控制与决策
Control and Decision

基于解空间反向跳跃和信息交互强化的新型混合蛙跳算法

申晓宁， 黄遥， 游璇， 王谦

引用本文:
申晓宁, 黄遥, 游璇, 等. 基于解空间反向跳跃和信息交互强化的新型混合蛙跳算法[J]. 控制与决策, 2021, 36(1): 105−114.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2019.0719

基于解空间反向跳跃和信息交互强化的新型混合蛙跳算法

申晓宁[1,2,3†], 黄 遥[1], 游 璇[1], 王 谦[1]

(1. 南京信息工程大学 自动化学院，南京 210044；2. 南京信息工程大学 江苏省大气环境与装备技术协同创新中心，南京 210044；3. 南京信息工程大学 江苏省大数据分析技术重点实验室，南京 210044)

摘 要: 种群多样性和信息交互的深度与方式对混合蛙跳算法的爬山能力、探索能力和开发能力有着深远影响. 针对混合蛙跳算法易于陷入局部最优、收敛速度慢和寻优精度差等缺点,提出一种基于解空间反向跳跃和信息交互强化的新型混合蛙跳算法. 首先,增加子群次优解与次劣解的信息交互,促进子群内部信息的利用,引入反向跳跃思想改进局部更新机制,降低迭代后期劣解产生概率,提升空间开发能力;然后,借鉴2-opt方法实现局部最优解变异,增加子群的多样性;最后,采用各局部最优解交叉的方式加深子群间的交互深度,同时利用反向跳跃机制防止种群同化. 采用23个单峰、多峰和固定维度下的复杂多峰函数作为测试集进行仿真实验,结果表明所提出算法具有更优的搜索性能,能够有效提高种群多样性,防止算法早熟收敛,且能够适应不同类型的函数优化问题.

关键词: 混合蛙跳算法；种群多样性；信息交互；反向跳跃；局部更新；函数优化

中图分类号: TP301.6 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2019.0719 **开放科学(资源服务)标识码(OSID):**

引用格式: 申晓宁, 黄遥, 游璇, 等. 基于解空间反向跳跃和信息交互强化的新型混合蛙跳算法[J]. 控制与决策, 2021, 36(1): 105-114.

A new shuffled frog leaping algorithm based on reverse leaping in solution space and information interaction enhancement

SHEN Xiao-ning[1,2,3†], *HUANG Yao*[1], *YOU Xuan*[1], *WANG Qian*[1]

(1. School of Automation，Nanjing University of Information Science and Technology，Nanjing 210044，China；2. Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology，Nanjing University of Information Science and Technology，Nanjing 210044，China；3. Jiangsu Key Laboratory of Big Data Analysis Technology，Nanjing University of Information Science and Technology，Nanjing 210044，China)

Abstract: The diversity of population and the depth and way of information interaction have a profound impact on the ability of the shuffled leaping frog algorithm in climbing, exploring and development. Aiming at the shortcomings of the shuffled leaping frog algorithm, such as the inclination to local optimum, the slowness of convergence speed and the poor optimization accuracy, a new shuffled frog leaping algorithm based on reverse leaping in solution space and information interaction enhancement is proposed. Firstly, the information exchange between sub-inferior solutions and sub-optimal solutions is added to promote the utilization of information within sub-groups. In order to reduce the probability of inferior solutions in the later iteration period and to enhance the spatial development ability, the idea of reverse leaping is introduced to improve the local update mechanism. Then, the 2-opt method is used to realize the local optimal variation of sub-groups and increase the diversity of sub-groups. Finally, the local optimal crossover method is adopted to deepen the depth of interaction among subpopulations, and the reverse leaping mechanism is used to prevent population assimilation. The simulation experiments are carried out with a test sets includes 23 unimodal, multimodal and fixed-dimension complex functions. The results of simulation show that the proposed algorithm has better search performance, improves the diversity of population effectively, prevents premature convergence of the algorithm, and can adapt to different types of function optimization problems.

Keywords: shuffled frog leaping algorithm；population diversity；information interaction；reverse leaping；local update；function optimization

收稿日期: 2019-05-24；**修回日期:** 2019-07-16.

基金项目: 国家自然科学基金项目(61502239, 51705260)；江苏省自然科学基金项目(BK20150924)；江苏省青蓝工程项目.

责任编委: 林崇.

†通讯作者. E-mail: sxnystsyt@sina.com.

0 引 言

优化问题,即寻找解决问题的最佳方案自古有之,如今更是贯穿工程领域的各个方面. 随着人类需求的多样化和工程复杂化,优化问题的规模和复杂度也呈指数级上升,因此,优化算法作为求解寻优问题的工具也在不断发展. 时至今日,人们通过观察自然界中多个物种具有的群体协作的生存方式,在仿生学算法的基础上提出群智能算法,如粒子群算法[1]、蚁群算法[2]、人工蜂群算法[3]等. 蛙跳算法(shuffled frog leaping algorithm, SFLA)是Eusuff等[4]为解决组合优化问题于2003年提出的,也是群智能算法的一种,具有概念简单、容易理解、参数较少、易于实现且计算较快、全局寻优能力强等特点,吸引了国内外诸多学者进行研究. 骆剑平等[5]通过建立Markov模型,分析蛙群序列的最终转移状态,证明了蛙跳算法是全局收敛的. Kong等[6]提出了一种定向混合蛙跳算法,引入定向运动和实时交互机制,通过利用邻组最优个体信息和全局有效信息改变陷入局部个体的跳跃方向,提高了算法的爬山能力和求解精度. Zhen等[7]根据模因组性能平衡的原则改进分组策略,提出一种模因组内个体都参与进化的局部更新策略,促进组内信息交互更加充分,从而实现求解精度的提高. Sharma等[8]提出了一种局部知情混合蛙跳算法,最差解的步长由局部最优或全局最优和一个随机选择的邻域解决定,提高了算法的收敛性,保持了其集约和分散能力. Liu等[9]提出的自适应分组云模型随机蛙跳算法将算法定义域划分为若干区域,并为每组提供一个模因组,采用“精英策略”并通过云模型算法更新精英解,实现了搜索空间的缩小,改善了局部最优的情况. 综上,针对混合蛙跳算法的早熟收敛、后期收敛度慢、易于陷入局部最优、搜索停滞和求解精度不高等问题,学者们通过改进搜索策略或采用与其他算法相结合的方式改善算法性能.

本文首先分析基本混合蛙跳算法结构中存在的不足,针对发现的问题分别提出4个改进策略,由此提出一种基于解空间反向跳跃和信息交互强化的新型混合蛙跳算法(a new shuffled frog leaping algorithm based on reverse leaping in solution space and information interaction enhancement, SFLA-RLSS&IIE),并给出了算法的实现步骤;然后采用一组具有不同特征的函数分别验证每个改进策略的有效性,实验结果表明改进策略能够从寻优精度、收敛速度和爬山能力等方面提升算法性能;最后将所提出算法与经典进化算法进行实验对比,结果表明改进算法在单峰、多峰和固定维度下的复杂多峰函数中均表现出较好的寻优性能.

1 基本蛙跳算法原理及其存在问题

蛙跳算法作为一种仿生学群体智能算法,模拟湿地青蛙种群觅食行为. 在湿地中分布若干石头,青蛙在石头上跳跃以寻找到食物最多的点,每只青蛙带有不同的信息,将种群分为不同的组,通过不同的青蛙个体进行信息交流,实现局部搜索;当局部搜索执行到一定程度时,将各子群混洗,实现全局信息交换. 根据这一仿生机制,蛙跳算法的流程如下.

step 1: 种群初始化. 随机生成N个个体,其中每个个体为一个V维的解$X_k = \{x_{k_1}, x_{k_2}, \dots, x_{k_V}\}$, $k = 1, 2, \dots, N$.

step 2: 子群划分. 计算每个解的适应值,并根据该值对个体进行降序排列,将种群划分为m子群,每子群内n个解,其中$N = m \times n$. 将第1个解放入第1子群,第2个解放入第2子群,第m个解放入第m子群,第$m+1$个解放入第1子群,以此类推,则解$X_{i+m(t-1)}$为第$i\,(i = 1, 2, \dots, m)$组第t个解.

step 3: 局部搜索. 将整个种群中适应度值最好的解标记为X_{g},第i个子群中的最优解和最劣解分别标记为$X_{i\mathrm{b}}$、$X_{i\mathrm{w}}$. 在迭代过程中,首先根据局部最优解$X_{i\mathrm{b}}$的信息更新局部最劣解$X_{i\mathrm{w}}$,如下所示:

$$D = \mathrm{rand}(\,)(X_{i\mathrm{b}} - X_{i\mathrm{w}}). \tag{1}$$

$$X'_{i\mathrm{w}} = \begin{cases} X_{i\mathrm{w}} + D,\ D_{\min} \leqslant D \leqslant D_{\max}; \\ X_{i\mathrm{w}} + D_{\min},\ D < D_{\min}; \\ X_{i\mathrm{w}} + D_{\max},\ D_{\max} < D. \end{cases} \tag{2}$$

其中:rand( )生成[0, 1]内均匀分布的随机数,$[D_{\min}, D_{\max}]$为跳跃步长的允许范围,当式(1)生成的步长D不在允许的范围内时,将实际步长取值为D最接近的边界值. 若此时产生的新解$X'_{i\mathrm{w}}$其适应度未能优于$X_{i\mathrm{w}}$,则采用全局最优解X_{g}替代$X_{i\mathrm{b}}$、更新$X_{i\mathrm{w}}$,有

$$D = \mathrm{rand}(\,)(X_{\mathrm{g}} - X_{i\mathrm{w}}), \tag{3}$$

$$X'_{i\mathrm{w}} = \begin{cases} X_{i\mathrm{w}} + D,\ D_{\min} \leqslant D \leqslant D_{\max}; \\ X_{i\mathrm{w}} + D_{\min},\ D < D_{\min}; \\ X_{i\mathrm{w}} + D_{\max},\ D_{\max} < D. \end{cases} \tag{4}$$

若产生的新解$X'_{i\mathrm{w}}$仍未优于$X_{i\mathrm{w}}$,则在解空间内随机生成一个新解取代局部最劣解,有

$$X'_{i\mathrm{w}} = \mathrm{rand}(\,)(\max - \min) + \min, \tag{5}$$

其中[min, max]为搜索区域,即解的定义域.

step 4: 全局混合. 当所有子群均完成内部更新

后,将子群重新混合后再次进行分组和子群内部更新,直至到达算法停止条件.

混合蛙跳算法结合模因算法[10]和粒子群算法的优点,即采用模因算法中的个体更新机制模拟青蛙跳跃,通过粒子群的群体性行为实现全局的信息共享,算法整体结构简单,思路清晰,易于实现且寻优性能良好. 但是,从结构中也不难发现存在诸多不足,主要体现在局部更新和全局混洗两个环节. 首先,式(1)和(3)中仅有全局最优解X_{g}、局部最优解X_{b}和局部最劣解X_{w}参与信息的交互,子群内部信息交互并不充分,迭代到一定程度后,X_{g}和X_{b}更新困难,X_{w}在解空间内朝一固定方向不断跳跃,易于陷入最优且收敛早熟;其次,式(5)采用随机的方式生成新解,其质量无法保证,尤其在搜索后期,该策略参与局部更新概率增加,当进化到一定精度后新解多为无效解,甚至部分劣解会降低爬坡能力,再次增加陷入局部最优的概率;再次,全局混洗策略只是简单地将所有个体重新排序分组,这种策略不但没有让各种群中信息实现深度交换,且伴随搜索进程,种群内个体的相似度不断增加,后期极易造成搜索停滞,从而导致寻优精度不高.

2　改进的混合蛙跳算法

蛙跳算法子群个体更新主要通过不同个体之间信息交互指导劣解跳跃方向和步长,实现其在解空间内的跳跃. 因此,本文从信息获取源、信息交互深度和解空间结构考虑,设计了4个方面的改进策略,提出一种基于解空间反向跳跃和信息交互强化的新型混合蛙跳算法(SFLA-RLSS&IIE).

2.1　引入其他信息源

通过分析基本蛙跳算法的子群更新策略可以发现,其信息交互的来源只有3类:全局最优解、局部最优解和局部最劣解,因此可以通过增加信息获取源的方式让子群内邻域解参与到信息交互中,实现子群内部信息的充分利用. 为了保证引入的信息源对于解更新具有一定的指导作用,在局部最劣解更新的同时,选择局部次优解$X_{i\mathrm{bb}}$对局部次劣解$X_{i\mathrm{ww}}$进行更新,有

$$D_1=\mathrm{rand}(\,)(X_{i\mathrm{bb}}-X_{i\mathrm{ww}}). \tag{6}$$

$$X'_{i\mathrm{ww}}=\begin{cases}X_{i\mathrm{ww}}+D_1,\ D_{\min}\leqslant D_1\leqslant D_{\max};\\ X_{i\mathrm{ww}}+D_{\min},\ D_1<D_{\min};\\ X_{i\mathrm{ww}}+D_{\max},\ D_{\max}<D_1.\end{cases} \tag{7}$$

以此类推将次优解引入式(3)、(4)和(5).

2.2　引入反向跳跃机制

解个体在解空间中的跳跃应该具有一定方向性和目的性才有助于算法收敛,而随机的跳跃会让群体变得无序和混乱,因此提出对局部更新式(5)的替代机制,如下所示:

$$D=\mathrm{rand}(\,)(-X_{\mathrm{g}}-X_{i\mathrm{w}}). \tag{8}$$

$$X'_{i\mathrm{w}}=\begin{cases}X_{i\mathrm{w}}+D,\ D_{\min}\leqslant D\leqslant D_{\max};\\ X_{i\mathrm{w}}+D_{\min},\ D<D_{\min};\\ X_{i\mathrm{w}}+D_{\max},\ D_{\max}<D.\end{cases} \tag{9}$$

因为式(1)和(3)已分别利用局部最优解和全局最优解,且没有性能更加优良的解指导局部最劣解更新,所以式(8)的更新策略仍考虑全局最优解作为信息获取源. 从解集空间的角度出发,选择全局最优解关于坐标原点的空间对称位置作为劣解的跳跃方向,这样既保证了获取信息的指导意义,又扩大了搜索空间,还提高了子群爬山能力,有助于跳出局部最优.

2.3　局部最优解变异

增加子群的多样性有助于增强算法跳出局部最优,提高寻优精度,但在迭代后期种群的全局最优解和局部最优解难以被更新,其作为主要信息源将影响整个群体的走向,长期保持不变,会将个体逐渐同化,最终导致算法停滞. 本文借鉴2-opt法[11]对局部最优解进行变异,即在$[1,V]$中随机生成两个整数p和$q(p<q)$,对局部最优解的第p维至第q维进行逆序排列处理,如图1所示,解的内部由结构1变异成结构2. 这种变异方式既保留了局部最优解的部分信息,又增加了解的多样性,同时仅变异局部最优解不会过度影响子群的探索能力.

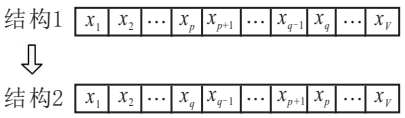

图1　局部最优解变异结构

2.4　子群信息深度交互

子群信息也是重要的信息源之一,主要作用于全局信息交互,因此信息交互的方式与深度十分重要. 蛙跳算法在全局混合的过程中只是实现了简单的信息交流,并未对各个子群信息进行深度挖掘. 然而,如果一味增加交互深度则只会增加种群的相似性和单一性,导致后续迭代中的分组将不再有意义. 针对这一问题,为了兼顾信息交互深度和多样性,同时引入交叉策略和空间反向跳跃思想. 将m个子群的局部最优解进行交叉,生成m个新解分别代替各子

群的局部最劣解,然后进入全局混合重新分组,即

$$\begin{cases} X_{1\mathrm{w}} = -a_1X_{1\mathrm{b}} - a_2X_{2\mathrm{b}} - \ldots - a_mX_{m\mathrm{b}};\\ X_{2\mathrm{w}} = -a_1X_{2\mathrm{b}} - a_2X_{3\mathrm{b}} - \ldots -\\ \qquad a_{m-1}X_{m\mathrm{b}} - a_mX_{1\mathrm{b}};\\ \quad\vdots\\ X_{m\mathrm{w}} = -a_1X_{m\mathrm{b}} - a_2X_{1\mathrm{b}} - \ldots -\\ \qquad a_{m-1}X_{(m-2)\mathrm{b}} - a_mX_{(m-1)\mathrm{b}}. \end{cases} \tag{10}$$

为了保证新解的多样性且能扩大搜索空间,跳出局部最优,类似第2.2节从空间结构角度出发,选择关于原点对称位置的解作为交叉因子进行交叉. 由于交叉权重系数影响着从交叉因子获取的信息量大小,且子群劣解的更新应主要从自身子群内部获取较多信息,同时从其他子群获取较少信息,以在一定程度上防止发生突变现象,取$a_1 > a_2 = a_3 = \ldots = a_m$,具体取值如下:

$$\begin{cases} a_1 = 1/m + \mathrm{rand}(\,)(1 - 1/m),\\ a_2 = a_3 = \ldots = a_m = (1 - a_1)/(m - 1). \end{cases} \tag{11}$$

2.5 改进算法实现

基本的蛙跳算法经过上述4种策略改进后得到一种基于解空间反向跳跃和信息交互强化的新型混合蛙跳算法,其伪代码如下.

算法1 基于解空间反向跳跃和信息交互强化的新型混合蛙跳算法.

00: 参数初始化(种群规模N,子群数m,解维数V,局部搜索次数L,混合迭代次数G);
01: 随机生成初始解群;
02: for $s = 1$ to G do
03: 计算每个解的适应度$f(X_k)$,按照适应度降序排列,并进行子群划分,解$X_{i+m(t-1)}$为第i组第t个解;
04: for $j = 1$ to L do
05: for $i = 1$ to m do
06: 计算第i个子群的局部最优解$X_{i\mathrm{b}}$、次优解$X_{i\mathrm{bb}}$、次劣解$X_{i\mathrm{ww}}$、最劣解$X_{i\mathrm{w}}$和全局最优解X_{g};
07: if $j > 1$
08: 根据第2.3节进行局部最优变异;
09: end if
10: 根据第2.1节引入信息源,由式(1)、(2)、(6)和(7)分别更新得到$X'_{i\mathrm{w}}$和$X'_{i\mathrm{ww}}$;
11: if $f(X'_{i\mathrm{w}}) < f(X_{i\mathrm{w}})$
12: $X_{i\mathrm{w}} = X'_{i\mathrm{w}}$;
13: else
14: 根据式(3)和(4)生成$X'_{i\mathrm{w}}$;
15: if $f(X'_{i\mathrm{w}}) < f(X_{i\mathrm{w}})$
16: $X_{i\mathrm{w}} = X'_{i\mathrm{w}}$;
17: else
18: 根据式(8)和(9)生成$X'_{i\mathrm{w}}$;
19: $X_{i\mathrm{w}} = X'_{i\mathrm{w}}$;
20: end if
21: end if
22: 参照代码11 ~ 代码21的方式更新$X'_{i\mathrm{ww}}$;
23: end for
24: end for
25: 记录局部最优解$[X_{1\mathrm{b}}\ \ X_{2\mathrm{b}}\ \ \ldots\ \ X_{m\mathrm{b}}]$和局部最劣解$[X_{1\mathrm{w}}\ \ X_{2\mathrm{w}}\ \ \ldots\ \ X_{m\mathrm{w}}]$;
26: 根据式(10)和(11)实现子群间信息交互,然后混洗;
27: end for

2.6 复杂度分析

SFLA-RLSS&IIE算法的时间复杂度主要取决于: 个体排序分组为$O(N)$,局部最优变异为$O(Lm - m)$,子群最劣解更新为$O(LmV)$,子群次劣解更新为$O(LmV)$,子群间信息交互为$O(m)$. 综上,SFLA-RLSS&IIE算法总的时间复杂度取上述分析结果中的最大值$O(LmV)$. 对于SFLA-RLSS&IIE算法的空间复杂度,种群初始化时产生N个V维度的个体,其空间复杂度为$O(NV)$,因为算法中个体更新都在原有个体上替换,不需要占用新的运算空间,所以算法的空间复杂度为$O(NV)$. 相较于SFLA算法的时间复杂度$O(LmV)$和空间复杂度$O(NV)$,本文所提出算法的计算复杂度并未增加.

3 仿真实验与分析

为了验证所提出算法及改进策略的有效性,采用Matlab R2016a进行仿真实验,计算机处理参数为Intel(R) Core(TM) i5-4210U CPU@1.70 GHz, 4 G运行内存. 实验采用4个指标检验算法性能,收敛平均值(mean)代表算法收敛的精度,收敛值标准差(std)衡量算法的稳定性,收敛的平均步数(C.I.)表明收敛速度状态,为进一步对比算法性能的优劣,引入Wilcoxon符号秩检验(W). 其中下文的第3.2节中定义差值最小精度为1.0e-05,第3.3节中定义差值最小精度为0,置信水平设置为0.05.

3.1 测试集选择

测试函数是检验进化算法在连续空间寻优问题性能测试的主要工具,为了能够充分验证所提出改进策略和改进算法的有效性,从测试函数的特征和复杂度出发,选取目前文献常用的23个测试函数作为测

试集[12-16]. 其中: 函数 $f_1 \sim f_7$ 为单峰函数, $f_8 \sim f_{13}$ 为多峰函数, $f_{14} \sim f_{23}$ 为固定维度下的复杂多峰函数. 测试集的详细信息可参见文献[12-13].

3.2　改进策略有效性验证

本文共提出4种改进策略, 分别为在局部更新中增加信息源(adding information source, AIS), 引进反向跳跃机制(reverse leaping principle, RLP), 实现局部最优解变异(variation of local optimal solutions, VLS)和在全局混合中深化子群间信息交换(deepening information exchange among subgroups, DIES). 为验证每一种策略的有效性, 将4种改进策略分别引入基本蛙跳算法, 并将4种改进算法与基本算法进行比较, 算法参数设置如下[12]. 种群个体总数 N 为80, 子群个数 n 为10, 解维数 V 为30, 每个子群局部迭代次数 L 为20, 全局迭代次数 G 为1 000, 每种算法独立运行30次. 表1、表2和表3统计了4种改进策略分别在单峰函数和多峰函数中的测试结果, 包含收敛平均值、标准差、收敛平均步数和Wilcoxon符号秩检验结果. 分

表 1　改进策略在单峰函数测试统计结果

函数	算法	mean	std	C.I.	W
	SFLA	1.675 3e-04	1.980 9e-04	1 000	
	SFLA_AIS	6.272 1e-13	1.625 7e-12	1 000	+
f_1	SFLA_RLP	5.257 9e-135	1.969 3e-134	1 000	+
	SFLA_VLS	2.772 3e-15	1.098 5e-14	1 000	+
	SFLA_DIES	**0**	**0**	**394**	+
	SFLA	0.046 67	0.062 0	1 000	
	SFLA_AIS	8.229 4e-07	2.379 4e-06	1 000	+
f_2	SFLA_RLP	4.616 4e-89	1.626 2e-88	1 000	+
	SFLA_VLS	8.888 5e-14	8.161 3e-14	1 000	+
	SFLA_DIES	**0**	**0**	**774**	+
	SFLA	59.721 0	22.061 1	1 000	
	SFLA_AIS	4.513	2.152 3	1 000	+
f_3	**SFLA_RLP**	**0**	**0**	**606**	+
	SFLA_VLS	1.033 8e-40	5.227 9e-40	1 000	+
	SFLA_DIES	0	0	691	+
	SFLA	5.895 1	1.103 0	992	
	SFLA_AIS	3.788 1	0.792 6	997	+
f_4	SFLA_RLP	2.948 1e-13	4.044 0e-13	987	+
	SFLA_VLS	0.513 8	0.533 6	1 000	+
	SFLA_DIES	**0**	**0**	**886**	+
	SFLA	78.635 4	48.157 3	1 000	
	SFLA_AIS	50.678 8	33.253 6	1 000	+
f_5	SFLA_RLP	28.798 5	0.321 8	1 000	+
	SFLA_VLS	**25.736 4**	**6.892 1**	**1 000**	+
	SFLA_DIES	28.759 9	0.069 6	997	+
	SFLA	3.169 5e-04	8.749 2e-04	1 000	
	SFLA_AIS	4.650 3e-12	1.448 4e-11	1 000	+
f_6	SFLA_RLP	4.822 3	0.655 2	1 000	−
	SFLA_VLS	**1.347 6e-15**	**1.810 2e-15**	**1 000**	+
	SFLA_DIES	2.756 51	0.559 9	994	−
	SFLA	0.001 1	3.564 2e-04	762	
	SFLA_AIS	5.926 5e-04	1.844 0e-04	836	+
f_7	**SFLA_RLP**	**3.103 4e-06**	**3.383 4e-06**	**522**	+
	SFLA_VLS	2.282 3e-04	1.086 5e-04	723	+
	SFLA_DIES	2.747 0e-05	2.612 2e-05	553	+

表 2　改进策略在多峰函数测试统计结果

函数	算法	mean	std	C.I.	W
	SFLA	−6.019 4e+03	669.489 1	1 000	
	SFLA_AIS	−7.079 8e+03	509.217 5	1 000	+
f_8	SFLA_RLP	−6.641 3e+03	609.367 4	1 000	+
	SFLA_VLS	**−1.142 1e+04**	**889.044 7**	**1 000**	+
	SFLA_DIES	−7.253 7e+03	555.609 6	998	+
	SFLA	14.402 2	4.217 9	1 000	
	SFLA_AIS	12.936 4	5.864 4	982	+
f_9	SFLA_RLP	0	0	99	+
	SFLA_VLS	0	0	802	+
	SFLA_DIES	**0**	**0**	**24**	+
	SFLA	0.925 2	0.695 7	1 000	
	SFLA_AIS	2.5782e-04	6.8262e-04	1000	+
f_{10}	**SFLA_RLP**	**1.125 0e-15**	**9.013 5e-16**	**203**	+
	SFLA_VLS	5.353 5e-09	5.859 5e-09	1 000	+
	SFLA_DIES	3.611 9e-15	1.528 3e-15	62	+
	SFLA	0.041 9	0.033 0	1 000	
	SFLA_AIS	0.045 6	0.058 6	998	−
f_{11}	SFLA_RLP	0	0	121	+
	FLA_VLS	0.002 3	0.003 6	830	+
	SFLA_DIES	**0**	**0**	**26**	+
	SFLA	0.011 0	0.042 8	1 000	
	SFLA_AIS	2.287 7e-09	1.252 7e-08	1 000	+
f_{12}	SFLA_RLP	1.640 4	0.310 9	1 000	−
	SFLA_VLS	**6.200 8e-18**	**1.252 0e-17**	**1 000**	+
	SFLA_DIES	0.005 0	0.003 1	989	+
	SFLA	0.001 9	0.004 2	1 000	
	SFLA_AIS	2.287 7e-09	1.252 7e-08	1 000	+
f_{13}	SFLA_RLP	2.374 2	0.310 9	1 000	−
	SFLA_VLS	**6.220 8e-18**	**1.251 6e-17**	**1 000**	+
	SFLA_DIES	0.296 4	0.189 5	987	−

表3 函数$f_1 \sim f_{13}$中Wilcoxon符号秩检验统计

改进策略	+	=	−	gm
SFLA_AIS	12	0	1	11
SFLA_RLP	10	0	3	7
SFLA_VLS	13	0	0	13
SFLA_DIES	11	0	2	9

别用“+”、“=”、“−”表示应用改进策略的算法显著“优于”、“等于”、“劣于”基本蛙跳算法,gm为“+”与“−”个数的差值.

对于单峰函数,除了在f_6中,仅有SFLA_AIS和SFLA_VLS表现较优外,其他函数中4种改进策略均表现出较高的寻优性能,其中SFLA_RLP和SFLA_DIES在$f_1 \sim f_4$的收敛精度上表现出了突出的性能;函数f_5为Rosenbrock函数,具有特征复杂、变量间彼此依赖等特点,对于通过最优解或较好解进行更新劣解的蛙跳算法而言,极容易陷入局部最优,因为改进策略并未改变蛙跳算法信息交互的本质,所以虽然改进策略表现出优于原算法的性能,但是结果并不能让人满意. 对于多峰函数,6个函数分别测试4种改进策略,在共24条测试结果中,仅有4条测试性能不佳,分别为f_{11}测试SFLA_AIS、f_{12}测试SFLA_RLP、f_{13}测试SFLA_RLP和SFLA_DIES. 由于函数f_{12}和f_{13}均引入了惩罚函数的复杂多峰函数,其惩罚函数约束了最优解在解空间的位置,改进策略SFLA_RLP和SFLA_DIES引入的解空间反向跳跃的思想会产生一些劣解,从而导致寻优性能下降. 其他测试中改进策略均呈现优势性能,根据表2平均收敛步数(C.I.)可以看出,策略SFLA_DIES也表现出较快的收敛速度. 通过表3的Wilcoxon符号秩检验统计结果可以看出,4种改进算法在绝大多数测试函数中均显著优于基本蛙跳算法,表明4种改进策略是可行有效的.

为了便于观察改进策略的收敛情况和特点,图2和图3分别给出了改进策略在单峰和多峰函数中的收敛曲线. 在单峰函数中,策略SFLA_AIS虽然有效地提升了算法的寻优精度,但是仅在函数f_6中表现相对突出;策略SFLA_RLP除函数f_6外,在寻优精度和收敛速度上均表现良好,且发现其曲线存在多处阶梯下降,阶梯较短,表明在搜索过程中多次经历局部最优,但是由于具有较强的爬坡能力,能够跳出局部最优;策略SFLA_VLS在f_3、f_6表现良好,相较于SFLA除收敛精度增加外,还具备一定的爬坡能力,整

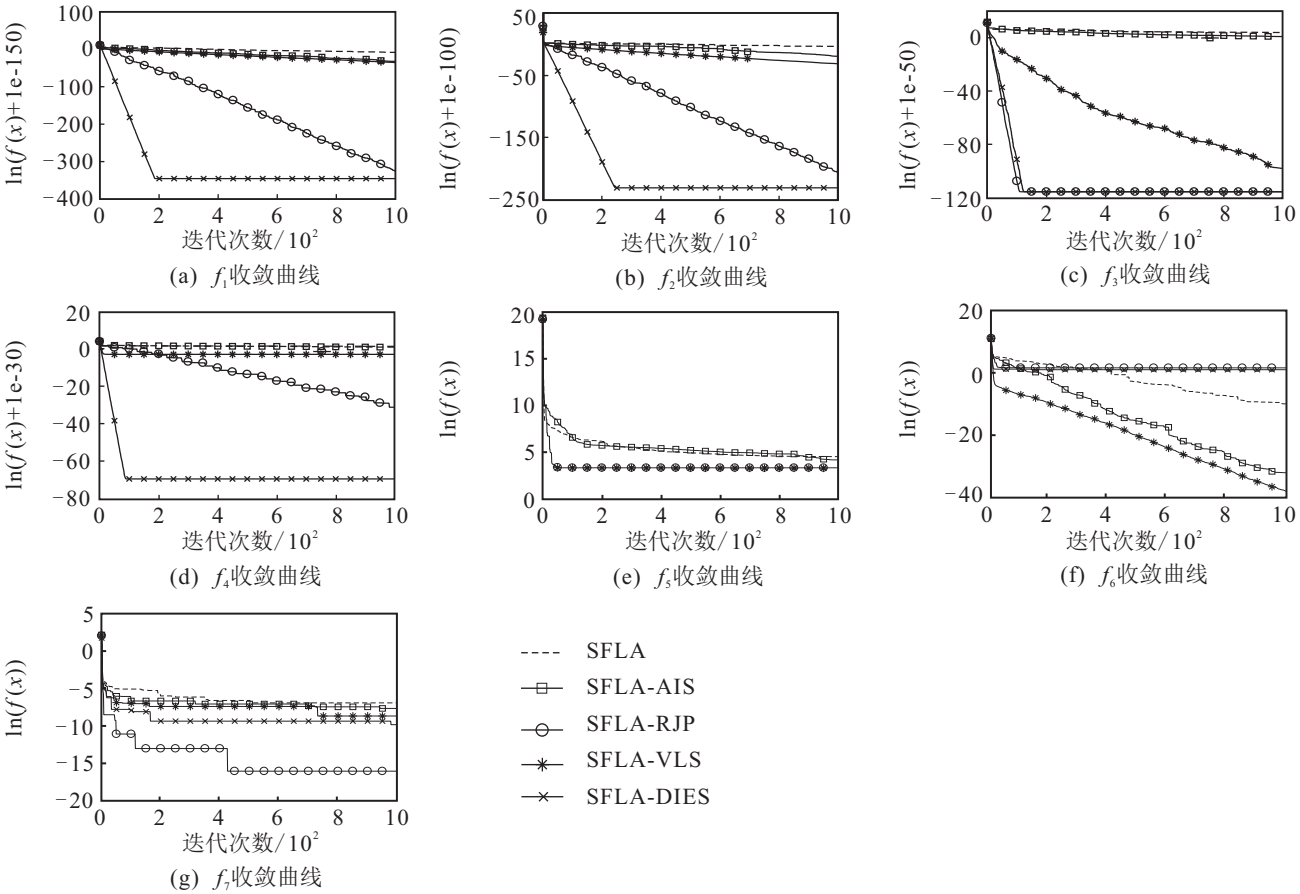

图2 单峰函数收敛曲线

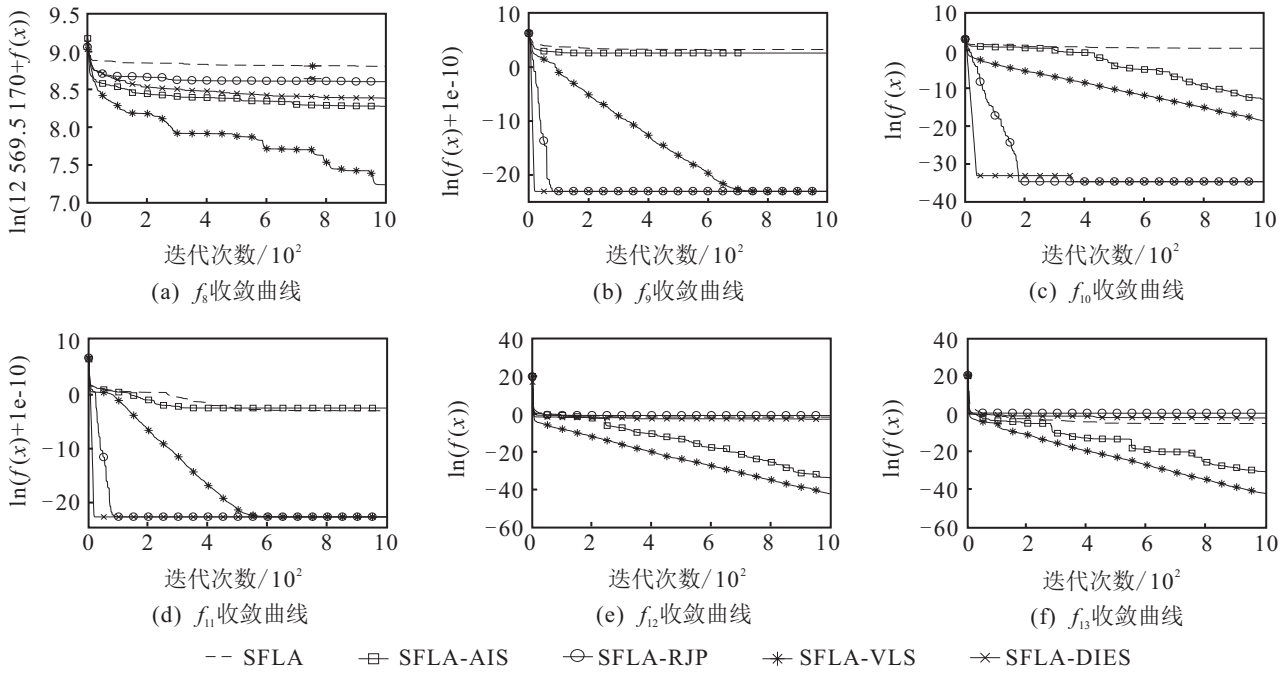

图 3　多峰函数收敛曲线

表4　改进策略在固定维度复杂函数测试统计结果

函数	算法	mean	std	C.I.	W	函数	算法	mean	std	C.I.	W
	SFLA	0.998 0	1.367 5e-16	20			SFLA	−3.862 8	5.923 9e-05	686	
	SFLA_AIS	0.998 0	1.009 9e-16	14	=		**SFLA_AIS**	**−3.862 8**	**3.161 7e-15**	**511**	=
f_{14}	SFLA_RLP	0.998 0	1.550 1e-16	139	=	f_{19}	SFLA_RLP	−3.862 8	0.003 4	864	=
	SFLA_VLS	0.998 0	1.2370e-16	17	=		SFLA_VLS	−3.862 8	2.471 1e-06	802	=
	SFLA_DIES	**0.998 0**	**0**	**22**	=		SFLA_DIES	−3.862 8	1.192 1e-15	813	=
	SFLA	6.632 6e-04	7.321 0e-05	773			SFLA	−3.284 6	0.059 2	809	
	SFLA_AIS	**3.234 3e-04**	**3.805 9e-05**	**871**	+		SFLA_AIS	−3.284 6	0.059 2	809	=
f_{15}	SFLA_RLP	3.237 6e-04	8.474 7e-05	799	+	f_{20}	SFLA_RLP	−3.283 7	0.057 6	580	−
	SFLA_VLS	3.237 5e-04	2.212 8e-10	829	+		**SFLA_VLS**	**−3.294 6**	**0.052 1**	**806**	+
	SFLA_DIES	3.237 5e-04	1.118 0e-08	585	+		SFLA_DIES	−3.284 3	0.065 2	113	−
	SFLA	−1.031 6	6.775 2e-16	17			SFLA	−6.055 2	1.286 22	244	
	SFLA_AIS	**−1.031 6**	**6.775 2e-16**	**14**	=		SFLA_AIS	−9.646 4	1.546 2	133	+
f_{16}	SFLA_RLP	−1.031 6	6.560 7e-05	16	=	f_{21}	SFLA_RLP	−9.857 7	1.374 9	132	+
	SFLA_VLS	−1.031 6	5.215 5e-16	62	=		SFLA_VLS	−9.655 1	1.895 2	115	+
	SFLA_DIES	−1.031 6	6.184 8e-16	22	=		**SFLA_DIES**	**−9.983 2**	**0.930 8**	**59**	+
	SFLA	0.397 9	0	71			SFLA	−10.227 1	0.962 9	205	
	SFLA_AIS	**0.397 9**	**0**	**36**	=		SFLA_AIS	−10.227 1	0.962 9	101	=
f_{17}	SFLA_RLP	0.398 0	5.301 4e-04	20	−	f_{22}	SFLA_RLP	−9.860 7	1.621 1	130	−
	SFLA_VLS	0.397 9	0	122	=		**SFLA_VLS**	**−10.402 9**	**1.189 3e-15**	**123**	+
	SFLA_DIES	0.397 9	3.439 1e-08	25	=		SFLA_DIES	−10.402 9	5.882 2e-15	63	+
	SFLA	3	1.253 3e-15	37			SFLA	−10.536 4	5.056 7e-15	568	
	SFLA_AIS	**3**	**1.332 2e-15**	**25**	=		SFLA_AIS	−10.456 1	0.987 3	214	−
f18	SFLA_RLP	3.038 9	0.183 9	202	−	f_{23}	SFLA_RLP	−10.354 7	0.044 2	332	−
	SFLA_VLS	3	2.238 7e-15	57	=		**SFLA_VLS**	**−10.536 4**	**2.212 8e-15**	**409**	=
	SFLA_DIES	3.004 8	0.022 7	62	−		SFLA_DIES	−10.320 8	0.180 6	746	−

体性能优于策略SFLA_AIS;策略SFLA_DIES其收敛曲线多为斜率较大的直线,收敛过程应该是快速的、持续的,直至收敛到最优解,但也会出现早熟收敛现象,如函数f_6. 在多峰函数中,策略SFLA_AIS具有较好的寻优能力和跳出局部最优的能力,如函数f_8、f_{10}、f_{12}、f_{13};策略SFLA_RLP在函数f_{12}、f_{13}中性能不佳;策略SFLA_VLS. 由图3收敛曲线可见:展示出了远高于单峰函数的适应性;策略SFLA_DIES依旧保持其收敛速度快、精度高的特点.

纵观4种策略在单峰和多峰函数中的测试统计结果和收敛曲线,发现以下特点:策略SFLA_AIS通过增加信息交换源的方式促进子群内部信息的交换,但是也会加速子群内个体的同化,所以其综合性能并不突出;策略SFLA_RLP在一定程度上保证了新解的质量,同时当局部更新中前两步产生的新解质量不佳时,后续搜索有可能陷入该子群的局部最优,向最优解反方向跳跃能够有效跳出原来的搜索范围,所以该策略具有突出的跳出局部最优的能力;策略SFLA_VLS通过局部最优变异的方式增加子群的多样性,对于寻优精度和提升爬山能力均有帮助,所以其收敛速度和精度相对均衡;策略SFLA_DIES采用局部最优交叉变异和反向跳跃思想组合的方式,既增加了种群多样性,又针对局部最优进行反向跳跃,扩大了搜索范围,所以其收敛速度和精度均表现优良;由于策略SFLA_RLP和SFLA_DIES均采用反向跳跃的思想,在函数测试中的表现具有一定相似性,如在函数f_6、f_{12}、f_{13}均变现不佳,在其他函数中表现良好. 此外,4种策略具有一定的互补性,当策略SFLA_RLP和SFLA_DIES表现不佳时,策略SFLA_AIS和SFLA_VLS的表现相对较好,如函数f_6、f_8、f_{12}和f_{13}.

为了进一步验证改进策略的有效性和普适性,利用10个固定维度复杂函数$f_{14}\sim f_{23}$对其性能进行测试,测试统计结果如表4所示. 由表4可见,SFLA在10个测试函数中均具有较为优良的性能,在这一前提下,改进策略能够不影响原算法的性能,且在部分测试函数上表现更优(寻优精度更高或是保持精度的前提下收敛速度更快). 综合以上实验结果可以看出,所提出改进策略是有效的,在爬山能力、寻优精度和收敛速度等方面的性能均有所提高.

3.3 改进算法性能测试

将以上4种改进策略应用到基本的混合蛙跳算法中,提出一种基于解空间反向跳跃和信息交互强化的新型混合蛙跳算法(SFLA-RLSS&IIE),并将其与近年来提出的4种具有代表性的进化算法(蝴蝶优化算法(BOA)[17]、改进的人工蜂群算法(PS-ABC)[18]、基于教与学优化算法(TLBO)[19]和改进的混合蛙跳算法(MS-SFLA)[12])在函数测试集中进行比较,算法共同参数如下:种群规模N为80,最大迭代次数G为1 000,解空间V为30维. 表5为单峰函数与多峰函数测试统计结果,分别用“+”“=”“−”表示所提出算法显著“优于”“等于”“劣于”相应的对比算法,并于表尾处对各对比算法中“+”“=”“−”的数量进行统计. 由表5可见:所提出算法SFLA-RLSS&IIE在11个测试函数中均值最优,其中7个函数中得到的全局最优与部分对比算法没有显著差别,在另外3个函数中的均值显著优于其他对比算法. 在单独对比中,所提出算法在13个测试函数中均显著优于BOA;与PS-ABC相比,寻优结果7个显著优胜,5个无显著差异,1个显著较差;相较于TLBO,测试结果中8个显著优胜,4个无显著差异,1个显著劣解;对比MS-SFLA,5个测试结果显著优胜,7个没有明显差异,1个显著劣解. 测试统计结果如表5最后一行所示,可以发现,所提出算法在绝大多数单峰与多峰测试函数中显著优于对比算法,虽然在个别测试函数上显著劣于比较算法,但其寻优结果也在可以接受的范围内. 总体上看,所提出算法的综合性能优于其他比较算法. 此外,所提出算法也存在不足,如在函数f_5中虽然显著优于BOA、PS-ABC、TLBO,但是显著差于MS-SFLA,且其寻优精度也不高,原因在于虽然所提出算法综合了4种改进策略,但4种改进策略对于算法求解函数f_5的性能并没有显著提升,且4种改进策略相互独立,所以RLSS&IIE-SFLA求解结果并不令人满意;在函数f_6和f_{13}中,分别显著劣于TLBO和PS-ABC,但求解精度均小于1e-15,在可接受范围内.

对比表5与表1、表2的结果可见,在函数f_{12}、f_{13}中,所提出算法的寻优精度(均值mean)虽然稍逊于单一策略SFLA_VLS,但是对两者进行置信度为0.05的Wilcoxon符号秩检验(差值最小精度为10e-05)不存在显著性差异. 同时观察其他函数不难发现,所提出算法的求解精度均优于单一策略,表明策略之间的组合不会影响彼此的性能,能够实现在不同函数上寻优性能的互补,从而使得改进算法在更大搜索空间的测试函数中表现出良好的性能.

将5种算法在10个固定维度的复杂多峰函数$f_{14}\sim f_{23}$中进行测试,以进一步验证本文算法的性能. 测试统计结果如表6所示,所提出改进算法在所有10个测试函数中的寻优结果均显著优于对比算法

表5　不同算法在单峰和多峰函数测试统计结果

函数	BOA			PS-ABC			TLBO			MS-SFLA			SFLA-RLSS & IIE	
	mean	std	W	mean	std	W	mean	std	W	mean	std	W	mean	std
f_1	1.56e-14	6.82e-16	+	0	0	=	0	0	=	0	0	=	**0**	**0**
f_2	8.80e-12	2.27e-12	+	0	0	=	0	0	=	0	0	=	**0**	**0**
f_3	1.61e-14	9.53e-16	+	6.12e+03	1.69e+03	+	0	0	=	0	0	=	**0**	**0**
f_4	1.11e-11	4.59e-13	+	8.59e-115	4.71e-114	+	4.25e-317	0	+	0	0	=	**0**	**0**
f_5	28.90	0.03	+	1.59	4.41	+	27.91	0.31	+	**1.09e-04**	**2.61e-04**	−	25	0.55
f_6	5.23	0.48	+	5.72e-16	8.25e-17	+	**2.04e-29**	**1.09e-29**	−	6.42e-07	1.25e-06	+	3.19e-16	1.79e-16
f_7	4.40e-04	2.10e-04	+	2.15e-02	6.88e-03	+	1.16e-02	5.50e-03	+	2.37e-04	1.83e-04	+	**2.19e-06**	**1.94e-06**
f_8	−4 372.20	322.81	+	−12 564.23	22.535 4	+	−12 428.60	155	+	−12 569.23	14.60	+	**−12 569.40**	**11.57**
f_9	4.72e-08	2.58e-07	+	0	0	=	6.95e-13	1.35e-12	+	0	0	=	**0**	**0**
f_{10}	1.02e-11	1.36e-12	+	8.88e-16	0	=	3.55e-15	8.32e-31	+	8.88e-16	0	=	**8.88e-16**	**0**
f_{11}	1.36e-12	9.92e-16	+	0	0	=	0	0	=	0	0	=	**0**	**0**
f_{12}	0.39	0.10	+	5.53e-16	8.69e-17	+	2.67e-08	6.79e-12	+	2.13e-08	1.99e-08	+	**1.13e-17**	**6.97e-17**
f_{13}	2.35	0.32	+	6.06e-18	5.61e-18	−	2.37e-08	4.81e-11	+	3.80e-05	2.47e-04	+	**2.01e-17**	**6.96e-17**
统计	**13**	**0**	**0**	**7**	**5**	1	**8**	**4**	**1**	**5**	**7**	**1**	+	=	−

表6　不同算法在固定维度的复杂多峰函数测试统计结果

函数	BOA		PS-ABC		TLBO		MS-SFLA		SFLA-RLSS&IIE
	mean	W	mean	W	mean	W	mean	W	mean
f_{14}	0.999 2	+	0.998 0	=	0.998 0	=	0.998 0	=	**0.998 0**
f_{15}	3.260 3e-04	+	4.138 6e-04	+4	3.080 0e-04	+	3.193 2e-04	+	**3.074 9e-4**
f_{16}	−1.031 6	=	−1.031 6	=	−1.031 6	=	−1.031 6	=	**−1.031 6**
f_{17}	0.463 7	=	0.630 0	=	0.397 9	=	0.397 9	=	**0.397 9**
f_{18}	3.005 9	=	3.000 0	=	3.000 0	=	3.000 0	=	**3.000 0**
f_{19}	−3.861 3	+	−3.862 6	+	−3.862 6	+	−3.862 5	+	**−3.862 8**
f_{20}	−3.091 6	+	−3.322 0	+	−3.286 6	+	−3.236 6	+	**−3.322 6**
f_{21}	−8.176 5	+	−10.153 2	=	−10.153 2	=	−10.153 0	+	**−10.153 2**
f_{22}	−9.347 6	+	−10.402 9	=	−10.402 9	=	−10.402 5	+	**−10.402 9**
f_{23}	−9.481 6	=	−10.536 4	=	−10.536 4	=	−10.536 1	=	**−10.536 4**

或与对比算法无显著差异,除函数 f_{15} 外,所提出算法在其他函数中均寻找到全局最优解,从而再次验证了改进算法具有优秀的搜索性能,能够适应不同类型的函数优化问题.

根据第2.6节分析可知,所提出改进策略并未增加算法复杂度,但是实验结果表明所提出算法的爬山能力、收敛速度和寻优精度均有显著提升,从而表明所提出算法的搜索效率有所提高,在相同计算代价下能够求得更优的解.

4　结　论

本文针对混合蛙跳算法的不足,通过增加局部更新的信息实现子群信息的充分交互,引入解空间反向跳跃思想提升算法跳出局部最优的能力,采用局部最优解变异降低子群内部同化速率,利用各子群最优解交叉变异后混洗的方式加深子群间信息交互深度,从而加快收敛速度. 综合以上策略,本文提出了一种基于解空间反向跳跃和信息交互强化的新型混合蛙跳算法. 以平均收敛值、方差、收敛步数作为测度,分别验证了每个策略在不同类型函数问题上的有效性,通过与4种具有代表性的进化算法进行比较,验证了所提出算法的搜索精度优于比较算法,能够适应不同类型的函数优化问题.

同时,本文也存在一些不足. 例如,针对不同规模的子群,参与信息交互的个体数量选择需要进一步讨

论和分析;当解的维度较小时,策略SFLA_VLS对局部最优解深度变异的概率增加,本文尚未分析该策略在不同维度下对算法性能的影响. 下一步将针对本文的不足和算法参数设置等理论问题作进一步研究.

参考文献(References)

[1] Kennedy J, Eberhart R C. Particle swarm optimization[C]. Proceeding of IEEE International Conference on Neural Networks. Perth: IEEE, 1995: 1942-1948.

[2] Dorigo M, Maniezzo V, Colorni A. Ant system: Optimization by a colony of cooperating agents[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2002, 26(1): 29-41.

[3] Karaboga D. An idea based on honey bee swarm for numerical optimization[R]. Kayseri: Engineering Faculty Computer Engineering Department, Ereiyes University, 2005.

[4] Eusuff M M, Lansey K E. Optimization of water distribution network design using the shufed frog leaping algorithm[J]. Tournal of Water Resours Planning and Management, 2003, 129(3): 210-225.

[5] 骆剑平, 李霞, 陈泯融. 混合蛙跳算法的Markov模型及其收敛性分析[J]. 电子学报, 2010, 38(12): 2875-2880.
(Luo J P, Li X, Chen M R. The markov model of shuffled frog leaping algorithm and its convergence analysis[J]. Acta Electronica Sinica, 2010, 38(12): 2875-2880.)

[6] Kong L P, Pan J S, Chu S C, et al. Directional shued frog leaping algorithm[C]. Advances in Smart Vehicular Technology, Transportation, Communication and Applications. Cham: Springer, 2017: 257-264.

[7] Zhen Z Y, Wang D B, Liu Y Y. Improved shuffled frog leaping algorithm for continuous optimization problem[C]. IEEE Congress on Evolutionary Computation. Trondheim: IEEE, 2009: 2992-2995.

[8] Sharma P, Sharma N, Sharma H. Locally informed shued frog leaping algorithm[C]. Proceedings of 6th International Conference on Soft Computing for Problem Solving. Patiala: Springer, 2017: 141-152.

[9] Liu H R, Yi F Y, Yang H L. Adaptive grouping cloud model shuffled frog leaping algorithm for solving continuous optimization problems[J]. Computational Intelligence and Neuroscience, 2016: 1-8.

[10] Merz P. Memetic algorithms for the traveling salesman problem[J]. Complex Systems ,1997, 13(4): 297-345.

[11] Englert M, Röglin H, Vöcking B. Worst case and probabilistic analysis of the 2-opt algorithm for the TSP[J]. Algorithmica, 2014, 68(1): 190-264.

[12] Liu C, Niu P F, Li G Q. Enhanced shufed frog-leaping algorithm for solving numerical function optimization problems[J]. Journal of Intelligent Manufacturing, 2018, 29(5): 1133-1153.

[13] 汪超, 王丙柱, 岑豫皖, 等. 基于多样性全局最优引导和反向学习的离子运动算法[J]. 控制与决策, 2020, 35(7): 1584-1596.
(Wang C, Wang B Z, Cen Y W, et al. Ions motion optimization algorithm based on diversity optimal guidance and opposition-based learning[J]. Control and Decision, 2020, 35(7): 1584-1596.)

[14] Mirjalili S, Lewis A. The whale optimization algorithm[J]. Advances in Engineering Software, 2016, 95: 51-67.

[15] Mirjalili S, Gandomi A H, Mirjalili S Z, et al. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems[J]. Advances in Engineering Software, 2017, 114: 163-191.

[16] Saremi S, Mirjalili S, Lewis A. Grasshopper optimisation algorithm: Theory and application[J]. Advances in Engineering Software, 2017, 105: 30-47.

[17] Arora S, Singh S. Butterfly optimization algorithm: A novel approach for global optimization[J]. Soft Computing, 2019, 23(3): 715-734.

[18] Li G Q, Niu P F, Xiao X J. Development and investigation of efcient articial bee colony algorithm for numerical function optimization[J]. Applied Soft Computing, 2012, 12(1): 320-332.

[19] Rao R V, Patel V. An improved teaching-learning-based optimization algorithm for solving unconstrained optimization problems[J]. Scientia Iranica, 2013, 20(3): 710-720.

作者简介

申晓宁(1981−), 女, 教授, 博士, 从事计算智能、多目标优化、调度技术等研究, E-mail: sxnystsyt@sina.com;

黄遥(1993−), 男, 硕士生, 从事群智能计算、调度技术的研究, E-mail: yao.h1015@qq.com;

游璇(1996−), 女，硕士生, 从事智能计算、多目标优化的研究, E-mail: 563752923@qq.com;

王谦(1990−), 男，硕士生, 从事智能计算、机器学习的研究, E-mail: 644314380@qq.com.

(责任编辑：郑晓蕾)