基于混沌优化策略的 SFLA 算法*

张海玉, 刘　军, 刘志都
(南阳师范学院 计算机与信息技术学院, 河南 南阳 473061)

摘　要: 针对基本混合蛙跳算法的缺陷,提出了一种基于混沌优化策略的改进混合蛙跳算法(SFLA)。在青蛙更新策略中引入自适应扰动机制,平衡了算法搜索深度,并利用高斯变异算子代替随机更新操作,提高了算法搜索速度;在全局迭代中借鉴混沌优化策略思想,以概率形式对最优个体进行优化,避免了族群陷入局部最优,并证明了改进算法以概率 1 收敛于全局最优解。最后用 MATLAB 对测试函数进行了仿真,仿真结果表明改进的混合蛙跳算法在收敛速度、优化精度上有较大改善。

关键词: 混沌优化策略; 混合蛙跳算法; 收敛性; MATLAB
中图分类号: TP18;TP301.6　　**文献标志码**: A　　**文章编号**: 1001-3695(2013)06-1708-04
doi:10.3969/j.issn.1001-3695.2013.06.027

Improved SFLA based on chaos optimization strategy

ZHANG Hai-yu, LIU Jun, LIU Zhi-du
(*College of Computer & Information Technology*, *Nanyang Normal University*, *Nanyang Henan* 473061, *China*)

Abstract: Based on the problems of the shuffled frog leaping algorithm (SFLA), this paper presented the improved SFLA. It proposed the mechanism based on mutation idea in differential evolution in order to balance algorithm search depth, and adopted Gaussian mutation to replace the original SFLA update strategy, which improved the convergence speed. Based on the chaos optimization strategy, it optimized the best solution in the form of probability, and avoided local optimum. It proved the improved SFLA to be converged to the global optimization solution with probability one. Finally with MATLAB, it implemented test function simulations. The simulation results show that the improved SFLA has better performance in the convergence speed and optimization precision.

Key words: chaos optimization strategy; shuffled frog leaping algorithm(SFLA); convergence; MATLAB

0　引言

混合蛙跳算法[1](SFLA)是一种基于群体智能的启发式计算技术,其结合了粒子群(PSO)算法和 Memetic 算法的优点,具有概念简单、计算速度快、搜索能力强、容易实现等特点,已被成功应用于函数优化[2]、TSP[3]、CVRP[4]、车间流水调度[5]等目标优化问题。相关研究表明,随着迭代次数的增加,SFLA 种群多样性降低,算法容易收敛于局部最优解,而且求解精度较低。为了提高 SFLA 解决复杂优化问题的能力,国内外学者进行了大量研究。提高 SFLA 收敛速度,避免其陷入局部最优是该算法的研究热点。有关混合蛙跳算法的改进研究主要集中在局部搜索策略和全局信息交换两方面。文献[6]将搜索加速因子引入到子族群局部搜索过程中,提高了算法的寻优能力;文献[7]借鉴分子动力学模拟思想,提出了一种基于分子动力学模拟的改进混合蛙跳算法,增强了算法后期跳出局部极值的能力;文献[8]提出了一种基于领域正交叉算子的混合蛙跳算法,通过引入领域正交叉算子,加快了算法收敛速度;文献[9]在全局信息交换过程中引入变异算子,并利用模糊控制器调整其变异尺度,提高了算法的求解精度。这些做法一定程度上改善了混合蛙跳算法的性能,但效果并不是很理想。文献[10]提出了一种改进型蛙跳算法,用来解决大量非线性、不可微和多峰值复杂问题的优化,但当空间维数比较大时,求解目标函数需要的时间会较长。

混沌优化算法(chaos optimization algorithm, COA)是一种基于混沌运动的智能优化算法[11],具有遍历性、规律性、全局搜索能力强等特点。本文将混沌优化算法引入到 SFLA 中,以概率形式对最优个体进行优化,以增强算法寻优能力,同时在青蛙更新策略中加入自适应差分扰动机制,并利用高斯变异算子代替随机更新操作,来提高算法收敛速度及求解精度。

1　混合蛙跳算法

2003 年,Eusuff 等人借鉴混合竞争进化的思想,在 PSO 算法的基础上提出了混合蛙跳算法。算法中每只青蛙被看做模因载体,通过不同青蛙个体间的交流实现模因进化。SFLA 将青蛙族群划分成规模相同的模因组,每个模因组独立执行局部搜索策略。当完成一定次数的局部迭代后,所有模因组内青蛙重新混合、排序和划分模因组,再次进行局部搜索,使得整个族群之间能够交流信息。局部搜索和全局信息交换的平衡策略使得 SFLA 能够收敛于全局最优解[12]。针对基本混合蛙跳算法的缺陷,本文提出了一种基于混沌优化策略的改进混合蛙跳

收稿日期: 2012-10-22; **修回日期**: 2012-11-24　　**基金项目**: 河南省基础与前沿技术研究基金资助项目(112300410225);河南省重点攻关基金资助项目(112102210408)

作者介绍:张海玉(1981-),女,山西新绛人,讲师,硕士,主要研究方向为数据库、数据挖掘、算法研究(hayuz2012@163.com);刘军(1962-),男,河南西峡人,副教授,主要研究方向为网络工程;刘志都(1957-),男,河南南阳人,教授,主要研究方向为网络工程.

算法(SFLA)。

1)划分族群

对于 n 维的目标优化问题,其解空间为 $S=\{X\mid X=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}\wedge L_i\leqslant x_i\leqslant U_i,i=1,2,\cdots,n\}\subseteq \mathrm{R}^n$,随机生成 F 只青蛙粒子组成初始群体,则第 t 代青蛙群体可以描述为 $P_t=\{X_1(t),\cdots,K_k(t),\cdots,X_F(t)\mid k=1,2,\cdots,F\}$。在 SFLA 每只青蛙代表问题的一个解,因此第 k 只青蛙的位置为 $X_k(t)=\{x_{k1}(t),x_{k2}(t),\cdots,x_{kn}(t)\}$,将 F 只青蛙按适应度值 $f[X_k(t)]$ 降序排列,并平均分配到 M 个族群中,每个族群的青蛙数为 N,显然有 $F=M\times N$。设 E_j 表示第 j 个族群,则有

$$E_j=\{X_{j+M(l-1)}(t)\in P_t\mid 1\leqslant l\leqslant N\}\quad 1\leqslant j\leqslant M \tag{1}$$

2)局部搜索

SFLA 算法规定只对每个族群中适应度值最差的解 $X_w(t)$ 进行更新,其更新策略(图 1(a))为

$$X_{\mathrm{new}}(t)=X_w(t)+r\times[X_b(t)-X_w(t)] \tag{2}$$

其中:$r\in[0,1]$ 且为随机数;$X_b(t)$ 为族群中适应度最好的解。如果 $X_{\mathrm{new}}(t)$ 适应度优于 $X_w(t)$,则用 $X_{\mathrm{new}}(t)$ 替代 $X_w(t)$;否则用 $X_g(t)$($X_g(t)$ 为整个青蛙群体中适应度最好的解)代替 $X_b(t)$ 重新执行更新策略。若 $X_{\mathrm{new}}(t)$ 仍然不能优于 $X_w(t)$,则随机生成青蛙取代 $X_w(t)$。每个族群重复执行上述过程,直到满足局部搜索次数为止。

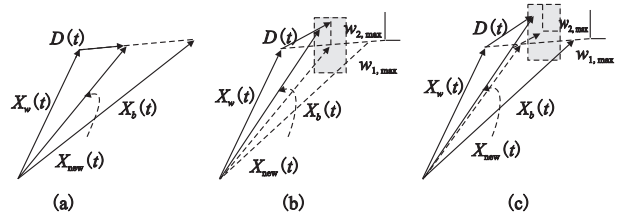


图1　青蛙更新策略

3)全局信息交换

当所有族群完成局部搜索后,将青蛙粒子重新混合,得到第 $t+1$ 代青蛙群体为 $P_{t+1}=\{X_1(t+1),\cdots,X_k(t+1),\cdots,X_F(t+1)\}$。计算青蛙适应度值 $f[X_k(t+1)]$,按适应度降序排列,更新群体最优解 $X_g(t+1)$,并重新执行划分族群、局部搜索操作,直到满足算法终止条件为止,如图 2 所示。

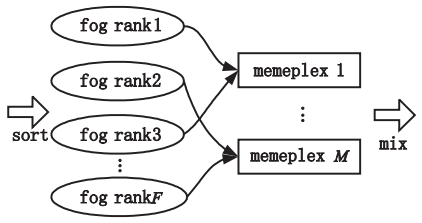


图2　混合蛙跳算法示意图

2　改进的混合蛙跳算法

在混合蛙跳算法自然模因进化过程中,局部搜索和全局信息交流的方式能够使得较差的青蛙获得较好青蛙的模因,从而向全局最优方向进化。然而在局部搜索机制中,如果只利用族群或全局最优解的模因信息会限制局部搜索区域,不仅降低了算法收敛速度,而且会使算法易于陷入局部最优,导致早熟现象[13]的发生。为了提高算法收敛速度、避免局部最优,本文对青蛙更新策略及局部搜索机制进行了改进,提高了算法性能。

2.1　青蛙更新策略改进

为提高局部搜索空间,本文参考文献[13]给出青蛙粒子更新策略(图 1(b))为

$$D(t)=r\times c\times(X_b(t)-X_w(t))+W \tag{3}$$

$$W=[r_1\times w_{1,\max},r_2\times w_{2,\max},\cdots,r_n\times w_{n,\max}]^{\mathrm{T}} \tag{4}$$

$$X_{\mathrm{new}}(t)=\begin{cases}X_w(t)+D(t) & \text{if } \|D(t)\|\leqslant D_{\max}\\ X_w(t)+\dfrac{D(t)}{\sqrt{D^T(t)D(t)}}D_{\max} & \text{if } \|D(t)\|>D_{\max}\end{cases} \tag{5}$$

其中:$c\in[1,2]$ 且为常数;$r_i\in[-1,1]$ 且为随机数;$w_{i,\max}$ 为第 i 维搜索空间最大不确定性。该青蛙更新策略在一定程度上加大了算法搜索范围。但是这种更新策略仍旧只采用 $X_b(t)$,因此并没有提高青蛙学习能力。本文借鉴 PSO 粒子更新思想,在青蛙模因交流过程中引入上次青蛙更新移动距离 $D'(t)$ 及历史最优个体 $X_g'(t)$,使得青蛙能够延续上次更新惯性,而且学习了历史最优个体的模因信息,避免了早熟现象的发生,如图 2(c)所示,改进后的青蛙移动步长为

$$D(t)=R[r_2D'(t)+r_3(X_g'(t)-X_w(t))]+rc(X_b(t)-X_w(t))+W \tag{6}$$

其中:r_2、$r_3\in[0,1]$ 且为随机数;R 为权重因子。

2.2　自适应权重因子

在混合蛙跳算法迭代后期,随着迭代次数不断增加,青蛙个体之间差异不断减小,如果 R 不发生改变,将影响算法的收敛速度,并降低其求解精度。为此,本文提出一种自适应权重因子 R:

$$R=R_L+(R_U-R_L)\times a \tag{7}$$

$$a=\mathrm{e}^{-20\times(t/t_{\max})^S} \tag{8}$$

其中:R_L、R_U 为权重因子的初始值和结束值;S 为大于 1 的整数。在混合蛙跳算法中,随着迭代次数增加,R 不断减小,最终取值为 R_U。通过调节 S,可以改变 R 的函数值曲线(图 3),S 的大小根据青蛙群体规模及最大迭代次数 $t_{\max}$ 确定。

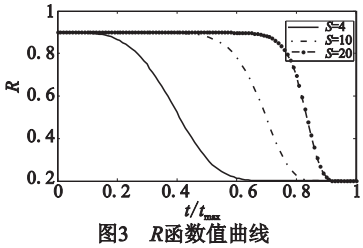


图3　R函数值曲线

2.3　高斯变异(Gauss mutation)

基本混合蛙跳算法中,如果经过局部或全局最优解更新后的个体适应度仍然比 $X_w(t)$ 差,则随机产生新的个体替代 $X_w(t)$。这种随机产生个体的做法增加了算法盲目性,降低了算法收敛速度。为了充分利用当前的信息进行扰动,本文采用高斯变异取代随机产生青蛙个体的操作。高斯变异是指在原有个体上加一个服从高斯分布的随机扰动,其定义如下:

$$X'_{\mathrm{new}}=X_b(t)+X_b(t)\times N(0,1) \tag{9}$$

其中:$N(0,1)$ 是服从均值为 0、均方差为 1 的高斯分布。

2.4　最优个体优化策略

随着迭代次数增加,青蛙个体差异逐渐减少,当青蛙与全局最优解的位置十分靠近时,群体多样性逐渐丧失,导致青蛙陷入局部最优解,因而产生了早熟现象。为了避免种群局部最优,关键要控制青蛙个体后期趋同性,为此本文借鉴混沌优化算法思想,以概率选择的形式对族群最优个体进行混沌优化,避免了算法早熟现象的发生。

1)混沌优化算法

混沌优化算法把混沌状态引入到优化变量中,并将混沌范围扩大到优化变量的取值范围,利用混沌变量进行搜索。由于 Logistic 映射具有良好的随机性能,因此常作为混沌优化模型。其迭代方程为

$$x_{n+1} = \mu \times x_n \times (1 - x_n) \quad 0 < x_n < 1 \tag{10}$$

其中:μ 为混沌变量。文献[14]指出,当 μ 取值范围为[3.85,4]时,其混沌序列的遍历性比较明显,算法性能良好。混沌算法实现过程为:

a)对 x_n 赋予 n 个初值,其中 n 为问题解空间的维数。

b)对 x_n 利用载波公式将其映射到混合蛙跳算法优化变量的取值范围内,设 x_L、x_U 分别为青蛙 $X_k(t)$ 位置变量的最小值与最大值,载波式为

$$x_{kn}(t) = x_L + (x_u - x_L) \times x_n \tag{11}$$

c)混沌搜索。

d)判断是否满足终止条件,若满足终止算法并输出结果;否则迭代次数加 1,转向 a)。

2)最优个体优化策略

在混合蛙跳算法全局信息交流过程中,利用概率选择的方式对群体最优解 $X_g(t)$ 进行优化,可以避免算法陷入局部最优。其基本思路为:在算法迭代初期,群体差异较大,以较小概率选择 $X_g(t)$ 进行混沌优化;随着迭代次数增加,青蛙差异性降低,此时以较大概率选择 $X_g(t)$,并进行混沌优化,其选择概率为

$$p_g = \begin{cases} 0 & CR > p \\ 1 & CR \leqslant p \end{cases} \tag{12}$$

$$p = \ln t/(1 + \ln t) \tag{13}$$

其中:CR 为[0,1]的随机数。

2.5 改进混合蛙跳算法流程

图 4 给出了改进后的混合蛙跳算法流程。

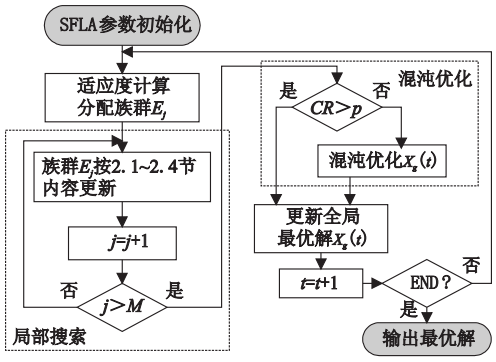


图4 改进混合蛙跳算法流程

3 改进混合蛙跳算法收敛性分析

为了分析方便又不失一般性,在改进的混合蛙跳算法更新策略中,只考虑 $\| D(t) \| \leqslant D_{\max}$ 的情况。因此,根据青蛙更新策略,在第 t 代有

$$X_{\text{new}}(t) = (1 - Rr_3 - rc)X_w(t) + D \tag{14}$$

$$D = Rr_2 D'(t+1) + r_3 X_g(t) + rcX_b(t) + w \tag{15}$$

式(14)为标准的离散线性系统的状态方程。由于 $|1 - Rr_3 - rc| \in [0,1]$,所以该系统是稳定的,即 $X_w(t)$ 是收敛的,但不一定收敛于全局最优解。文献[15]基于齐次有限 Markov 链证明了改进算法的收敛性,本文是在文献[15]的基础上将自适应权重因子及高斯变异引入局部更新中,显然没有改变算法的齐次有限 Markov 链的性质,即加入自适应权重因子及高斯变异 SFLA 以概率 1 收敛于全局最优解。考虑混沌优化算法,其实质可以描述为

$$X_g = \max\{X'(t), X(t)\} \tag{16}$$

其中:$X_g(t)$ 是最终群体最优个体;$X(t)$ 为混沌后的最优个体;$X'(t)$ 为混沌前的最优个体。可见加入混沌优化算法仍为 Markov 链,并且从改进混合蛙跳算法的思路可以得出:适应度低的族群能以一定概率转移到适应度相同或更高的族群,但适应度高的族群不能转移到适应度低的族群。

定义 1 设 $H = \{S_t\}$ 为青蛙族群迭代过程中所有可能族群状态的集合,定义 p_{ij} 为 s_i 到状态 s_j 的转移概率。根据上述结论有 $p_{ij} = 0, i > j; p_{ij} > 0, i \leqslant j$。设状态转移矩阵为 $\boldsymbol{R}$,因此有

$$\boldsymbol{R}^{\mathrm{T}} = \begin{bmatrix} p_{11} & 0 & \cdots & 0 \\ p_{12} & p_{22} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ P_{1t} & p_{2t} & \cdots & p_{tt} \end{bmatrix} = \begin{bmatrix} C^{\mathrm{T}} & 0 \\ Q^{\mathrm{T}} & T^{\mathrm{T}} \end{bmatrix} \tag{17}$$

根据文献[15]提出的引理 1,有 $\boldsymbol{R}^{\mathrm{T}\infty} = \lim\limits_{t\to\infty} \boldsymbol{R}^{\mathrm{T}t} = \begin{bmatrix} C^{\mathrm{T}\infty} & 0 \\ Q^{\mathrm{T}\infty} & 0 \end{bmatrix}$,因此 $\boldsymbol{R}^{\mathrm{T}\infty}$ 为稳定随机矩阵,即改进后混合蛙跳算法以概率 1 收敛于全局最优解。

4 实验仿真

4.1 实验仿真

为了验证本文改进算法的可行性及其性能,将本文提出的加入高斯变异与混沌优化算法的混合蛙跳算法(Gauss mutation and CAO SFLA,GCSFLA)与文献[6]提出的 MSFLA、基本混合蛙跳算法进行对比。GCSFLA 参数设置为:$F = 200$,$M = 20$,$N = 10$,局部迭代 10 次,全局迭代 500 次。上述参数设定参照文献[16],该文献指出采用此类参数,算法在求解结果和求解时间上有较好表现。在函数选择上本文选取四个经典测试函数作为实验仿真对象,分别是 Sphere 函数 f_1、Rosenbrock 函数 f_2、Griewank 函数 f_3 和 Schaffer F7 函数 f_4。四个测试函数的理论最优值为 0,其表达式如下:

$$f_1(x_i) = \sum_{i=1}^{n} x_i^2 \quad x_i \in [-100, 100] \tag{18}$$

$$f_2(x_i) = \sum_{i=1}^{n} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \quad x_i \in [-5, 5] \tag{19}$$

$$f_3(x_i) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \left(\cos \frac{x_i}{\sqrt{i}} \right) \quad x_i \in [-500, 500] \tag{20}$$

$$f_4(x_i) = \sum_{i=1}^{n} (x_i^2 + x_{i+1}^2)^{0.25} [\sin^2(50(x_i^2 + x_{i+1}^2)^{0.1}) + 1] \quad x_i \in [-50, 50] \tag{21}$$

在 MATLAB 仿真平台上,采用三种算法分别对四个测试函数进行仿真,运行 100 次。为了结果的准确性,取 50 次最好实验结果的平均最优值为输出最优结果值。表 1 给出了实验仿真结果。

4.2 性能分析

从表 1 结果可以看出,对于简单函数 f_1,三种算法在不同维数下都表现出较强的全局寻优能力,最优解求解精度都很高;对于多峰函数,随着维数的增加,算法的函数优化性能降低,特别是在 30 维的情况下,SFLA 与 MSFLA 的求解精度明显偏低,而 GCSFLA 依然保持较好的寻优能力。为了进一步比较

三种算法跳出局部最优的能力,提出算法运算成功率 V 概念:每次算法得出的最优解如果在精度 q 范围内,则称此次运算成功,所有实验结果中达到实验精度的运算次数与总的实验次数的比值称为运算成功率。为此,针对四个测试函数,其维数设定为 30 维,分别运行 100 次,仿真结果如表 2 所示。

表 1　实验对比结果

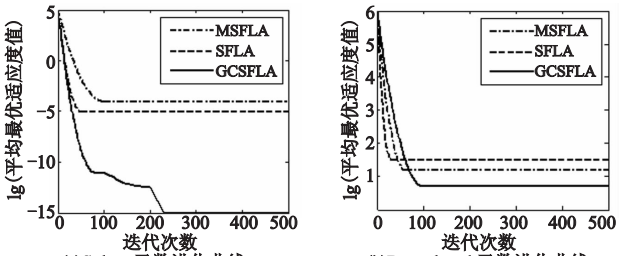
函数	算法	函数维数 n		
		10	20	30
f_1	GCSFLA	0	0	0
	MSFLA	0	1.0e-5	8.2e-5
	SFLA	0	1.0e-4	5.4e-4
f_2	GCSFLA	6.8e-3	3.247 6	5.673 8
	MSFLA	5.346 3	13.473 8	16.342 7
	SFLA	10.437 2	17.635 2	22.634 1
f_3	GCSFLA	1.1e-2	2.3e-4	1.0e-5
	MSFLA	5.3e-2	1.8e-2	3.2e-4
	SFLA	3.2e-2	1.7e-2	4.2e-3
f_4	GCSFLA	1.0e-5	2.3e-1	3.145 2
	MSFLA	2.3e-3	1.367 2	13.478 3
	SFLA	4.6e-2	5.720 1	16.382 1

表 2　算法运算成功率

F	q	算法	V/%	F	q	算法	V/%
f_1	10^{-3}	GCSFLA	100	f_3	10^{-2}	GCSFLA	100
		MSFLA	100			MSFLA	100
		SFLA	100			SFLA	64
f_2	25	GCSFLA	96	f_4	10	GCSFLA	93
		MSFLA	87			MSFLA	45
		SFLA	45			SFLA	18

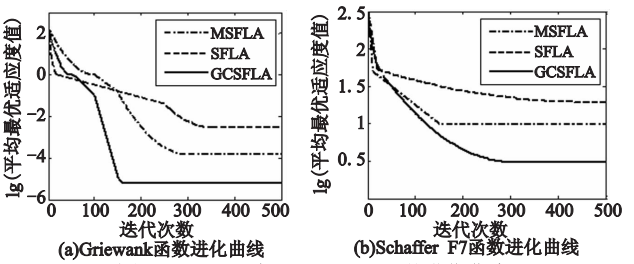
从表 2 可以清楚地看出,GCSFLA 跳出局部最优解的能力要强于其他两种算法,特别是对于维数较高的复杂函数,GCSFLA 表现出了良好的性能。

图 5 和 6 给出了三种算法在不同测试函数上的平均适应度值曲线,测试函数的维数仍然设定为 30。为了方便观察结果,对平均最优适应度值取对数。



(a)Sphere函数进化曲线　(b)Rosenbrock函数进化曲线

图5　Sphere与Rosenbrock函数进化曲线



(a)Griewank函数进化曲线　(b)Schaffer F7函数进化曲线

图6　Griewank与Schaffer F7函数进化曲线

从图 5、6 可以看出,GCSFLA 在精度上明显优于其他两种算法,特别是对于复杂函数优化问题,GCSFLA 寻优优势明显,这是因为 GCSFLA 算法在青蛙更新策略中引入了自适应权重因子及高斯变异,提高了算法搜索速度,并且混沌优化算法的引入使得改进的混合蛙跳算法能够避免局部最优,求解精度更高。

5　结束语

本文针对基本混合蛙跳算法的不足,提出了自适应权重因子,改进了青蛙更新策略,采用高斯变异替代了随机生成青蛙的方式,提高了算法收敛速度,引入了混沌优化策略,避免了算法陷入局部最优解,并证明了改进的混合蛙跳算法以概率 1 收敛于全局最优,仿真结果表明改进的混合蛙跳算在算法精度、避免局部最优上有明显改善。

参考文献:

[1] EUSUFF M M, LANSEY K E. Optimization of water distribution network design using the shuffled frog leaping algorithm[J]. Water Resources Planning and Management,2003,129(3): 210-225.

[2] AMIRI B, FATHIAN M, MAROOSI A. Application of shuffled frog leaping algorithm on clustering[J]. The International Journal of Advanced Manufacturing Technology,2009,45(1-2):199-209.

[3] 罗雪晖,杨烨,李霞. 改进混合蛙跳算法求解旅行商问题[J]. 通信学报,2009,30(7):130-135.

[4] 万博,卢昱,陈立云,等. 求解 CVRP 的改进混合蛙跳算法研究[J]. 计算机应用研究,2011,28(12):4503-4506.

[5] ALIREZA R V, ALI H M. Solving a bicriteria permutation flow shop problem using shuffled frog-leaping algorithm[J]. Soft Computing, 2008,12(5):435-452.

[6] ELBELTAGI E, HEGAZY T, GRIERSON D. A modified shuffled frog-leaping optimization algorithm applications to project management[J]. Structure and Infrastructure Engineering,2007,3(1) :53-60.

[7] 张满丹,胡峰,赵力,等. 基于分子动力学模拟的改进混合蛙跳算法[J]. 数据采集与处理,2012,27(3):327-332.

[8] 孟庆莹,王联国. 基于领域正交叉算子的混合蛙跳算法[J]. 计算机工程与应用,2011,47(36): 54-56,85.

[9] 葛宇,王学平,梁静. 改进的混合蛙跳算法[J]. 计算机应用,2012,32(1):234-237.

[10] 许金元. 混合型蛙跳算法及其应用研究[J]. 计算机应用研究,2011,28(8):2835-2837.

[11] 王耀南,刘良江,周博文,等. 一种基于混沌优化算法的 PCB 板元件监测方法[J]. 仪器仪表学报,2010,31(2):411-415.

[12] RAHIMI-VAHED A, MIRZAEI A H. A hybrid multi-objective shuffled frog-leaping algorithm for a mixed model assembly line sequencing problem[J]. Computers and Industrial Engineering, 2007, 53 (4): 642-666.

[13] ZHANG Xun-cai, HU Xue-mei, CUI Guang-zhao, *et al*. An improved shuffled frog leaping algorithm with cognitive behavior[C]// Proc of the 7th World Congress on Intelligent Control and Automation. New York: IEEE Press, 2008: 6197-6202.

[14] 徐洪丽,钱旭,岳训,等. 一种新的基于 Logistic 混沌映像的自适应混沌蚁群优化算法求解动态车辆路径问题[J]. 计算机应用研究,2012,29(6):2058-2060.

[15] 贺毅朝,曲文龙,许冀伟. 一种改进的混合蛙跳算法及其收敛性分析[J]. 计算机工程与应用,2011,47(22):37-40.

[16] ELBCTAGI E, HEGAZY T, GRIEMON D. Comparison among five evolutionary-based optimization algorithms [J]. Advanced Engineering Informatics,2005,19(1):43-53.

[17] HUYNH T H. A modified shuffled frog leaping algorithm for optimal tuning of multivariable PID controllers[C]//Proc of IEEE International Conference on Industrial Technology. New York: IEEE Press, 2008: 1-6.