·人工智能与模式识别·　　文章编号：1000-3428(2022)08-0085-13　　文献标志码：A　　中图分类号：TP181

多策略调和的布谷鸟搜索算法

彭　虎[1]，李源汉[1]，邓长寿[1]，吴志健[2]

（1.九江学院 计算机与大数据科学学院，江西 九江 332005；2.武汉大学 计算机学院，武汉 430072）

摘　要：布谷鸟搜索(CS)算法是一种新型的群智能算法，结构简单且寻优能力较强，但存在勘探与开采不平衡以及易陷入局部极值的问题。提出一种多策略调和的布谷鸟搜索(MSRCS)算法，基于概率规则选择由自适应步长和改进解更新方法组成的调和策略对布谷鸟个体进行更新，其中自适应步长引导布谷鸟在更好的方向上寻优，3种改进的解更新方法分别从自身邻域、当前最优个体和随机位置3个角度对勘探和开采进行调和，从而提升全局搜索和局部搜索在迭代过程中的适应性。在CEC2013测试集的28个基准函数上的实验结果表明，MSRCS算法至少有12个测试函数优于原始CS及其7种改进算法且排名第一，在求解单峰、多峰和组合函数问题时寻优能力更强，同时相比于3种经典群智能优化算法具有更快的收敛速度和更高的解精度。

关键词：群智能算法；布谷鸟搜索算法；自适应步长；解更新方法；全局搜索

开放科学（资源服务）标志码（OSID）：

中文引用格式：彭虎，李源汉，邓长寿，等. 多策略调和的布谷鸟搜索算法[J]. 计算机工程，2022，48(8)：85-97.

英文引用格式：PENG H，LI Y Y，DENG C S，et al. Multi-strategy reconciled cuckoo search algorithm[J]. Computer Engineering，2022，48(8)：85-97.

Multi-Strategy Reconciled Cuckoo Search Algorithm

PENG Hu[1]，LI Yuanhan[1]，DENG Changshou[1]，WU Zhijian[2]

（1.School of Computer and Big Data Science，Jiujiang University，Jiujiang，Jiangxi 332005，China；
2.School of Computer Science，Wuhan University，Wuhan 430072，China）

【Abstract】The Cuckoo Search（CS）Algorithm is a new swarm intelligence optimization algorithm with a simple structure and good searching ability. Its disadvantages include an imbalance between exploration and exploitation and easily falling into the local optimum. To solve these problems，we propose a Multi-Strategy Reconciled Cuckoo Search（MSRCS）algorithm. The proposed algorithm is based on probability rules in selecting reconciliatory strategies，including a self-adaptive step size and modified solution-update methods to realize individual updates. The self-adaptive step size leads Cuckoos in a better direction. Three modified solution-update methods are searched from their respective neighborhoods，move toward the contemporary optimum，and generate a random solution to balance exploration and exploitation. This algorithm effectively improves the adjusting ability of global and local searches during the iteration process. The experimental results obtained using 28 benchmark functions of the CEC2013 test show that MSRCS has at least 12 functions that are superior to the original CS and its seven improved algorithms and ranks first，indicating better optimization ability in solving unimodal，multimodal，and combinatorial function problems. In addition，MSRCS yields better convergence speed and solution accuracy than three classical swarm intelligence optimization algorithms.

【Key words】swarm intelligence algorithm；Cuckoo Search（CS）algorithm；self-adaptive step size；solution-update method；global search

DOI：10. 19678/j. issn. 1000-3428. 0061622

0　概述

群智能算法是模拟自然界生物群体行为而设计的算法，能有效解决现实世界中的复杂问题。随着学者们对群智能算法研究的深入，在其基础上提出了蚁群优化（Ant Colony Optimization，ACO）[1]、粒子群优化（Particle Swarm Optimization，PSO）[2]、人工蜂群（Artificial Bee Colony，ABC）[3]、萤火虫算法（Firefly Algorithm，FA）[4]、布谷鸟搜索（Cuckoo Search，CS）[5]等多种优化算法。这些算法不仅能够解决复杂函数优化问题，而且适用于控制工程[6]、生产调度[7]、图像处理[8]等领域。

基金项目：国家自然科学基金（61763019）；江西省自然科学基金（20202BABL202019）。

作者简介：彭　虎（1981—），男，副教授、博士，主研方向为智能计算及应用；李源汉，本科生；邓长寿、吴志健，教授、博士。

收稿日期：2021-05-12　　**修回日期：**2021-07-30　　**E-mail：**hu_peng@whu.edu.cn

CS算法是受布谷鸟的寄生育雏行为启发而提出的,与其他群智能算法相比,一方面具有参数少、运算简单、寻优能力强等优点,另一方面存在勘探能力与开采能力不平衡以及容易陷入局部最优解的缺陷。近年来,国内外学者们提出了很多新的CS改进算法。RAKHSHANI等[9]提出捕捉和漂移布谷鸟搜索算法(SDCS),利用2种学习模式,一种倾向于增强全局搜索,另一种倾向于增强局部搜索,同时引入新的交叉和变异搜索算子,提高SDCS的搜索能力。PENG等[10]提出一种新的高斯布谷鸟搜索算法(GBCS),以随机的方式选择Lévy分布和高斯分布,防止算法的过早收敛以及增强算法的搜索能力。PENG等[11]还提出一种多策略串行布谷鸟搜索算法(MSSCS),通过对布谷鸟寻巢行为、驱逐行为和乞求行为的观察,设计3种新的学习策略以及一种多策略串行框架来提高算法的性能。LI等[12]提出一种自适应参数方法的改进布谷鸟算法(SACS),将2种新的变异规则通过线性递减概率规则进行组合以平衡算法的勘探与开发,再根据两个新参数的相对成功率,引入自适应参数作为均匀随机值,以增强种群的多样性。刘景森等[13]提出一种具有动态步长和发现概率的布谷鸟搜索算法(DCS),通过引入步长调整因子动态约束Lévy飞行移动步长,还使用具有均匀分布和F分布特性的随机惯性权重增强算法的多样性。周诗源等[14]提出一种基于布谷鸟搜索优化算法的多文档摘要方法,将经过数据预处理后的多文档句子信息作为CS算法的输入,再基于多目标函数生成包含原始文档重要信息的句子以组成最终的摘要。向庭立等[15]提出一种基于布谷鸟搜索的覆盖优化策略,将混合传感器节点随机部署在目标区域,利用CS算法初步确定移动传感器节点的候选目标位置,通过位置优化方案得到移动传感器节点的最佳目标位置以完成覆盖优化。AGARWAL等[16]提出一种基于布谷鸟搜索算法的任务调度方法,有助于在可用的虚拟机之间有效地分配任务,并保持总体响应时间最小。

通过分析CS算法Lévy飞行和随机游走的特性,发现在迭代过程中CS算法存在勘探能力与开采能力不平衡以及易陷入局部最优的缺陷。为解决此问题,本文提出一种多策略调和的布谷鸟搜索(Multi-Strategy Reconciled Cuckoo Search,MSRCS)算法。该算法设计基于自适应步长和改进解更新方法的调和策略,并利用线性递减概率规则对调和策略进行选择,实现多策略调和以平衡算法的勘探能力与开采能力,并在不同迭代过程中调节全局搜索和局部搜索。

1 布谷鸟搜索算法

2009年,YANG等[5]受布谷鸟的孵化寄生行为启发,提出布谷鸟搜索算法。在自然界中,布谷鸟寻找适合产卵的巢穴的行为是随机的。因此,为了模拟布谷鸟找到巢穴的方式,设置以下简化规则:

1)每只布谷鸟一次只下一颗蛋,并随机选择一个鸟巢下蛋。

2)在随机选择的鸟巢中,最好的鸟巢将被保存到下一代。

3)鸟巢的数量是恒定的。鸟巢的主人抛弃外来蛋的概率为$p_a\in(0,1)$。

CS算法中每一个鸟巢代表问题的一个解,携带着鸟蛋的布谷鸟会通过Lévy飞行随机地移动到一个寄主鸟的巢中产卵。通过评价函数,对替换后的巢进行评价。因此,许多更好的解将被发现和更新。同时布谷鸟的蛋如果被寄主鸟发现,则寄主鸟抛弃布谷鸟的蛋以及放弃自己的蛋和巢,并再建新巢。

原始的CS算法主要包括Lévy飞行和随机游走2种重要策略。每一代布谷鸟都通过Lévy飞行寻找更好的巢来产卵。因此,布谷鸟的移动由Lévy飞行主导,即:

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

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

其中:$\boldsymbol{X}_i^{t+1}$是由式(1)产生的新解;$\boldsymbol{X}_i^t$和$\boldsymbol{X}_{\text{best}}^t$分别表示当前解和当代最优解;$t$为当前迭代次数;$\alpha$是控制布谷鸟移动步长大小的步长因子,$\alpha_0$通常等于0.01;$\otimes$表示向量的乘积。Lévy飞行产生的步长Lévy($\lambda$)服从Lévy分布,步长公式如下:

$$\text{Lévy}(\lambda)=\frac{\varphi\times\mu}{|v|^{1/\beta}} \tag{3}$$

其中:μ和v是服从均匀分布的随机数;β为1.5。φ的表示方法如下:

$$\varphi=\frac{\Gamma(\beta+1)\sin(\pi\beta/2)}{\Gamma[(\beta+1)/2\times\beta\times2^{(\beta-1)/2}]} \tag{4}$$

布谷鸟通过式(1)的Lévy飞行寻找到新的寄主鸟巢并将蛋放置于巢后,若寄主鸟发现布谷鸟的蛋,则会按概率p_a抛弃部分巢,并使用随机游走策略新建与被抛弃巢数量相等的新巢,公式如下:

$$\boldsymbol{X}_i^{t+1}=\boldsymbol{X}_i^t+r\otimes(p_a\pm\varepsilon)\otimes(\boldsymbol{X}_j^t-\boldsymbol{X}_k^t) \tag{5}$$

其中:r是服从均匀分布且在(0,1)中的随机数;$\boldsymbol{X}_j^t$和$\boldsymbol{X}_k^t$分别为第t代随机挑选的两个解。

CS算法流程如下:

1.初始化个体X_i(i=1,2,…,n),计算其适应度值;
2.While $t<t_{max}$
3.For i=1:n
4.根据Lévy飞行式(1)和式(2)更新解X_i^t;
5.计算新解X_i^{t+1}的适应度值,若新解优于旧解则替换;
6.End For
7.根据抛弃概率p_a和随机游走式(5)来更新解;
8.计算新解X_i^{t+1}的适应度值,若新解优于旧解则替换;
9.End While

2 多策略调和的CS算法

2.1 自适应步长

自适应步长是提高CS搜索效率的有效方法。张永韡等[17]根据布谷鸟每次迭代成功更新解的改善率调整步长大小,提出动态适应布谷鸟搜索算法(DACS)。若改善率较大,说明搜索空间相对单调,算法找到更优解的概率较高,故适当增大步长增加种群的探索性和搜索范围;若改善率较小,说明搜索空间相对复杂,找到更优解的概率较低,以适当减小

步长来提高算法局部开发性。

Lévy飞行是原始CS算法中一种重要的随机搜索策略,以随机的小步长或偶然的大步长控制布谷鸟移动。在原始CS算法迭代过程的前期,Lévy飞行搜索范围较小,效益较低,迭代次数增加从而导致算法收敛性较差。在迭代过程的中后期,算法逐渐靠近最优解,全局搜索减弱并逐渐转至局部搜索,易使算法陷入局部极值。

基于以上分析,本文提出一种新的自适应步长策略以提高CS算法的搜索效率,即自适应步长随迭代次数的增加而减小。在迭代前期,MSRCS的大步长扩大了其搜索范围,提高算法的全局搜索能力,有利于算法更快找到更优解,加快收敛速度;在迭代后期,小步长有利于算法的局部搜索。步长的规律变化对布谷鸟的搜索起引导作用,即在整体上从全局搜索逐渐过渡到局部搜索。该策略主要是由线性递减的自适应步长控制因子α_0决定,公式如下:

$$\alpha_0=\eta\times\cos\left[\frac{\pi}{3}\times\left(1+\frac{t}{2G}\right)\right] \tag{6}$$

其中:η是控制步长变化范围的调和因子,并决定步长的初值。根据余弦函数在范围内呈递减变化的特性,将步长与迭代次数进行关联,使步长也随着迭代次数的增加而减小。自适应α_0为近似一条直线的曲线,为整个搜索过程提供了轻微的缓冲作用,α_0的取值范围为[0.00,0.25]。

2.2　解更新方法

由于原始CS算法具有参数少、操作简单的特点,因此单策略CS算法的可操作性有限,布谷鸟的行动相似,且对于求解复杂的高维优化问题,若一只布谷鸟陷入局部极值,则无法靠邻域内相同情况的其他布谷鸟跳出局部极值。GAO等[18]为了解决这些问题,提出一种多策略自适应布谷鸟算法(MSACS),采用自适应参数控制5种新的搜索策略进行相互协作,以提高算法的搜索效率。由此证明多策略CS算法能够针对迭代过程的不同阶段采用不同策略,相比灵活性较差的单策略,不仅能增加算法的多样性,还能增强算法应对不同场景的搜索能力。LIU等[19]对BSO算法的多策略改进也取得了显著成效。因此,本文提出CS-current、CS-best和CS-rand 3种不同的解更新方法。

CS-current解更新方法以布谷鸟自身当前位置为中心,在其邻域内搜索潜在更优解,并加入$\boldsymbol{X}_{\text{best}}^t$控制寻优方向,防止布谷鸟在其他方向上进行无效的搜索和评价次数的浪费。如图1所示,黑点表示布谷鸟个体,虚线表示布谷鸟的邻域范围,此时布谷鸟个体分布较分散,每个个体邻域内都可能存在更优解,布谷鸟以自身为学习目标,尝试在自身邻域内搜索更优解,增强局部搜索能力有助于寻找解周围的潜在更优解。CS-current解更新方法的表达式如下:

$$\boldsymbol{X}_i^{t+1}=\boldsymbol{X}_i^t+\boldsymbol{R}_d\times[(\boldsymbol{X}_{\text{best}}^t-\boldsymbol{X}_j^t)+(\boldsymbol{X}_j^t-\boldsymbol{X}_k^t)] \tag{7}$$

其中:$\boldsymbol{X}_i^t$是当前解;$\boldsymbol{X}_{\text{best}}^t$是当代最优解;$\boldsymbol{X}_j^t$和$\boldsymbol{X}_k^t$是当前种群随机选取的两个解;$\boldsymbol{R}_d$是一行$d$维且在(0,1)中的随机数矩阵,服从均匀分布。相于比一个单纯的随机数,它改变了简单的学习方式,使各维度的信息多样化。

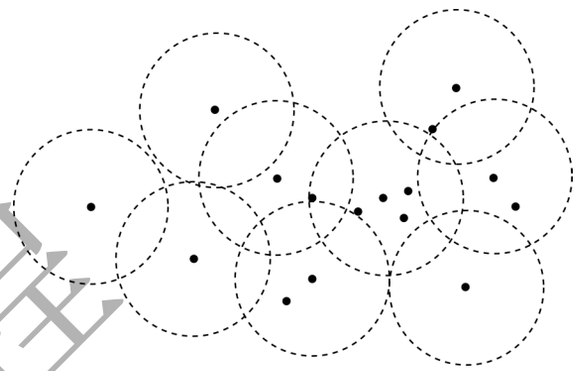


图1　CS-current解更新方法示意图

Fig.1　Schematic diagram of the solution-update method of CS-current

CS-best解更新方法以当前最优解为引导,使当前布谷鸟向着当前最优解的方向移动。彭虎等[20]在ELDDE算法中设计的精英区域学习策略,通过对精英个体的学习显著提高了算法性能。如图2所示,灰点表示当代最优解$\boldsymbol{X}_{\text{best}}^t$,虚线箭头表示布谷鸟的移动方向,此时布谷鸟种群密度较大,且大部分布谷鸟个体集中在$\boldsymbol{X}_{\text{best}}^t$周围,但仍有少部分分散在密集种群外,在密集种群外的布谷鸟以$\boldsymbol{X}_{\text{best}}^t$为学习目标,向其所在种群密度大的方向移动,有利于加快算法的收敛速度。CS-best解更新方法的表达式如下:

$$\boldsymbol{X}_i^{t+1}=\boldsymbol{X}_{\text{best}}^t+\boldsymbol{R}_d\times[(\boldsymbol{X}_{\text{best}}^t-\boldsymbol{X}_j^t)+(\boldsymbol{X}_k^t-\boldsymbol{X}_i^t)] \tag{8}$$

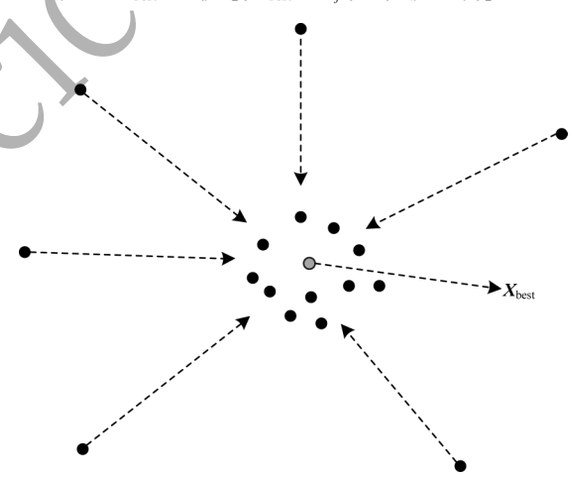


图2　CS-best解更新方法示意图

Fig.2　Schematic diagram of the solution-update method of CS-best

CS-rand解更新方法通过学习两个随机个体各维度的不同信息,生成一个新的随机个体。如图3所示,白点表示随机选取的两个布谷鸟个体,灰点表示以两个随机个体为学习目标产生的新个体,新个体在密集种群范围之外,且有可能优于当代最优解,有助于算法跳出局部极值以寻找到更优解。CS-rand解更新方法的表达式如下:

$$\boldsymbol{X}_i^{t+1}=\boldsymbol{V}+r\times\boldsymbol{R}_d\times[(\boldsymbol{X}_j^t-\boldsymbol{X}_i^t)+(\boldsymbol{X}_k^t-\boldsymbol{X}_i^t)] \quad (9)$$

$$\boldsymbol{V}=r^*\boldsymbol{X}_m^t+(1-r)^*\boldsymbol{X}_n^t \quad (10)$$

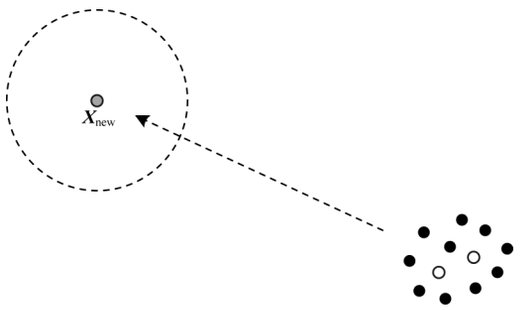


图3 CS-rand解更新方法示意图

Fig.3 Schematic diagram of the solution-update method of CS-rand

基于以上分析发现,3种新的解更新方法能够从多方面寻找算法更优解,根据算法在迭代过程中的缺陷使用不同的解更新方法在搜索空间中进行寻优,不仅能在迭代过程中加快收敛速度,而且有效克服算法收敛停滞,提高了算法跳出局部极值的能力。

2.3 多策略调和

为适应算法在不同阶段的需求,使策略的选择更加有效,对自适应步长和3种改进的解更新方法组成的调和策略进行选择,以调和算法的勘探与开采。

在SACS[8]算法中的线性递减概率规则$(1-t/G)$利用一个随机值与该规则产生的调和值进行比较,使算法能够在不同阶段对策略进行自适应选择。相比于原始CS算法,该规则使用动态变化的值有效地从前期向后期自适应过渡。在MSRCS算法中,使用了线性递减规则作为多策略调和的框架,对调和策略进行概率选择。在迭代前期,调和值较大,选择调和策略1的概率也更大,但随着迭代次数的增加,调和值逐渐减小,则概率逐渐向调和策略2偏移。当$t=G/2$时,选择两个调和策略的概率相等。之后选择调和策略2的概率逐渐增加。

调和策略1将自适应步长和CS-current结合,并设置调和因子p_1=0.8来调节选择概率。

调和策略1的流程如下:

1.If r<p_1

2.使用自适应Lévy飞行更新个体;

3.Else

4.使用CS-current方法更新个体;

在迭代前期,算法的最优解不明显,且解较为分散,适度使用CS-current有利于算法更快地找到自身邻域内的潜在最优解,在迭代前期对算法的勘探与开采起调和作用。

调和策略2将自适应步长和CS-best、CS-rand结合。自适应步长和解更新方法的概率调和因子p_2=0.5,对2种解更新方法的概率调和因子为p_1。

调和策略2的流程如下:

1.If r<p_2

2.使用自适应Lévy飞行更新个体;

3.Else

4.If r<p_1

5.使用CS-best方法更新个体;

6.Else

7.使用CS-rand方法更新个体;

8.End

9.End

在迭代中后期,自适应步长逐渐减小,全局搜索逐渐过渡至局部搜索,解向当前最优解逐步收敛。若目标函数值与全局最优解近似,自适应的小步长或者CS-best则有利于向最优解收敛。若目标函数值与全局最优解相差较大,或陷入局部极值,则基于随机思想,算法有小概率使用CS-rand尝试寻找更优解或尝试跳出局部极值,且适当增大选择CS-best和CS-rand的概率,在迭代后期使算法的收敛性、多样性和解的精度得以提升。

调和策略的选择建立在线性递减概率规则的框架下,根据迭代次数动态更新选择概率,达到调和策略之间从前期向后期的自适应过渡,以及调和算法的勘探和开采能力。

MSRCS算法的流程如下:

1.初始化个体X_i(i=1,2,…,n),计算其适应度值;

2.While t<t_{max}

3.If r<$(1-t/G)$

4.使用调和策略1更新个体X_i;

5.Else

6.使用调和策略2更新个体X_i;

7.End if

8.计算新解X_i^{t+1}的适应度值,若新解优于旧解则替换;

9.根据抛弃概率p_a和随机游走式(5)更新个体;

10.计算新解X_i^{t+1}的适应度值,若新解优于旧解则替换;

11.End While

2.4 算法复杂度

为证明MSRCS不会增加CS的时间复杂度,对CS和MSRCS进行复杂度分析。假设在CS流程中,初始化阶段生成解的时间为t_1,计算d维问题的目标函数时间为$f(d)$,种群规模为n,则整个初始化阶段的时间复杂度为$O(n(t_1d+f(d)))\approx O(d+f(d))$。迭代阶段采用式(1)计算Lévy飞行生成新解的时间为t_2,比较产生的新解与旧解的时间为t_3,新解替换旧解的时间为t_4,且记录该过程最优解的时间为t_5,则Lévy飞行阶段的时间复杂度为$O(n(t_2d+f(d)+t_3+t_4d+t_5d))\approx O(d+f(d))$。采用式(5)计算随机游走生成新解的时间为$t_6$,比较新解与旧解,替换旧解及记录最优解的时间与Lévy飞行阶段相同,随机游走阶段的时间复杂度为$O(n(t_2d+f(d)+t_3+t_4d+t_5d))\approx O(d+f(d))$。最后比较当代最优解与当前最优解的时间为$t_7$,替换当前最优解的时间为$t_8$,更新当前最优解的时间复杂度为$O(n(t_7+t_8d))\approx O(d)$。假设最大迭代次数为$G$,CS算法总时间复杂度为$T(n)=G\cdot(3O(n(d+f(d)))+O(d))\approx O(G\cdot n(d+f(d)))$。

MSRCS仅在Lévy飞行阶段进行改进,其他部

分与CS相同。假设调和策略1产生新解的时间为t_9，调和策略2产生新解的时间为t_{10}，该阶段的时间复杂度为$O(n(t_9 d+f(d)+t_3+t_4 d+t_5 d))$或$O(n(t_{10} d+f(d)+t_3+t_4 d+t_5 d))$。MSRCS总的时间复杂度为$T(n)=O(G\cdot n(t_1 d+f(d)))+O(G\cdot p_a\cdot n(t_9 d+f(d)+t_3+t_4 d+t_5 d))+O(G\cdot(1-p_a)\cdot n(t_{10} d+f(d)+t_3+t_4 d+t_5 d)+t_8 d))\approx O(G\cdot n(d+f(d)))$。

由此可见，在各参数相同的情况下，MSRCS与CS的算法时间复杂度一致。同时MSRCS未使用额外的存储空间存放数据，空间复杂度也相同。因此，MSRCS的改进并未增加算法的复杂度。

3　实验与结果分析

3.1　实验设置

为了验证MSRCS的性能，在广泛使用的CEC2013[21]测试集上进行实验。CEC2013包含28个测试函数，其中f_1~f_5为单峰函数，f_6~f_{20}为多峰函数，f_{21}~f_{28}为组合函数，函数的求解约束范围为$[-100,100]^d$。为了探究MSRCS的准确性和收敛性，将其与原始CS及其7种改进算法进行对比分析，这7种CS改进算法分别为ACS[22]、DACS[17]、SACS[8]、GBCS[10]、GCS[23]、NACS[24]和SDCS[9]，表1给出了各算法的具体参数设置，其中CS、ACS、DACS、SACS、GBCS、GCS、NACS和SDCS参数遵循相应文献中的设置。通过显著性水平为0.05的Wilcoxon秩和检验与Friedman检验对实验结果进行统计分析。Wilcoxon秩和检验通过将两组的值排秩次，比较两组的秩次总和来判断是否存在统计学差异。Friedman检验利用秩分析多组相关样本之间是否存在统计学差异。MSRCS及其他比较算法在Matlab R2016a上编程实现。MSRCS的Matlab代码可从https://whuph.github.io/index.html下载。

表1　算法参数设置

Table 1　Parameter setting of algorithms

算法	参数设置
CS	α_0=0.01, β=1.5, p_a=0.25
ACS	p=0.25
DACS	f_p=1.05, f_α=2, $p_{\max}$=1, $p_{\min}$=0, p_a=0.25
SACS	α_0=0.01, β=1.5
GBCS	α_0=0.01, β=1.5, p_a=0.25
GCS	μ=0.000 1, α_0=0.5, p_a=0.25
NACS	m=3, p_s=0.8, p_a=0.25
SDCS	α_0=0.01, β=1.5, ω=0.005, J=0.03, p_a=0.25
MSRCS	β=1.5, p_a=0.25, p_1=0.8, p_2=0.5

为了保证实验的公平性，设置种群数量n为20、问题维度d为30、发现概率p_a为0.25、总评价次数$F_{\max}$为1E6、最大迭代次数$G_{\max}$为25 000。步长控制因子通过对比不同η值的实验结果，选出其中结果最好的η值作为最终的η值。每个测试函数在相同环境下运行30次，并记录下其平均误差值与标准误差值。

3.2　与原始CS及其改进算法的比较

将MSRCS与原始CS以及7种CS改进算法进行性能对比。表2和表3给出了d=30时MSRCS与其他8种CS算法在CEC2013测试函数上的实验结果，其中，加粗的字体表示最好结果，Avg和Std分别表示平均误差值和标准误差值，rank表示Friedman检验的平均排名结果，Wilcoxon秩和检验结果中的“+”表示对比算法结果优于MSRCS，“−”表示对比算法结果差于MSRCS，“≈”表示对比算法和MSRCS结果相差不大。

表2　d=30时CS、ACS、DACS、SACS、MSRCS算法的CEC2013测试函数实验结果

Table 2　Experimental results of CEC2013 test functions for CS, ACS, DACS, SACS, MSRCS algorithms when d=30

函数	Avg±Std				
	CS	ACS	DACS	SACS	MSRCS
f_1	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
f_2	7.08E+05±2.97E+05	1.58E+06±7.57E+05	**3.83E+03±3.70E+03**	2.13E+06±4.73E+05	4.38E+04±1.70E+04
f_3	**1.00E+10±0.00E+00**	**1.00E+10±0.00E+00**	**1.00E+10±0.00E+00**	**1.00E+10±0.00E+00**	**1.00E+10±0.00E+00**
f_4	9.60E+02±3.90E+02	1.34E+02±6.43E+01	1.11E+03±6.63E+02	6.75E+03±1.54E+03	**7.49E−01±6.91E−01**
f_5	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	7.20E−14±5.48E−14	5.31E−14±5.67E−14
f_6	4.08E+00±7.52E+00	3.54E+00±8.97E+00	**3.52E+00±8.98E+00**	1.61E+01±2.09E+00	4.66E+00±9.73E+00
f_7	1.12E+02±2.65E+01	7.35E+01±2.82E+01	1.28E+02±1.02E+01	1.03E+02±1.46E+01	**3.93E+01±1.65E+01**
f_8	2.09E+01±4.80E−02	2.09E+01±4.61E−02	2.09E+01±4.25E−02	2.09E+01±4.26E−02	**2.08E+01±9.74E−02**
f_9	2.98E+01±1.90E+00	2.84E+01±1.97E+00	3.24E+01±1.77E+00	3.02E+01±1.82E+00	**2.59E+01±3.14E+00**
f_{10}	**2.64E-02±1.73E−02**	3.34E−02±1.95E−02	9.25E−02±5.22E−02	9.71E−02±3.38E−02	1.10E−01±5.38E−02
f_{11}	1.21E+01±6.95E+00	1.68E+01±7.88E+00	5.54E+01±1.71E+01	**3.32E−02±1.79E−01**	2.16E+01±5.91E+00
f_{12}	1.61E+02±3.31E+01	1.02E+02±2.51E+01	1.11E+02±3.36E+01	1.87E+02±4.12E+01	**7.45E+01±2.09E+01**
f_{13}	2.03E+02±3.14E+01	**1.49E+02±2.84E+01**	1.80E+02±4.33E+01	2.16E+02±2.36E+01	1.25E+02±2.14E+01
f_{14}	4.21E+02±1.60E+02	5.77E+02±2.72E+02	2.95E+03±5.68E+02	**9.42E+01±2.99E−01**	1.52E+03±4.00E+02
f_{15}	4.21E+03±2.46E+02	4.48E+03±2.89E+02	3.90E+03±2.98E+02	3.96E+03±4.03E+02	**3.78E+03±5.31E+02**
f_{16}	1.32E+00±2.07E−01	1.02E+00±1.69E−01	1.42E+00±2.52E−01	1.22E+00±1.85E−01	**6.00E−01±2.92E−01**

续表

函数	Avg±Std				
	CS	ACS	DACS	SACS	MSRCS
f_{17}	5.18E+01±7.77E+00	5.01E+01±6.38E+00	1.20E+02±2.45E+01	**3.05E+01±5.88E−02**	5.88E+01±8.36E+00
f_{18}	2.26E+02±4.07E+01	1.70E+02±1.80E+01	1.37E+02±2.76E+01	2.23E+02±2.29E+01	**9.50E+01±1.76E+01**
f_{19}	7.41E+00±1.50E+00	5.71E+00±2.25E+00	9.26E+00±3.70E+00	**5.50E−01±1.93E−01**	3.85E+00±1.38E+00
f_{20}	1.23E+01±5.13E−01	1.22E+01±5.97E−01	1.49E+01±3.56E−01	1.28E+01±7.48E−01	**1.12E+01±6.78E−01**
f_{21}	2.56E+02±5.07E+01	2.74E+02±8.32E+01	3.07E+02±7.11E+01	**2.27E+02±4.47E+01**	3.07E+02±9.00E+01
f_{22}	7.11E+02±2.32E+02	1.05E+03±4.27E+02	3.39E+03±7.36E+02	**4.25E+01±4.26E+01**	1.41E+03±5.72E+02
f_{23}	5.12E+03±4.14E+02	5.34E+03±5.09E+02	4.97E+03±4.53E+02	5.02E+03±4.40E+02	**4.10E+03±5.25E+02**
f_{24}	2.81E+02±1.07E+00	2.74E+02±7.19E+00	2.86E+02±6.24E+00	2.82E+02±5.16E+00	**2.61E+02±1.29E+01**
f_{25}	2.99E+02±5.08E+00	2.85E+02±8.31E+00	2.95E+02±7.75E+00	3.02E+02±6.43E+00	**2.76E+02±9.43E+00**
f_{26}	**2.00E+02±2.70E−02**	2.21E+02±5.45E+01	2.30E+02±6.68E+01	2.06E+02±3.24E+01	2.11E+02±4.15E+01
f_{27}	1.06E+03±1.83E+02	1.03E+03±4.80E+01	1.14E+03±5.56E+01	**9.40E+02±3.02E+02**	9.52E+02±7.33E+01
f_{28}	2.96E+02±2.22E+01	3.38E+02±2.06E+02	3.00E+02±2.10E−13	**2.89E+02±4.12E+01**	3.00E+02±1.17E−13
rank	5.071 4	4.571 4	6.392 9	5.285 7	**3.321 4**
+/≈/−	16/5/7	16/4/8	20/5/3	14/6/8	

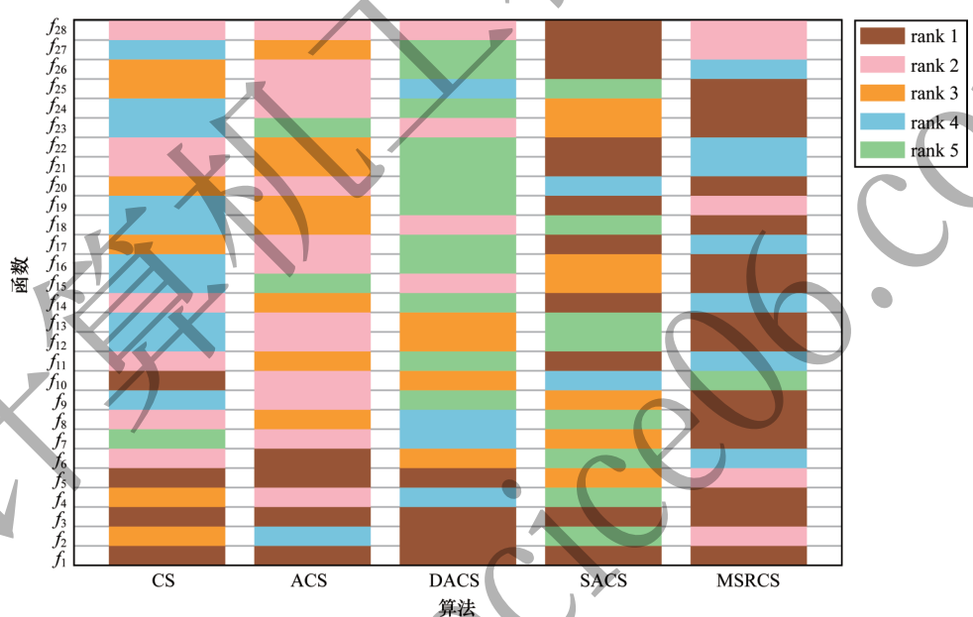
表3 d=30时GBCS、GCS、NACS、SDCS、MSRCS算法的CEC2013测试函数实验结果

Table 3 Experimental results of CEC2013 test functions of GBCS, GCS, NACS, SDCS, MSRCS algorithms when d=30

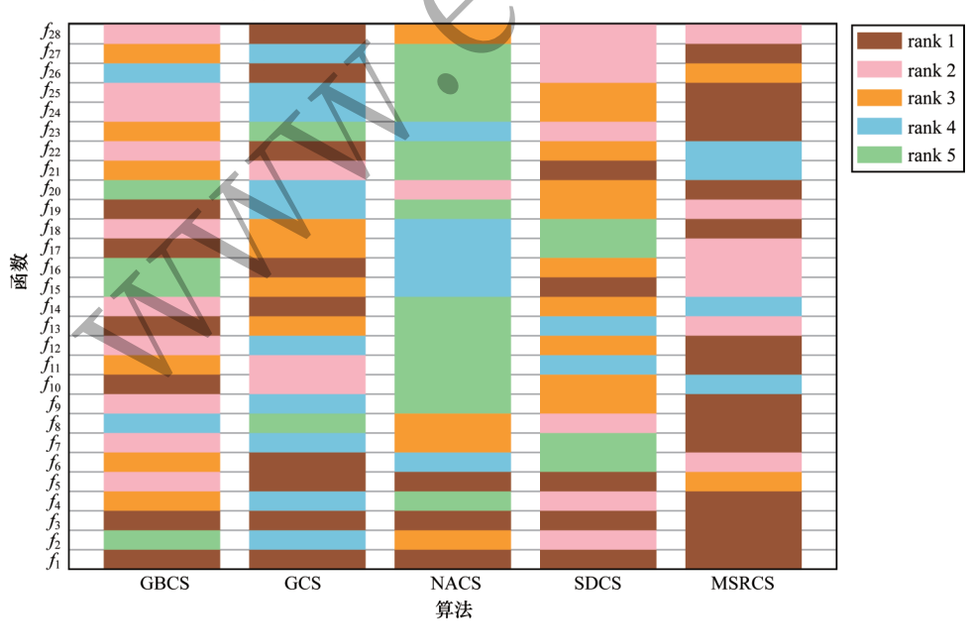
函数	Avg±Std				
	GBCS	GCS	NACS	SDCS	MSRCS
f_1	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**
f_2	1.13E+06±6.13E+05	2.96E+05±1.13E+05	7.87E+04±3.97E+04	**3.91E+04±2.17E+04**	4.38E+04±1.70E+04
f_3	**1.00E+10±0.00E+00**	**1.00E+10±0.00E+00**	**1.00E+10±0.00E+00**	**1.00E+10±0.00E+00**	**1.00E+10±0.00E+00**
f_4	3.20E+01±1.73E+01	9.27E+02±5.50E+02	5.78E+03±4.22E+03	1.81E+01±2.00E+01	**7.49E−01±6.91E−01**
f_5	1.52E−14±3.86E−14	**0.00E+00±0.00E+00**	3.79E−15±2.04E−14	1.14E−14±3.41E−14	5.31E−14±5.67E−14
f_6	**3.66E+00±8.95E+00**	3.79E+00±8.15E+00	6.20E+00±4.51E+00	1.54E+01±6.93E+00	4.66E+00±9.73E+00
f_7	5.12E+01±1.63E+01	9.63E+01±1.50E+01	1.01E+02±2.43E+01	1.14E+02±2.64E+01	**3.93E+01±1.65E+01**
f_8	2.09E+01±4.49E−02	2.09E+01±3.75E−02	2.09E+01±4.87E−02	2.09E+01±4.12E−02	**2.08E+01±9.74E−02**
f_9	2.70E+01±1.79E+00	2.89E+01±2.04E+00	3.01E+01±2.36E+00	2.63E+01±2.98E+00	**2.59E+01±3.14E+00**
f_{10}	3.74E−02±1.97E−02	**3.33E−02±2.27E−02**	3.20E−01±1.22E−01	1.07E−01±4.96E−02	1.10E−01±5.38E−02
f_{11}	2.43E+01±7.69E+00	2.76E+01±1.31E+01	**3.24E−01±1.33E−01**	2.84E+01±1.06E+01	2.16E+01±5.91E+00
f_{12}	**7.18E+01±1.44E+01**	1.60E+02±3.21E+01	1.77E+02±4.68E+01	1.55E+02±4.34E+01	7.45E+01±2.09E+01
f_{13}	**1.15E+02±2.12E+01**	2.05E+02±3.03E+01	2.51E+02±5.00E+01	2.26E+02±4.68E+01	1.25E+02±2.14E+01
f_{14}	9.50E+02±3.55E+02	**5.90E+02±2.54E+02**	1.94E+03±5.58E+02	1.16E+03±3.28E+02	1.52E+03±4.00E+02
f_{15}	4.59E+03±3.54E+02	3.77E+03±3.20E+02	4.01E+03±6.07E+02	**3.52E+03±4.64E+02**	3.78E+03±5.31E+02
f_{16}	1.45E+00±2.07E−01	2.67E−01±5.03E−02	1.20E+00±2.86E−01	1.10E+00±3.37E−01	**6.00E−01±2.92E−01**
f_{17}	**4.92E+01±6.50E+00**	6.60E+01±1.48E+01	1.18E+02±3.34E+01	1.31E+02±3.58E+01	5.88E+01±8.36E+00
f_{18}	1.32E+02±1.44E+01	1.51E+02±2.48E+01	1.86E+02±3.80E+01	2.67E+02±6.73E+01	**9.50E+01±1.76E+01**
f_{19}	**3.77E+00±1.16E+00**	7.07E+00±2.02E+00	1.66E+01±8.32E+00	5.90E+00±2.66E+00	3.85E+00±1.38E+00
f_{20}	1.26E+01±1.07E+00	1.20E+01±4.97E−01	1.20E+01±7.34E−01	1.20E+01±5.56E−01	**1.12E+01±6.78E−01**
f_{21}	3.28E+02±9.00E+01	2.59E+02±5.93E+01	3.44E+02±9.11E+01	**2.48E+02±7.52E+01**	3.07E+02±9.00E+01
f_{22}	**1.07E+03±2.62E+02**	9.47E+02±4.03E+02	2.32E+03±6.82E+02	1.34E+03±3.96E+02	1.41E+03±5.72E+02
f_{23}	4.73E+03±3.66E+02	4.63E+03±4.82E+02	4.77E+03±4.99E+02	4.43E+03±5.86E+02	**4.10E+03±5.25E+02**
f_{24}	2.64E+02±1.10E+01	2.80E+02±1.13E+01	2.82E+02±1.00E+01	2.72E+02±7.82E+00	**2.61E+02±1.29E+01**
f_{25}	2.77E+02±8.03E+00	2.99E+02±6.50E+00	2.99E+02±6.24E+00	2.83E+02±6.13E+00	**2.76E+02±9.43E+00**
f_{26}	2.12E+02±4.16E+01	2.00E+02±5.62E−03	2.24E+02±6.01E+01	**2.00E+02±3.08E−03**	2.11E+02±4.15E+01
f_{27}	9.55E+02±8.74E+01	1.09E+03±4.74E+01	1.07E+03±6.86E+01	1.02E+03±6.13E+01	**9.52E+02±7.33E+01**
f_{28}	3.00E+02±2.56E−13	3.00E+02±4.15E−14	6.20E+02±7.19E+02	3.00E+02±1.63E−13	**3.00E+02±1.17E−13**
rank	4.357 1	4.500 0	6.714 3	4.750 0	**3.321 4**
+/≈/−	12/10/6	15/7/6	23/3/2	16/8/4	

从表2可以看出,MSRCS在f_4、f_7、f_8、f_9、f_{12}、f_{15}、f_{16}、f_{18}、f_{20}、f_{23}、f_{24}、f_{25}上有较好的结果。在单峰函数中,MSRCS在f_4上效果明显优于其他算法,在f_2上仅劣于DACS,在f_1、f_3上各算法的性能趋于一致。在多峰函数f_7、f_8、f_9、f_{12}、f_{15}、f_{16}、f_{18}、f_{20}上的结果优于其他算法,在f_{19}上仅劣于SACS。在组合函数f_{23}、f_{24}、f_{25}上MSRCS的结果较好。通过表2和表3底部的Friedman检验与Wilcoxon秩和检验的结果可以看出,MSRCS在28个测试函数上的平均排名为3.321 4,排名第一,相比于GBCS有12个函数更优,相比于其他算法至少有14个函数更优。可见,MSRCS在处理不同类型的问题时均具有更好的稳定性和收敛性。

为了更直观地显示表2和表3结果的排名情况,整理总结每个函数的平均误差值(最小化问题)排名,绘制了基于排名统计情况的堆叠直方图。如图4所示,每个排名用一个彩色块表示,从第一到第五分别为棕色、粉色、橘色、蓝色和绿色,含棕色块越多的算法性能越强,反之绿色块越多算法性能越差,彩色效果见《计算机工程》官网HTML版。图4(a)表示CS、ACS、DACS、SACS和MSRCS的排名情况,图4(b)表示GBCS、GCS、NACS、SDCS和MSRCS的排名情况。由图4可知,排名第一的棕色块在MSRCS中的比例最多,说明MSRCS与其他算法相比表现最佳。



(a)CS、ACS、DACS、SACS和MSRCS算法的排名



(b)GBCS、GCS、NACS、SDCS和MSRCS算法的排名

图4　在CEC2013测试函数上各算法的排名堆叠直方图

Fig.4　Stacked histograms of the ranks of the algorithms on CEC2013 test function

为进一步说明MSRCS的收敛能力和寻优精度，选取原始CS以及4种具有代表性的CS改进算法，分别从CEC2013测试集的单峰函数、多峰函数和组合函数中挑选具有代表性的f_5、f_9、f_{13}、f_{20}、f_{24}、f_{25}函数，其中f_5为单峰函数，f_9、f_{13}、f_{20}为多峰函数，f_{24}、f_{25}为组合函数，并在图5中绘制了其收敛曲线。从图5(a)中可以看出，MSRCS在单峰函数f_5的整个迭代过程中收敛速度相比于其他算法是最快的，在18 000次迭代时已经收敛到了最优值。这证明了MSRCS中的CS-best解更新方法在单峰函数问题上有利于算法更快地收敛于最优解。在f_{13}、f_{24}、f_{25}函数中，CS、ACS、DACS、SACS、GCS都存在着陷入局部极值从而停止收敛的情况，在其他算法都陷入局部极值时，MSRCS跳出局部最优达到了更高的精度值，说明MSRCS算法中的CS-rand解更新方法尝试跳出局部极值并寻找更优解。对于f_9、f_{20}函数，MSRCS优势较为明显，由于其根据概率选择不同调和策略来平衡勘探与开采，因此收敛曲线并非阶段性断层而是呈逐步寻优状，以靠近全局最优解。根据上述分析，MSRCS的调和策略在各种类型的测试函数上均具有较好的收敛速度、较强的寻优能力以及跳出局部极值的能力。

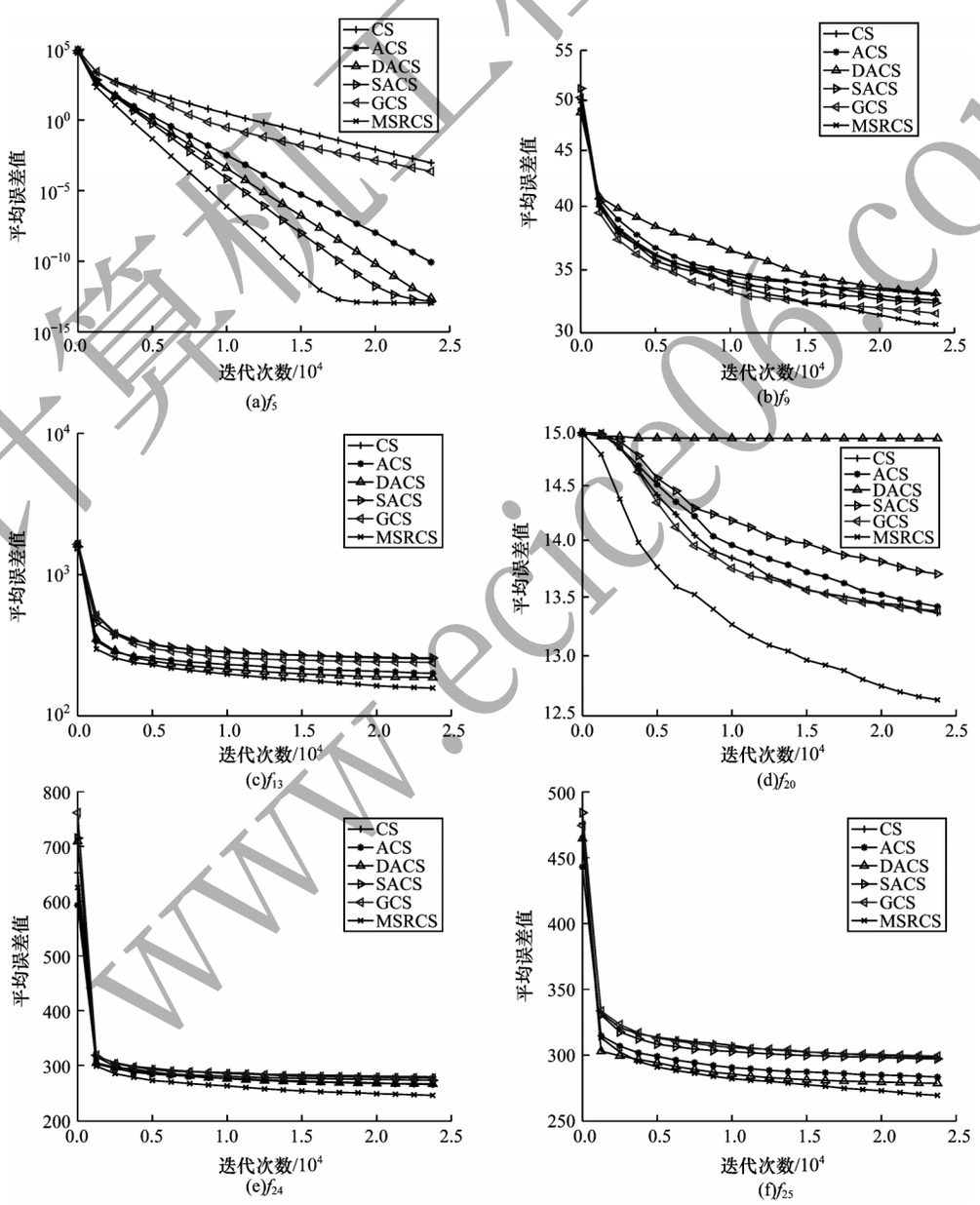


图5 CS、ACS、DACS、SACS、GCS、MSRCS算法在部分测试函数上的收敛曲线

Fig.5 Convergence curves of CS, ACS, DACS, SACS, GCS, MSRCS algorithms on the sectional test functions

3.3 多策略调和的有效性分析

为了验证多策略调和的有效性，本文提出了MSRCS-1、MSRCS-2、MSRCS-3这3种MSRCS改进算法，分别为只使用自适应步长、不使用线性递减规则(将

迭代过程分为前期和后期）以及单独使用调和策略公式，测试结果如表4所示，实验参数设置与3.1节一致。从表4可以看出：在Wilcoxon检验结果中，由于MSRCS分别有12、14、19个函数比MSRCS-1、MSRCS-2和MSRCS-3更好，因此MSRCS中的调和策略应用在测试函数上的效果相比于其他单独使用策略的改进算法更好；在Friedman检验的平均排名（rank）中，数据显示MSRCS为1.982 1，MSRCS-1为2.303 6，MSRCS-2为2.678 6，MSRCS-3为3.035 7。可见，在多策略调和的作用下MSRCS效果最佳。

在此基础上，进一步对多策略调和MSRCS的收敛性及稳定性进行验证。图6给出了MSRCS和3种改进算法在函数f_{11}、f_{12}、f_{13}、f_{14}上的收敛曲线图。由图6可知，含多策略调和的MSRCS在解的精度方面要优于其他改进算法，且在其他改进算法都陷入局部极值时能够达到更高的精度，由此证明了MSRCS的多策略调和跳出局部极值和持续寻优的能力较好。此外，还证明了多策略调和相比于其他改进算法的单策略更加有效。

表4　d=30时MSRCS算法在CEC2013测试函数上单独使用不同策略的实验结果

Table 4　Experimental results of MSRCS algorithm by using different strategy on the CEC2013 test functions when d=30

函数	Avg±Std			
	MSRCS-1	MSRCS-2	MSRCS-3	MSRCS
f_1	**0.00E+00±0.00E+00**	**0.00E+00±0.00E+00**	**2.27E−13±0.00E+00**	**0.00E+00±0.00E+00**
f_2	8.69E+05±4.49E+05	9.64E+05±5.70E+05	4.84E+04±3.52E+04	**4.38E+04±1.70E+04**
f_3	**1.00E+10±0.00E+00**	**1.00E+10±0.00E+00**	**1.00E+10±0.00E+00**	**1.00E+10±0.00E+00**
f_4	1.30E+02±5.86E+01	7.88E+00±4.90E+00	**4.77E−04±6.46E−04**	7.49E−01±6.91E−01
f_5	**7.58E−15±2.84E−14**	9.85E−14±3.86E−14	3.33E−13±5.82E−14	5.31E−14±5.67E−14
f_6	**1.41E+00±5.43E+00**	6.05E+00±1.02E+01	2.15E+01±2.27E+01	4.66E+00±9.73E+00
f_7	5.64E+01±1.80E+01	4.35E+01±1.60E+01	6.87E+01±3.20E+01	**3.93E+01±1.65E+01**
f_8	2.09E+01±4.82E−02	2.09E+01±3.37E−02	**2.08E+01±8.60E−02**	2.08E+01±9.74E−02
f_9	2.68E+01±1.87E+00	2.68E+01±1.36E+00	2.79E+01±2.19E+00	**2.59E+01±3.14E+00**
f_{10}	**4.09E−02±2.30E−02**	1.12E−01±8.84E−02	3.08E−01±2.06E−01	1.10E−01±5.38E−02
f_{11}	**1.77E+01±6.74E+00**	1.85E+01±5.83E+00	9.04E+01±4.09E+01	2.16E+01±5.91E+00
f_{12}	**7.02E+01±1.25E+01**	7.34E+01±1.71E+01	1.09E+02±4.18E+01	7.45E+01±2.09E+01
f_{13}	**1.20E+02±2.52E+01**	1.27E+02±2.18E+01	1.61E+02±3.29E+01	1.25E+02±2.14E+01
f_{14}	5.97E+02±1.88E+02	**6.31E+02±2.86E+02**	1.86E+03±5.52E+02	1.52E+03±4.00E+02
f_{15}	4.41E+03±4.78E+02	4.55E+03±3.60E+02	**3.53E+03±3.94E+02**	3.78E+03±5.31E+02
f_{16}	1.51E+00±2.23E−01	1.48E+00±2.24E−01	7.67E−01±3.41E−01	**6.00E−01±2.92E−01**
f_{17}	**4.77E+01±7.23E+00**	4.78E+01±6.85E+00	8.37E+01±1.91E+01	5.88E+01±8.36E+00
f_{18}	1.39E+02±1.78E+01	1.35E+02±1.25E+01	1.18E+02±2.94E+01	**9.50E+01±1.76E+01**
f_{19}	**3.63E+00±1.22E+00**	3.89E+00±1.37E+00	2.97E+01±5.53E+01	3.85E+00±1.38E+00
f_{20}	1.16E+01±3.57E−01	1.16E+01±3.85E−01	1.33E+01±1.53E+00	**1.12E+01±6.78E−01**
f_{21}	**2.78E+02±5.37E+01**	3.13E+02±8.14E+01	3.13E+02±8.14E+01	3.07E+02±9.00E+01
f_{22}	7.42E+02±2.46E+02	**7.94E+02±3.20E+02**	2.11E+03±6.16E+02	1.41E+03±5.72E+02
f_{23}	4.98E+03±3.88E+02	4.82E+03±5.30E+02	**3.87E+03±5.36E+02**	4.10E+03±5.25E+02
f_{24}	2.71E+02±9.51E+00	**2.60E+02±1.07E+01**	2.67E+02±9.84E+00	2.61E+02±1.29E+01
f_{25}	2.85E+02±5.19E+00	2.80E+02±8.37E+00	2.82E+02±1.25E+01	**2.76E+02±9.43E+00**
f_{26}	**2.00E+02±2.03E−02**	2.11E+02±3.97E+01	3.06E+02±7.55E+01	2.11E+02±4.15E+01
f_{27}	9.72E+02±1.22E+01	9.79E+02±6.35E+01	9.52E+02±9.74E+01	**9.52E+02±7.33E+01**
f_{28}	3.00E+02±1.99E−13	3.00E+02±2.92E−13	5.44E+02±5.31E+02	**3.00E+02±1.17E−13**
rank	2.303 6	2.678 6	3.035 7	**1.982 1**
+/≈/−	12/9/7	14/12/2	19/8/1	

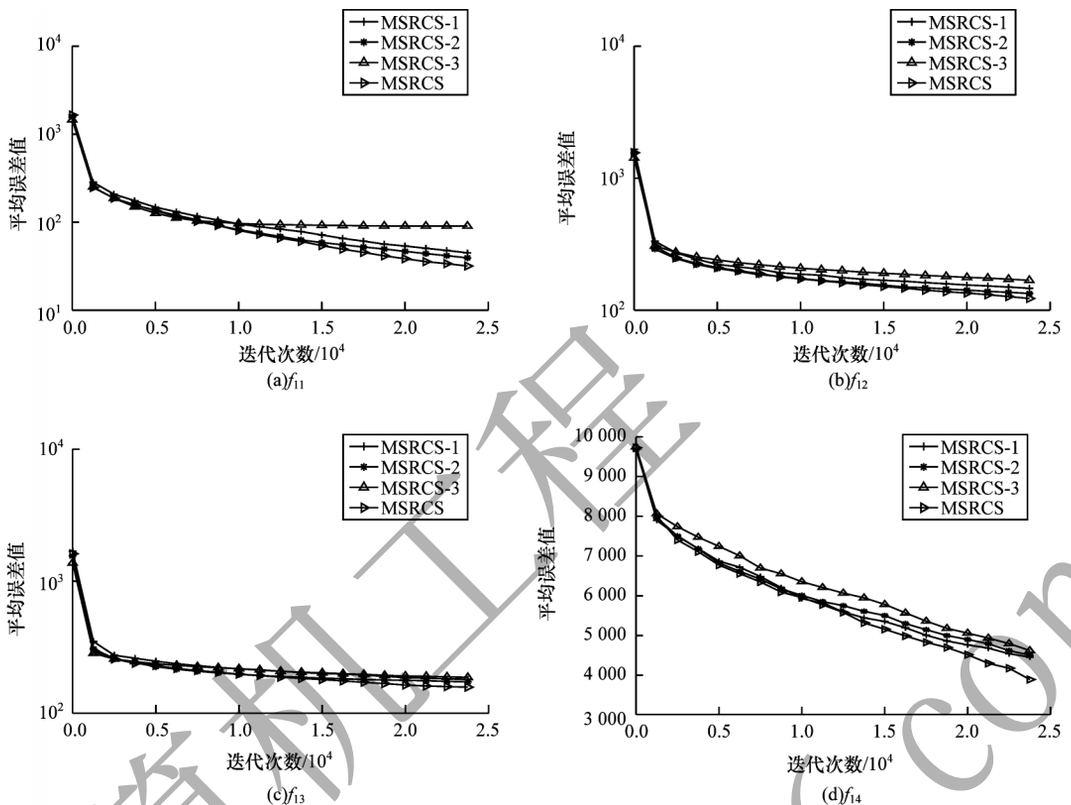


图6 MSRCS-1、MSRCS-2、MSRCS-3、MSRCS算法在f_{11}、f_{12}、f_{13}、f_{14}函数上的收敛曲线

Fig.6 Convergence curves of MSRCS-1, MSRCS-2, MSRCS-3, MSRCS algorithms on f_{11}, f_{12}, f_{13}, f_{14} functions

3.4 与群智能优化算法的比较

差分进化(Differential Evolution, DE)算法[25]、人工蜂群(ABC)算法[3]和萤火虫算法(FA)[4]是被广泛研究与使用的群智能优化算法。为进一步验证MSRCS的性能,选取DE、ABC和FA的改进算法进行比较,分别为CUDE[26]、MEABC[27]和NSRaFA[28]。

鉴于实验的公平性,对这些算法设置种群数量n为20、问题维度d为30及评价次数F_{max}为1E6,且每个测试函数独立运行30次,CUDE、MEABC和NSRaFA参数遵循相应文献中的设置,MSRCS参数设置与表1一致。CEC2013测试集的实验结果如表5所示,其中加粗的字体表示最好结果。从表5可以看出:MSRCS有11个函数优于其他算法;对于单峰函数f_2、f_3、f_4,CUDE相比于其他3种算法具有更好的结果,MSRCS在f_1上找到了最优值;在多峰函数f_7、f_8、f_{10}、f_{12}、f_{13}、f_{16}、f_{18}上MSRCS有相对优势,说明其在解决多峰函数问题时有着比其他算法更好的效果;MEABC在组合函数f_{21}、f_{22}、f_{26}、f_{28}上的结果较好;在Wilcoxon值和检验结果中,MSRCS相比于其他3种改进算法,至少有12个函数有更优结果。可见,MSRCS在解决单峰和多峰问题时相对于其他算法具有更强的寻优能力。

图7给出了CUDE、MEABC、NSRaFA和MSRCS在f_1、f_5、f_6、f_7、f_{10}、f_{28}函数上的收敛曲线图,其中包含单峰函数、多峰函数以及组合函数。

表5 d=30时CUDE、MEABC、NSRaFA、MSRCS算法在CEC2013测试函数上的实验结果

Table 5 Experimental results of CUDE, MEABC, NSRaFA, MSRCS algorithms on the CEC2013 test functions when d=30

函数	Avg±Std			
	CUDE	MEABC	NSRaFA	MSRCS
f_1	3.79E−13±2.06E−13	5.31E−13±1.22E−13	2.17E+00±4.81E−01	**0.00E+00±0.00E+00**
f_2	**4.83E+03±3.72E+03**	3.84E+06±1.05E+06	1.78E+07±6.28E+06	4.38E+04±1.70E+04
f_3	**7.66E+06±1.51E+07**	2.44E+08±1.64E+08	2.17E+09±2.50E+09	1.00E+10±0.00E+00
f_4	**1.08E−11±1.03E−11**	1.02E+05±1.55E+04	1.47E+03±5.18E+02	7.49E−01±6.91E−01
f_5	6.40E−13±4.93E−13	7.31E−13±5.24E−13	1.56E+00±4.10E−01	**5.31E−14±5.67E−14**
f_6	**1.37E+00±4.71E+00**	1.17E+01±5.54E+00	6.54E+01±3.77E+01	4.66E+00±9.73E+00
f_7	6.25E+01±1.83E+01	1.12E+02±1.90E+01	9.12E+01±2.82E+01	**3.93E+01±1.65E+01**
f_8	2.09E+01±5.91E−02	2.09E+01±4.79E−02	2.09E+01±5.12E−02	**2.08E+01±9.74E−02**
f_9	**2.42E+01±4.01E+00**	3.18E+01±1.77E+00	2.62E+01±3.22E+00	2.59E+01±3.14E+00
f_{10}	1.48E−01±7.36E−02	1.19E−01±3.46E−02	1.15E+01±4.16E+00	**1.10E−01±5.38E−02**

续表

函数	Avg±Std			
	CUDE	MEABC	NSRaFA	MSRCS
f_{11}	2.73E+01±1.50E+01	**3.65E−01±5.44E−01**	9.45E+01±2.49E+01	2.16E+01±5.91E+00
f_{12}	8.79E+01±2.33E+01	2.09E+02±4.48E+01	1.10E+02±3.43E+01	**7.45E+01±2.09E+01**
f_{13}	1.51E+02±3.10E+01	2.62E+02±2.74E+01	1.66E+02±3.12E+01	**1.25E+02±2.14E+01**
f_{14}	1.14E+02±1.25E+02	**1.51E+01±3.00E+01**	3.36E+03±6.94E+02	1.52E+03±4.00E+02
f_{15}	**3.71E+03±7.99E+02**	3.96E+03±4.04E+02	4.04E+03±6.60E+02	3.78E+03±5.31E+02
f_{16}	1.25E+00±5.17E−01	8.56E−01±1.56E−01	2.21E+00±2.36E−01	**6.00E−01±2.92E−01**
f_{17}	3.61E+01±4.89E+00	**3.07E+01±2.74E−01**	1.66E+02±2.14E+01	5.88E+01±8.36E+00
f_{18}	1.18E+02±3.07E+01	2.01E+02±2.43E+01	1.76E+02±2.20E+01	**9.50E+01±1.76E+01**
f_{19}	9.06E+00±5.97E+00	**2.14E−01±8.17E−02**	1.44E+01±2.66E+00	3.85E+00±1.38E+00
f_{20}	**1.04E+01±7.17E−01**	1.47E+01±2.36E−01	1.50E+01±1.27E−06	1.12E+01±6.78E−01
f_{21}	2.96E+02±7.12E+01	**2.02E+02±4.43E+01**	3.70E+02±6.63E+01	3.07E+02±9.00E+01
f_{22}	1.55E+02±6.52E+01	**3.35E+01±4.57E+01**	3.45E+03±6.85E+02	1.41E+03±5.72E+02
f_{23}	4.37E+03±8.58E+02	4.97E+03±4.23E+02	4.85E+03±8.77E+02	**4.10E+03±5.25E+02**
f_{24}	**2.45E+02±1.07E+01**	2.89E+02±8.16E+00	2.82E+02±7.76E+00	2.61E+02±1.29E+01
f_{25}	2.81E+02±1.07E+01	3.10E+02±7.03E+00	2.85E+02±8.40E+00	**2.76E+02±9.43E+00**
f_{26}	2.47E+02±6.66E+01	**2.00E+02±1.25E−01**	2.10E+02±3.68E+01	2.11E+02±4.15E+01
f_{27}	**7.77E+02±9.39E+01**	8.29E+02±3.76E+02	9.85E+02±1.06E+02	9.52E+02±7.33E+01
f_{28}	2.93E+02±3.59E+01	**2.87E+02±4.88E+01**	4.86E+02±5.74E+02	3.00E+02±1.17E−13
+/≈/−	12/5/11	17/3/8	23/3/2	

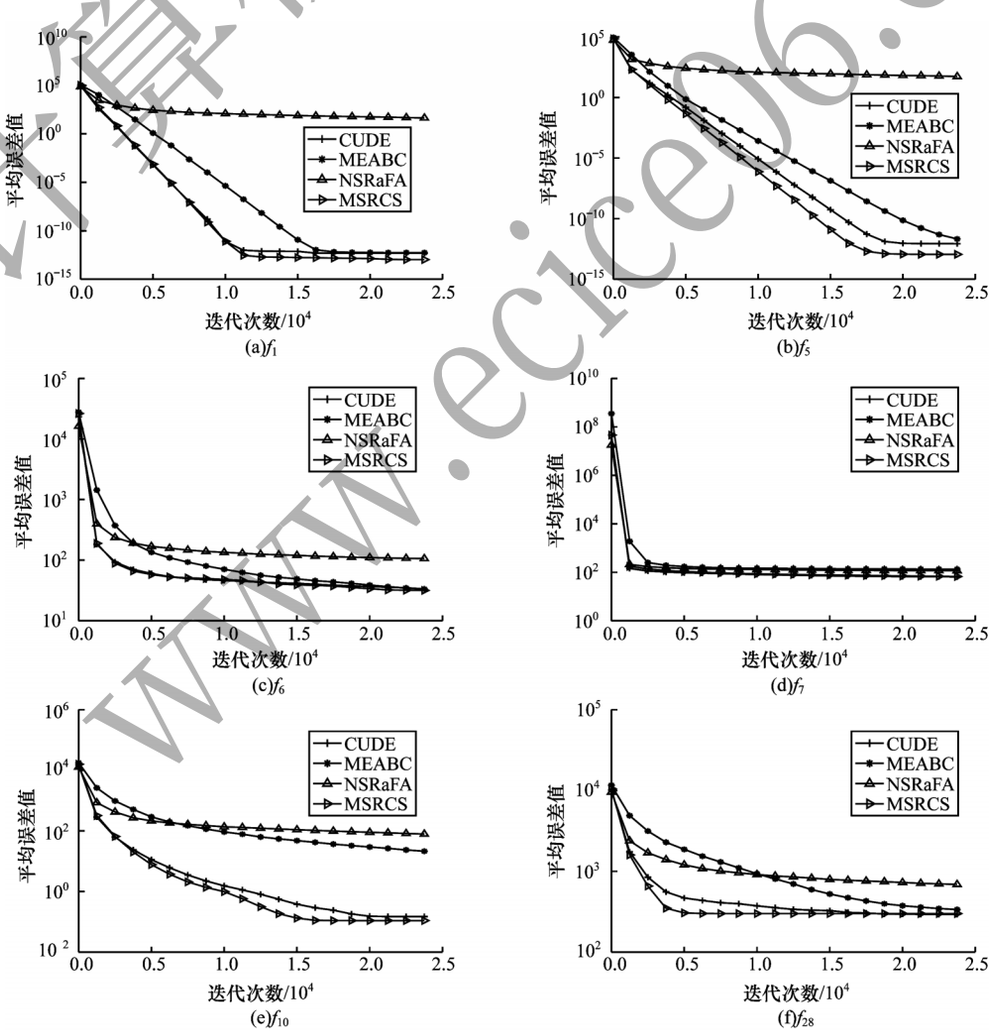


图7　CUDE、MEABC、NSRaFA、MSRCS算法在f_1、f_5、f_6、f_7、f_{10}、f_{28}函数上的收敛曲线

Fig.7　Convergence curves of CUDE, MEABC, NSRaFA, MSRCS algorithms on f_1, f_5, f_6, f_7, f_{10}, f_{28} functions

从图7可以看出，在f_1收敛图中，MSRCS前期寻找到的最优解与CUDE相差不大，但是在后期MSRCS寻找到了比CUDE更优的最优解，由此可见MSRCS中的CS-rand解更新方法跳出局部极值到达了更高的精度。MSRCS除了在解的精度方面较其他改进算法更优外，在整个迭代过程中的收敛速度也相比于其他算法更快。综上所述，MSRCS相比于其他算法有着相对更好的收敛能力、寻优能力以及跳出局部极值的能力。

3.5 自适应参数分析

布谷鸟步长控制因子的大小决定其搜索范围。步长控制因子越大，搜索范围越广，算法的全局搜索能力和种群多样性增强；反之，步长越小，算法的局部搜索能力和收敛性增强。在改进的自适应步长式(6)中，调和因子η控制布谷鸟的搜索范围大小，因此η的大小决定步长的变化范围。

为了测试自适应步长对MSRCS性能的提升效果，本文对参数η进行基于CEC2013测试集的Friedman检验并筛选出效果最好的η值。实验选择无自适应、η=0.3、η=0.5和η=0.7这4种情况，得到MSRCS在CEC2013测试集的28个函数中的平均排名结果如表6所示。从表6可以看出，无自适应的排名为3.160 7、η=0.3时的排名为2.625 0、η=0.5时的排名为1.964 3、η=0.7时的排名为2.250 0。由于当η=0.5时对MSRCS的性能提升最明显，因此选取η=0.5，自适应步长的变化范围为[0.00,0.25]。

表6 不同η值的MSRCS平均排名

Table 6 Average rank of MSRCS with the different η value

η值	rank
无自适应	3.160 7
0.3	2.625 0
0.5	1.964 3
0.7	2.250 0

3.6 MSRCS计算时间分析

根据MSRCS算法流程可以看出，MSRCS仅改变了算法对策略的选择机制，有概率选择新的解更新方法代替Lévy飞行，减少了对Lévy飞行随机步长的计算。使用单一的Lévy飞行成效有限，对于平衡算法的勘探与开采，多策略调和对每个时期的策略选择相对更加有效。

MSRCS和CS在不同维度下计算CEC2013测试集上28个测试函数的运行时间，如表7所示。从表7可以看出，无论是在问题维度d为30、50或100时，CS和MSRCS的运行时间并无显著差异。由此再次验证了CS和MSRCS的复杂度分析，MSRCS和CS在计算复杂度上并无显著差异。

表7 CS和MSRCS算法在CEC2013测试集上的运行时间

Table 7 Runtime of CS and MSRCS algorithms on CEC2013 test set 10^3 s

问题维度d	CS	MSRCS
30	1.06	1.06
50	1.99	1.95
100	4.52	4.45

4 结束语

本文提出一种多策略调和的布谷鸟搜索(MSRCS)算法，在多策略框架下根据迭代阶段的不同需求进行策略选择，有效加快算法的收敛速度以及丰富种群学习的多样性。但由于调和策略基于随机思想，无法准确感知布谷鸟的状态并做出决策，因此MSRCS算法对CEC2013测试集的28个函数进行实验和统计分析，并与原始CS和其7种改进算法以及3种经典群智能优化算法进行比较，还对自适应步长参数以及策略有效性进行研究，结果表明MSRCS算法性能优于对比算法，同时证明了调和策略的可操作性。后续可将MSRCS算法应用于车间调度、旅行商等复杂组合优化问题，进一步扩大其适用范围。

参考文献

[1] DORIGO M, BIRATTARI M, STUTZLE T. Ant colony optimization[J]. IEEE Computational Intelligence Magazine, 2006, 1(4): 28-39.

[2] KENNEDY J, EBERHART R. Particle swarm optimization [C]//Proceedings of International Conference on Neural Networks. Washington D. C., USA: IEEE Press, 1995: 1942-1948.

[3] KARABOGA D. Artificial bee colony algorithm [J]. Scholarpedia, 2010, 5(3): 6915.

[4] YANG X S. Firefly algorithm, stochastic test functions and design optimization [J]. International Journal of Bio-Inspired Computation, 2010, 2(2): 78.

[5] YANG X S, DEB S. Cuckoo search via Lévy flights[C]// Proceedings of World Congress on Nature & Biologically Inspired Computing. Washington D. C., USA: IEEE Press, 2009: 210-214.

[6] 石优，林琳，吴岩. 基于布谷鸟搜索的模糊PID拥塞控制方法[J]. 计算机工程，2020，46(11)：238-245.
SHI Y, LIN L, WU Y. Fuzzy PID congestion control method based on Cuckoo search[J]. Computer Engineering, 2020, 46(11): 238-245. (in Chinese)

[7] 赵博颖，肖鹏，张力. 基于混合并行布谷鸟搜索的作业调度算法[J]. 计算机工程与设计，2019，40(3)：719-724.
ZHAO B Y, XIAO P, ZHANG L. Job scheduling algorithm based on hybrid parallel Cuckoo search [J]. Computer Engineering and Design, 2019, 40(3): 719-724. (in Chinese)

[8] 邓小亚. 基于复合布谷鸟算法的彩色图像多阈值分割[J]. 计算机与数字工程,2019,47(4):944-948.
DENG X Y. Multi threshold image segmentation based on hybrid cuckoo algorithm[J]. Computer & Digital Engineering, 2019,47(4):944-948. (in Chinese)

[9] RAKHSHANI H,RAHATI A. Snap-drift cuckoo search: a novel cuckoo search optimization algorithm[J]. Applied Soft Computing,2017,52:771-794.

[10] PENG H,DENG C S,WANG H,et al. Gaussian bare-bones cuckoo search algorithm[C]//Proceedings of the Genetic and Evolutionary Computation Conference Companion. New York,USA:ACM Press,2018:93-94.

[11] PENG H,ZENG Z G,DENG C S,et al. Multi-strategy serial cuckoo search algorithm for global optimization [J]. Knowledge-Based Systems,2021,214:106729.

[12] LI X T,YIN M H. Modified cuckoo search algorithm with self adaptive parameter method[J]. Information Sciences, 2015,298:80-97.

[13] 刘景森,刘晓珍,李煜. 具有动态步长和发现概率的布谷鸟搜索算法[J]. 系统仿真学报,2020,32(2):289-298.
LIU J S,LIU X Z,LI Y. Cuckoo search algorithm with dynamic step and discovery probability[J]. Journal of System Simulation,2020,32(2):289-298. (in Chinese)

[14] 周诗源,王英林. 基于布谷鸟搜索优化算法的多文档摘要方法[J]. 计算机工程,2020,46(7):58-64,71.
ZHOU S Y,WANG Y L. Multiple document summarization method based on optimized Cuckoo search algorithm[J]. Computer Engineering,2020,46(7):58-64,71. (in Chinese)

[15] 向庭立,王红军,史英春. 基于布谷鸟搜索的混合传感器网络覆盖优化策略[J]. 计算机工程,2019,45(12):79-85.
XIANG T L,WANG H J,SHI Y C. Hybrid sensor network coverage optimization strategy based on Cuckoo search[J]. Computer Engineering,2019,45(12):79-85. (in Chinese)

[16] AGARWAL M,SRIVASTAVA G M S. A Cuckoo search algorithm-based task scheduling in cloud computing[M]. Berlin,Germany:Springer,2018.

[17] 张永韡,汪镭,吴启迪. 动态适应布谷鸟搜索算法[J]. 控制与决策,2014,29(4):617-622.
ZHANG Y W,WANG L,WU Q D. Dynamic adaptation Cuckoo search algorithm[J]. Control and Decision,2014, 29(4):617-622. (in Chinese)

[18] GAO S Z,GAO Y,ZHANG Y M,et al. Multi-strategy adaptive cuckoo search algorithm[J]. IEEE Access,2019, 7:137642-137655.

[19] LIU J N,PENG H,WU Z J,et al. Multi-strategy brain storm optimization algorithm with dynamic parameters adjustment [J]. Applied Intelligence,2020,50(4):1289-1315.

[20] 彭虎,吴志健,周新宇,等. 基于精英区域学习的动态差分进化算法[J]. 电子学报,2014,42(8):1522-1530.
PENG H,WU Z J,ZHOU X Y,et al. Dynamic differential evolution algorithm based on elite local learning[J]. Acta Electronica Sinica,2014,42(8):1522-1530. (in Chinese)

[21] LIANG J J,QU B Y,SUGANTHAN P N,et al. Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization[EB/OL]. [2021-04-08]. https://www. researchgate. net/publication/256995189_Problem_Definitions_and_Evaluation_Criteria_for_the_CEC_2013_Special_Session_on_Real-Parameter_Optimization.

[22] NAIK M,NATH M R,WUNNAVA A,et al. A new adaptive Cuckoo search algorithm[C]//Proceedings of the 2nd International Conference on Recent Trends in Information Systems. Washington D. C.,USA:IEEE Press, 2015:1-5.

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

[24] CHENG J T,WANG L. Cuckoo search algorithm with neighborhood attraction for numerical optimization[J]. IEEE Access,2019,7:122261-122274.

[25] STORN R,PRICE K. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces[J]. Journal of Global Optimization,1997,11(4): 341-359.

[26] PENG H,WU Z J,DENG C S. Enhancing differential evolution with commensal learning and uniform local search [J]. Chinese Journal of Electronics,2017,26(4):725-733.

[27] WANG H,WU Z J,RAHNAMAYAN S,et al. Multi-strategy ensemble artificial bee colony algorithm [J]. Information Sciences,2014,279:587-603.

[28] WANG H,CUI Z H,SUN H,et al. Randomly attracted firefly algorithm with neighborhood search and dynamic parameter adjustment mechanism[J]. Soft Computing, 2017,21(18):5325-5339.

编辑　陆燕菲