	11.92
SF	2	3594	2559	1	0.998	0	≈	3024	2374	1	0.998	0	1.19
SB	2	−	−	0	−1.0316	0	≈	−	−	0	−1.0316	0	−
SH	2	−	−	0	−186.7309	9.10E−14	≈	83300	37342	0.2	−186.7309	3.59E−11	−
SC	2	38916	10798	1	1.19E−08	1.62E−08	☆	16326	6273	1	0	0	2.38
$H_{3,4}$	3	4926	965	1	−3.8628	0	≈	6570	639	1	−3.8628	0	0.75
HV	3	18492	2967	1	6.44E−32	1.44E−31	☆	510	0	1	2.80E−42	3.91E−42	36.26
CO	4	66132	8486	1	6.94E−11	1.40E−10	☆	1260	891	1	2.17E−18	3.94E−18	52.49
KO	4	37338	5790	1	3.07E−04	6.87E−17	≈	8442	3999	1	3.07E−04	1.89E−18	4.42
PE	4	−	−	0	1.64E−03	1.09E−03	☆	−	−	0	1.05E−04	7.03E−05	−
PS	4	−	−	0	3.70E−04	2.66E−04	☆	37262	35737	0.9	6.05E−07	9.94E−07	−
$S_{4,5}$	4	21444	2704	1	−10.1532	1.78E−15	≈	29022	3653	1	−10.1532	0	0.74
$S_{4,7}$	4	22092	2803	1	−10.4029	8.88E−16	≈	28014	3933	1	−10.4029	0	0.79
$S_{4,10}$	4	24204	6020	1	−10.5364	1.26E−15	≈	33630	5056	1	−10.5364	8.88E−16	0.72
$H_{6,4}$	6	−	−	0	−3.322	0	≈	−	−	0	−3.322	0	−
FP_2	2	7650	1156	1	0	0	≈	390	0	1	0	0	19.62
FP_5	5	91500	13817	0.4	1.87E−04	3.96E−04	☆	1050	411	1	2.83E−28	3.06E−28	87.14
FP_{10}	10	−	−	0	3.50E+01	4.16E+01	☆	1890	1207	1	7.11E−15	9.24E−15	−
ML_{10}	10	−	−	0	−0.93971	3.13E−01	≈	−	−	0	−0.81971	4.92E−02	−
MS_{10}	10	−	−	0	−5.6881	4.19E+00	≈	−	−	0	−5.3002	4.49E+00	−
MI	10	−	−	0	−9.352	1.77E−01	≈	−	−	0	−9.1206	3.37E−01	−
WI	10	−	−	0	1.00E+10	−	☆	−	−	0	0.14547	6.99E−02	−
PO	24	−	−	0	1.68E−03	6.13E−04	☆	−	−	0	1.13E−05	6.45E−06	−
EF	25	−	−	0	6.81E+01	2.09E+01	≈	−	−	0	6.51E+01	8.22E+00	−
Ave.		0.55						0.65					15.93

从表 1 中可以看出,对于固定低维的函数而言,除了 ES、$H_{3,4}$、$S_{4,5}$、$S_{4,7}$和 $S_{4,10}$这 5 个函数虽然 CS 算法比 PSCS 算法评价次数略少,但是 PSCS 在保证求解精度的前提下,对于 $H_{3,4}$、$S_{4,5}$、$S_{4,7}$和 $S_{4,10}$这 4 个函数的求解标准偏差 SD 却更小,说明 PSCS 算法性能有较强的健壮性.然而,其他优化函数 PSCS 算法不仅表现出寻优精度显著提高,而且平均成功评价次数也明显减少,PSCS 算法的成功率从 CS 算法的 0.55 提高到 0.65,平

均加速率 AR 为 15.93.整体上,显示出 PSCS 算法较好　　的寻优性能与求解速度.

表 2　CS 与 PSCS 算法对 26 个高维函数的测试结果($D=30$, maxNFES = 100000)

Sy	CS					PSCS			
	Mean	SD	Best	Worst		Mean	SD	Best	Worst
AC	1.52E−03	4.87E−04	1.04E−03	2.06E−03	☆	2.91E−14	1.59E−15	2.84E−14	3.20E−14
GR	2.41E−04	2.91E−04	1.59E−05	6.24E−04	☆	0.00E+00	0.00E+00	0.00E+00	0.00E+00
P_1	4.06E−02	8.75E−02	6.56E−04	1.97E−01	☆	1.57E−32	0.00E+00	1.57E−32	1.57E−32
P_2	1.79E−05	3.21E−05	1.51E−06	7.52E−05	☆	1.35E−32	0.00E+00	1.35E−32	1.35E−32
QN	4.10E−05	1.50E−05	2.71E−05	6.14E−05	≈	2.30E−05	1.25E−05	3.16E−06	3.37E−05
RA	6.51E+01	4.26E+00	5.99E+01	7.07E+01	☆	1.39E+00	1.66E+00	2.89E−10	3.98E+00
NR	4.95E+01	8.05E+00	3.89E+01	5.90E+01	≈	1.64E+01	1.34E+00	1.50E+01	1.80E+01
RO	2.19E+01	8.47E−01	2.11E+01	2.31E+01	☆	2.85E−09	1.03E−09	1.66E−09	4.01E−09
S_{12}	4.39E+01	1.71E+01	2.61E+01	7.19E+01	☆	2.00E−03	1.35E−03	1.01E−06	3.50E−03
SM	5.04E−07	2.57E−07	2.54E−07	9.11E−07	☆	0.00E+00	0.00E+00	0.00E+00	0.00E+00
ST	0.00E+00	0.00E+00	0.00E+00	0.00E+00	≈	0.00E+00	0.00E+00	0.00E+00	0.00E+00
S_{21}	4.75E−01	1.22E−01	2.69E−01	5.93E−01	☆	9.83E−07	4.40E−07	6.08E−07	1.74E−06
S_{22}	5.55E−03	2.93E−03	2.80E−03	1.05E−02	☆	0.00E+00	0.00E+00	0.00E+00	0.00E+00
S_{26}	3.55E+03	2.53E+02	3.16E+03	3.87E+03	☆	0.00E+00	0.00E+00	0.00E+00	0.00E+00
WE	8.31E−01	4.81E−01	3.26E−01	1.49E+00	☆	7.11E−15	0.00E+00	7.11E−15	7.11E−15
ZA	6.08E+00	1.74E+00	4.58E+00	8.97E+00	☆	1.10E−01	4.04E−02	7.24E−02	1.54E−01
F_1	1.76E−06	7.80E−07	8.62E−07	2.94E−06	☆	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F_2	2.07E+02	4.99E+01	1.20E+02	2.44E+02	☆	5.18E−03	4.71E−03	1.43E−03	1.25E−02
F_3	6.10E+06	2.28E+06	3.67E+06	9.81E+06	☆	6.21E+04	1.68E+04	4.71E+04	9.05E+04
F_4	8.47E+03	3.88E+03	4.50E+03	1.42E+04	☆	7.87E+03	1.44E+03	5.89E+03	9.62E+03
F_5	4.44E+03	5.55E+02	3.71E+03	5.24E+03	☆	9.75E+02	5.46E+02	4.37E+02	1.78E+03
F_6	8.00E+09	4.47E+09	8.90E+03	1.00E+10	☆	8.32E+00	1.64E+01	9.40E−05	3.77E+01
F_7	1.45E−01	1.07E−01	4.08E−02	2.82E−01	☆	9.77E−16	3.27E−16	5.55E−16	1.44E−15
F_8	2.09E+01	5.47E−02	2.09E+01	2.10E+01	≈	2.00E+01	7.26E−05	2.00E+01	2.00E+01
F_9	7.49E+01	1.19E+01	5.57E+01	8.80E+01	☆	1.99E+00	1.41E+00	0.00E+00	2.98E+00
F_{10}	1.71E+02	4.88E+01	1.31E+02	2.33E+02	☆	1.54E+02	1.79E+01	1.28E+02	1.77E+02

表 3　CS 与 PSCS 算法对 26 个高维函数的测试结果($D=30$, maxNFES = 300000)

Sy	CS					PSCS			
	Mean	SD	Best	Worst		Mean	SD	Best	Worst
AC	6.82E−13	2.89E−13	2.20E−13	9.20E−13	☆	2.84E−14	0.00E+00	2.84E−14	2.84E−14
GR	0.00E+00	0.00E+00	0.00E+00	0.00E+00	≈	0.00E+00	0.00E+00	0.00E+00	0.00E+00
P_1	2.63E−13	5.89E−13	7.66E−21	1.32E−12	☆	1.57E−32	0.00E+00	1.57E−32	1.57E−32
P_2	2.18E−24	3.60E−24	2.13E−25	8.55E−24	☆	1.35E−32	0.00E+00	1.35E−32	1.35E−32
QN	4.14E−06	2.12E−06	2.41E−06	7.68E−06	≈	6.72E−06	5.08E−06	1.17E−06	1.11E−05
RA	2.95E+01	7.10E+00	1.77E+01	3.57E+01	☆	3.71E−10	6.81E−11	3.16E−10	4.90E−10
NR	2.22E+01	5.08E+00	1.35E+01	2.64E+01	≈	1.24E+01	1.82E+00	1.00E+01	1.50E+01
RO	8.37E+00	3.00E+00	5.41E+00	1.32E+01	☆	2.22E−09	9.51E−10	1.46E−09	3.87E−09
S_{12}	1.98E−02	1.24E−02	9.00E−03	3.52E−02	☆	3.06E−07	2.34E−07	1.07E−07	6.93E−07
SM	1.25E−25	6.90E−26	3.76E−26	1.87E−25	☆	0.00E+00	0.00E+00	0.00E+00	0.00E+00
ST	0.00E+00	0.00E+00	0.00E+00	0.00E+00	≈	0.00E+00	0.00E+00	0.00E+00	0.00E+00
S_{21}	5.88E−02	5.21E−02	9.78E−03	1.20E−01	☆	7.85E−08	5.53E−08	5.48E−09	1.56E−07
S_{22}	1.45E−11	7.33E−12	3.48E−12	2.34E−11	☆	0.00E+00	0.00E+00	0.00E+00	0.00E+00
S_{26}	1.79E+03	2.06E+02	1.54E+03	2.01E+03	☆	0.00E+00	0.00E+00	0.00E+00	0.00E+00
WE	1.10E−02	1.43E−02	2.93E−04	3.37E−02	☆	6.25E−04	1.65E−04	4.24E−04	7.91E−04
ZA	5.34E−04	2.73E−04	1.36E−04	9.03E−04	☆	8.51E−06	2.58E−06	4.30E−06	1.08E−05
F_1	7.42E−25	4.87E−25	2.11E−25	1.26E−24	☆	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F_2	4.96E−02	3.90E−02	2.11E−02	1.18E−01	☆	2.90E−06	5.01E−06	2.41E−07	1.18E−05
F_3	2.47E+06	9.21E+05	1.47E+06	3.84E+06	☆	3.54E+04	2.02E+04	1.43E+04	6.29E+04
F_4	5.31E+02	2.68E+02	2.62E+02	8.21E+02	≈	9.41E+02	9.89E+02	2.52E+02	2.63E+03
F_5	2.56E+03	5.47E+02	1.79E+03	3.17E+03	☆	7.76E+02	6.44E+02	1.94E+02	1.84E+03
F_6	1.00E+10	−	1.00E+10	1.00E+10	☆	2.30E−01	5.14E−02	1.65E−01	3.03E−01
F_7	6.48E−04	9.13E−04	2.91E−08	2.12E−03	☆	9.77E−16	3.27E−16	5.37E−16	8.25E−16
F_8	2.09E+01	9.29E−02	9.29E−02	2.10E+01	≈	2.04E+01	1.85E−02	2.04E+01	2.04E+01
F_9	3.26E+01	3.95E+00	2.61E+01	3.67E+01	☆	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F_{10}	1.68E+02	3.81E+01	1.28E+02	2.31E+02	☆	1.20E+02	1.58E+01	1.01E+02	1.40E+02

对于高维复杂函数的全局寻优,PSCS 算法比标准 CS 算法更加优越,从表 2 中明显看出,PSCS 算法有 6 个函数(GR、SM、ST、S_{22}、S_{26}和 F_1)直接搜索到全局最优值,在相同条件下 PSCS 算法测试的平均误差都优于 CS 算法.根据平均误差在 0.05 水平下的双侧 t-检验结果显示,在 26 个标准测试函数中 PSCS 算法有 20 个测试函数优于 CS 算法.显示出其优越的搜索性能.

对于复杂变换旋转的 $F_1 \sim F_{10}$中的单峰函数而言,PSCS 算法在 $F_1 \sim F_5$函数上的平均误差都明显优于 CS 算法,其中 F_1函数,本文算法在有限的评价次数内直接搜索到全局最优值,显示出其优越的性能;对于 $F_1 \sim F_{10}$中的多峰函数而言,除了在 F_8函数上,PSCS 算法的平均误差近似且略优于 CS 算法,搜索精度优势不够明显外,但在 $F_6 \sim F_7$和 $F_9 \sim F_{10}$函数上的平均误差都明显优于 CS 算法.不管对于具有变换特点的函数、还是变换且旋转的函数而言,PSCS 算法都显示出其优越的全局寻优能力,尤其对于 F_1、F_2、F_6和 F_7函数的测试,本文算法相比于 CS 算法的求解精度大幅提高.

表 4 CS 与 PSCS 算法对高维函数的平均成功函数评价次数($D = 30$, maxNFES = 300000)

Sy	CS			PSCS			AR
	Mean NFEs	SD NFEs	SR	Mean NFEs	SD NFEs	SR	
AC	165132	6202	1.0	17250	3354	1.0	9.57
GR	127104	17954	1.0	27000	16854	1.0	4.71
P_1	166980	37803	1.0	9750	5687	1.0	17.13
P_2	103008	4721	1.0	6750	3137	1.0	15.26
RA	–	–	0.0	99000	37274	1.0	–
RO	–	–	0.0	13500	3354	1.0	–
S_{12}	–	–	0.0	267000	16432	1.0	–
SM	96888	1749	1.0	3750	0	1.0	25.84
ST	38724	3963	1.0	23250	3137	1.0	1.67
S_{21}	–	–	0.0	104250	22249	1.0	–
S_{22}	186204	6311	1.0	3750	0	1.0	49.65
S_{26}	–	–	0.0	16500	7776	1.0	–
F_1	102756	1675	1.0	3750	0	1.0	27.40
F_2	–	–	0.0	295000	4330	0.7	–
F_7	289032	24525	0.2	31500	11124	1.0	9.18
F_9	–	–	0.0	177750	26330	1.0	–
Ave.			0.51			0.98	17.82

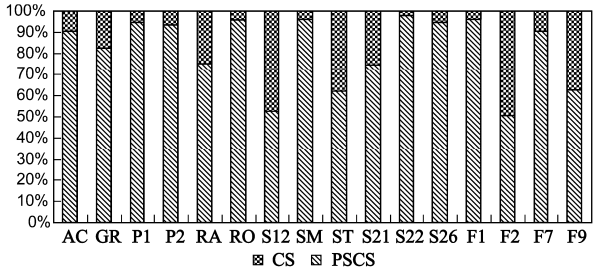


图3 CS和PSCS算法关于16个测试问题的函数评价次数(D=30, maxNFES=300000)

表 3 和表 4 为本文进一步测试 PSCS 算法和 CS 算法在评价次数为 300000 的情况下,平均误差、平均成功评价次数等结果.从结果上来看,表 4 中的平均寻优率 SR 从 0.51 提高到 0.98,说明随着评价次数的提高,PSCS 算法性能更具优势.进一步验证了 PSCS 算法不仅寻优率高,求解速度也相比 CS 算法显著提高的上述结论.另外,图 3 为 CS 和 PSCS 算法关于不同测试问题在最大评价次数为 300000 次的情况下函数评价次数百分比堆积柱形图,图中 CS 算法若评价次数达到最高评价次数仍未能成功评价,则以最高评价次数绘图.图中结果有效地说明本文 PSCS 算法相比传统 CS 算法寻优率高,评价次数少.

4.4 与改进 CS 算法以及其他智能优化算法的比较

为分析 PSCS 算法与其他改进 CS 算法的性能差异,表 5 列出本文 PSCS 算法与 ICS 算法[8]、CSPSO 算法[16]和 OLCS 算法[17,18]在 $D = 30$ 维空间上的性能比较结果.

分析表 5 可知,针对单峰函数,各算法的性能各异.在 F_1函数上,PSCS 算法和 ICS 算法性能相当,都能收敛到全局最好解,但明显优于 CSPSO 算法和 OLCS 算法的全局搜索性能;在 F_2和 F_4函数实验上,CSPSO 算法性能最优,其次是 PSCS 算法.而在 F_3和 F_5函数上,PSCS 算法的性能最优.针对复杂多峰函数而言,除了 F_{10}函数上,本文算法性能弱于 ICS 算法,其余 $F_6 \sim F_9$函数 PSCS 算法的性能都是最优,尤其 F_7和 F_9函数,本文算法求解精度提高显著.根据表 5 中针对平均误差检验统计结果,10 个复杂函数测试中 PSCS 算法有 7 个函数达到了最优,另外 3 个函数的测试结果也具有很好的竞争优势.另外,表 6 和图 4 为 PSCS 算法与其他改进的 CS 的平均函数成功评价次数的实验比对,进一步说明了本文策略的优越性.总体看来,PSCS 算法明显优于其它改进的 CS 算法.

在表 7 中,本文还将 PSCS 算法与近年来发表的其他智能优化算法[23](OEA, HPSO-TVAC, CLPSO, APSO)进行了比较,实验数据来自文献[23].表 7 中 OEA 算法的评价次数为 $3.0 * 10^5$,其他算法的评价次数均为 $2.0 *$

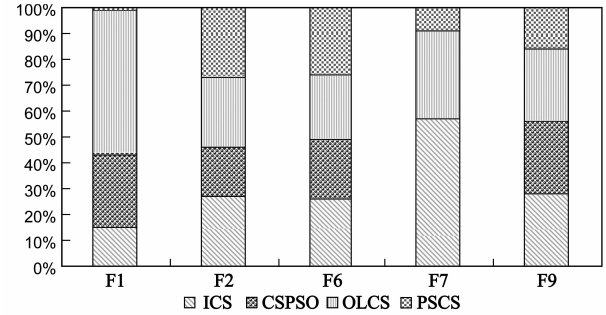


图4 ICS、CSPSO、OLCS和PSCS算法关于5个测试问题的函数评价次数(D=30, maxNFES=300000)

10^5.从表 7 可以看出,本文算法对于 SM,GR,S_{22}以及 ST 函数都能搜索到全局最优值,与 OEA,HPSO-TVAC,CLPSO 和 APSO 算法相比求解精度更高,搜索能力更强.

表 5　PSCS 与其他改进的 CS 的实验对比($D=30$,maxNFES = 300000)

Sy	D	ICS			CSPSO			OLCS			PSCS	
		Mean	SD		Mean	SD		Mean	SD		Mean	SD
F_1	30	0.00E+00	0.00E+00	≈	2.65E-28	2.67E-28	☆	2.41E-26	6.21E-26	☆	0.00E+00	0.00E+00
F_2	30	1.67E-03	2.87E-03	☆	1.41E-11	2.68E-10	◎	5.70E-02	4.79E-02	☆	1.01E-06	1.11E-06
F_3	30	3.85E+05	1.78E+05	☆	8.01E+05	6.49E+05	☆	2.57E+06	7.13E+05	☆	3.54E+04	2.02E+04
F_4	30	4.81E+02	3.92E+02	◎	5.93E+01	4.39E+01	◎	2.37E+03	1.23E+03	☆	1.01E+03	2.48E+02
F_5	30	1.63E+03	5.45E+02	☆	3.25E+03	9.33E+02	☆	2.44E+03	7.31E+02	☆	7.76E+02	6.44E+02
F_6	30	1.26E+01	9.73E+00	☆	6.56E+00	1.78E+01	☆	2.45E+01	1.99E+01	☆	2.30E-01	5.14E-02
F_7	30	2.09E-03	2.49E-03	☆	2.22E-02	1.20E-15	☆	4.72E-04	1.13E-03	☆	9.77E-16	3.27E-16
F_8	30	2.09E+01	2.06E-02	☆	2.09E+01	5.62E-02	☆	2.09E+01	5.31E-02	☆	2.00E+01	7.26E-05
F_9	30	1.61E+01	4.15E+00	☆	1.57E+02	2.21E+01	☆	3.54E+01	6.64E+00	☆	0.00E+00	0.00E+00
F_{10}	30	7.65E+01	1.05E+01	◎	2.52E+02	5.86E+01	☆	1.54E+02	3.74E+01	☆	1.20E+02	1.58E+01
☆/≈/◎		7/1/2			8/0/2			10/0/0			–	

表 6　PSCS 与其他改进的 CS 的平均函数成功评价次数的实验对比($D=30$,maxNFES = 300000)

Sy	D	ICS			CSPSO			OLCS			PSCS		
		Mean NFEs	SD NFEs	SR	Mean NFEs	SD NFEs	SR	Mean NFEs	SD NFEs	SR	Mean NFEs	SD NFEs	SR
F_1	30	51914	923	1	95845	2940	1	195993	2554	1	3750	0	1
F_2	30	–	–	0	206075	13386	1	–	–	0	295000	4330	0.7
F_6	30	–	–	0	270272	20778	0.5	–	–	0	–	–	0
F_7	30	193959	48311	1	2244	15645	1	110102	20000	1	31500	11124	1
F_9	30	–	–	0	–	–	0	–	–	0	177750	26330	1
Ave.				0.40			0.70			0.40			0.73

表 7　本文算法 PSCS 与 OEA,HPSO-TVAC,CLPSO,APSO 算法的比较

Sy	D	OEA		HPSO-TVAC		CLPSO		APSO		PSCS	
		mean	SD	mean	SD	mean	SD	mean	SD	mean	SD
SM	30	2.48E-30	1.13E-29	3.38E-41	8.50E-41	1.89E-19	1.49E-19	1.45E-150	5.73E-150	0	0
RO	100	2.27E-01	9.41E-01	1.30E+01	1.65E+01	1.10E+01	1.45E+01	2.84E-00	3.27E-00	1.10E-08	1.46E-08
AC	30	5.34E-14	2.95E-13	2.06E-10	9.45E-10	2.01E-12	9.22E-13	1.11E-14	3.55E-15	2.84E-14	0
GR	30	1.32E-02	1.56E-02	1.07E-02	1.14E-02	6.45E-13	2.07E-12	1.67E-02	2.41E-02	0	0
RA	30	5.43E-17	1.68E-16	2.39E-00	3.71E-00	2.57E-11	6.64E-11	5.80E-15	1.01E-14	3.71E-10	6.81E-11
NR	30	–	–	1.83E-00	2.65E-00	1.67E-01	3.79E-01	4.14E-16	1.45E-15	1.30E+01	2.00E+00
S_{22}	30	2.07E-13	2.44E-02	6.90E-23	6.89E-23	1.01E-13	6.54E-14	5.15E-84	1.44E-83	0	0
S_{12}	30	1.88E-09	3.73E-09	2.89E-07	2.97E-07	3.97E+02	1.42E+02	1.00E-10	2.13E-10	1.79E-05	1.77E-05
ST	30	0	0	0	0	0	0	0	0	0	0
QN	30	3.30E-03	1.10E-03	5.54E-02	2.08E-02	3.92E-03	1.14E-03	4.66E-03	1.70E-03	8.11E-06	1.64E-06
P_1	30	9.21E-30	6.44E-31	7.07E-30	4.05E-30	1.59E-21	1.93E-21	3.76E-31	1.20E-30	1.57E-32	0

在表 8 中,本文算法 LFABC 与其他改进的差分进化算法[23](SaDE,jDE,JADE)进行了比较.算法中的实验参数同文献[23],实验比较的结果如表 8 所示.从表中的实验结果来看,对于大都数实验函数 PSCS 算法相比于改进的 DE 算法都有更好的搜索性能.

5　算法复杂性分析与讨论

5.1　复杂性分析

在智能算法全局优化过程中,计算量主要集中在目标函数的评估阶段,其复杂性往往表现为对目标函数的评价次数.如果最大迭代次数为 M,标准的 CS 算法和 PSCS 算法的时间复杂度分别为 $O(M*N)$和 $O(M*(N+\mathrm{nPS}))$.如果固定最大迭代次数进行评测,PSCS 算法函数评价次数与 CS 算法的评价次数近似相等.如果设置相同的最大评价次数,PSCS 算法与 CS 算法的复杂度 $O(\mathrm{maxNFES})$即相同.所以,可以看出本文的实验结果是基于各算法相同的时间复杂度的前提下

测得,与标准 CS 算法及改进的 CS 算法相比并未增加时间复杂度,体现出实验的公平性.

表 8 本文算法 PSCS 与 SaDE,jDE,JADE 算法的比较

Sy	D	maxNFES	SaDE		jDE		JADE		PSCS	
			mean	SD	mean	SD	mean	SD	mean	SD
SM	30	$1.5*10^5$	4.5E-20	1.9E-14	2.5E-28	3.5E-28	1.8E-60	8.4E-60	0	0
RO	100	$2.0*10^6$	1.8E+01	6.7E+00	8.0E-02	5.6E-01	8.0E-02	5.6E-01	1.59E-10	2.18E-11
AC	30	$5.0*10^4$	2.7E-03	5.1E-04	3.5E-04	1.0E-04	8.2E-10	6.9E-10	3.27E-14	3.89E-15
GR	30	$5.0*10^4$	7.8E-04	1.2E-03	1.9E-05	5.8E-05	9.9E-08	6.0E-07	0	0
RA	30	$1.0*10^5$	1.2E-03	6.5E-04	1.5E-04	2.0E-04	1.0E-04	6.0E-05	3.71E-08	6.81E-09
S_{22}	30	$2.0*10^5$	1.9E-14	1.1E-14	1.5E-23	1.0E-23	1.8E-25	8.8E-25	0	0
S_{12}	30	$5.0*10^5$	9.0E-37	5.4E-36	5.2E-14	1.1E-13	5.7E-61	2.7E-60	4.49E-10	2.85E-10
S_{21}	30	$5.0*10^5$	7.4E-11	1.8E-10	1.4E-15	1.0E-15	8.2E-24	4.0E-23	3.18e-08	1.41e-08
ST	30	$1.0*10^4$	9.3E+02	1.8E+02	1.0E+03	2.2E+02	2.9E+00	1.2E+00	3.11E+02	4.40E+02
QN	30	$3.0*10^5$	4.8E-03	1.2E-03	3.3E-03	8.5E-04	6.4E-04	2.5E-04	6.72E-06	5.08E-06
P_1	30	$5.0*10^4$	1.9E-05	9.2E-06	1.6E-07	1.5E-07	4.6E-17	1.9E-16	1.57E-32	0
P_2	30	$5.0*10^4$	6.1E-05	2.0E-05	1.5E-06	9.8E-07	2.0E-16	6.5E-16	1.35E-32	0

表 9 PSCS 与 CS 不同搜索空间的计算复杂性

D	T_0	CS			PSCS		
		T_1	T_2	C	T_1	T_2	C
10	10.17	32.86	34.95	0.20	120.97	130.59	0.95
30		51.16	57.20	0.59	210.09	223.38	1.31
50		72.87	75.65	0.27	314.10	315.17	0.10

为了更好地定量评价算法的复杂性,本文采用式(9)度量算法的复杂性.表 9 为 CS 算法和 PSCS 算法在不同搜索空间上的计算复杂性.从表 9 可知,PSCS 算法的计算复杂度并未提高,主要由于 PSCS 算法利用自适应竞争排名构建机制与合作分享策略指导模式搜索的局部趋化,提高了算法的搜索性能.同时模式搜索的最大迭代次数 150 并未消耗太多的评价时间,由于竞争排名与合作机制对局部趋化的指导作用提高了求解精度反而使得总的评价次数减少.此外,我们进一步发现,随着维数增加,算法的时间消耗将逐步弱化,以致 PSCS 算法的复杂度与 CS 算法的复杂度差距缩小,并在维数增大到 50 的情况下 PSCS 的复杂度 0.10 优于 CS 算法的复杂度值 0.27,这也再一次验证了本文算法有较强的高维收敛速度与全局搜索性能的结论.

5.2 讨论

综合平均误差、平均函数评价次数、寻优率与加速率等的比较结果,PSCS 算法整体性能优异.以复杂高维的香蕉型 RO 函数求解问题为例,目前已有的算法迭代后期基本停止进化,而本文算法表现出较好的全局搜索性能,其原因为算法在全局寻优中有效地结合了模式搜索的局部趋化提供了全局寻优的有效信息,使得算法能有效地辨识搜索方向,从而达到了很好的全局探测能力与较高的寻优精度.

PSCS 算法对于复杂变换的 F_1和 F_9函数的优化,都能搜索到全局最优值,取得了很好的搜索性能.虽然两个函数变量间相互独立,复杂多变,但 PSCS 算法结合优势群体,基于合作分享使得种群有较好的学习能力,利用模式搜索趋化性能有效地引导整个群体进行加速搜索,提高求解精度.然而,在其它变量间相互关联的函数上,PSCS 算法的性能也优于 CS 算法,其原因是算法利用竞争排名与合作分享机制指导模式搜索,在一定程度上保存了变量间的相关性,使得算法能够利用搜索到的全局较好解加速整个种群进化,并保持种群的多样性.

6 结语

本文将全局寻优能力很强的莱维飞行机制与模式搜索趋化策略有效地结合在一起,并利用自适应竞争排名构建机制与合作分享策略全局指导模式搜索的局部趋化,构成了新的基于模式搜索的布谷鸟全局优化算法.算法针对 52 个典型的基准测试函数进行了性能测试,结果表明,本文 PSCS 算法在全局寻优能力,收敛速度和求解精度方面相比 CS 算法都有大幅提高.同时,PSCS 算法优化性能稳定,鲁棒性强,除了对于固定维数有较好的搜索效果外,特别适合多峰及高维函数的优化问题,与最新提出的改进的 CS 算法相比,具有更好地全局优化性能,是一种解决连续的全局优化问题较为理想的方法.

参考文献

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

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

[3] YANG X S. Cuckoo search for inverse problems and simulated driven shape optimization [J]. Journal of Computational Methods in Sciences and Engineering, 2011, 12(1): 129-137.

[4] LAYEB A, BOUSSALIA S R. A novel quantum inspired cuckoo search algorithm for bin packing problem[J]. International Journal of Information Technology and Computer Science, 2012, 4(5): 58 – 67.

[5] VALIAN E, MOHANNA S, TAVAKOLI S. Improved cuckoo search algorithm for feed forward neural network training[J]. International Journal of Artificial Intelligence & Applications, 2011, 2(3): 36 – 43.

[6] GANDOM I A, YANG X, ALAVI A. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems[J]. Engineering with Computers, 2013, 29(1): 17 – 35.

[7] YANG X S, DEB S. Multi-objective cuckoo search for design optimization[J]. Computers & Operations Research, 2013, 40(6): 1616 – 1624.

[8] 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.

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

[10] TUBA M, SUBOTIC M, STANAREVIC N. Modified cuckoo search algorithm for unconstrained optimization problems[A]. Proceedings of the 5th European Conference on European Computing Conference[C]. Wisconsin: World Scientific and Engineering Academy and Society (WSEAS), 2011. 263 – 268.

[11] 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.

[12] 张永▮,汪镭,吴启迪.动态适应布谷鸟搜索算法[J].控制与决策,2014,29(4):1 – 6.
ZHANG Yong-wei, WANG Lei, WU Qi-di. Dynamic adaptation cuckoo search algorithm[J]. Control and Decision, 2014, 29(4): 1 – 6. (in Chinese)

[13] 王李进,尹义龙,钟一文.逐维改进的布谷鸟搜索算法[J].软件学报,2013,24(11):2687 – 2698.
WANG Li-jin, YIN Yi-long, ZHONG Yi-wen. Cuckoo search algorithm with dimension by dimension improvement[J]. Journal of Software, 2013, 24(11): 2687 – 2698. (in Chinese)

[14] 胡欣欣,尹义龙.求解连续函数优化问题的合作协同进化布谷鸟搜索算法[J].模式识别与人工智能,2013,26(11):1041 – 1049.
HU Xin-xin, YIN Yi-long. Cooperative co-evolutionary cuckoo search algorithm for continuous function optimization problems[J]. PR & AI, 2013, 26(11): 1041 – 1049. (in Chinese)

[15] GHODRATI A, LOTFI S. A hybrid CS/PSO algorithm for global optimization[A]. Pan J S. Proceedings of the 4th Asian Conference on Intelligence Information and Database Systems[C]. Berlin: Springer, 2012. 89 – 98.

[16] WANG F, HE X S, LUO L G, et al. Hybrid optimization algorithm of PSO and cuckoo search[A]. Proceedings of the 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC)[C]. Zhengzhou: IEEE, 2011. 1172 – 1175.

[17] LIX T, WANG J N, YIN M H. Enhancing the performance of cuckoo search algorithm using orthogonal learning method[J]. Neural Computing and Applications, 2014, 24(6): 1233 – 1247.

[18] LIX T, YIN M H. Parameter estimation for chaotic systems using the cuckoo search algorithm with an orthogonal learning method[J]. Chinese Physics B, 2012, 21(5): 113 – 118.

[19] 李坤,黎明,陈昊.进化算法的困难性理论研究进展[J].电子学报,2014,42(2):383 – 390.
LI Kun, LI Ming, CHEN Hao. Research progress of hardness theories on evolutionary algorithm[J]. Acta Electronica Sinica, 2014, 42(2): 383 – 390. (in Chinese)

[20] HOOKE R, JEEVES T A. Direct search solution of numerical and statistical problems[J]. Journal of the ACM, 1961, 8(2): 212 – 229.

[21] GAO W F, LIU S Y. A modified artificial bee colony algorithm[J]. Computers and Operations Research, 2012, 39(3): 687 – 697.

[22] SUGANTHAN P N, HANSEN N, LIANG J J, et al. Problem Definitions and Evaluation Criteria for the CEC2005 Special Session on Real-parameter Optimization[R]. Singapore: Nanyang Technological University, 2005.

[23] GAO W F, LIU S Y. Improved artificial bee colony algorithm for global optimization[J]. Information Processing Letters, 2011, 111(17): 871 – 882.

作者简介

马　卫　男,1983 年生于江苏东台.讲师.现为南京大学计算机科学与技术系博士研究生.主要研究方向为智能优化、进化计算和计算机视觉.
E-mail: maweitian@163.com

孙正兴(通信作者)　男,1964 年生于江苏张家港.现为南京大学计算机科学与技术系教授、博士生导师.主要研究方向为智能人机交互、多媒体计算、计算机视觉.
E-mail: szx@nju.edu.cn