**Software Engineering and Applications** 软件工程与应用, 2023, 12(6), 949-957
Published Online December 2023 in Hans. https://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2023.126093

Hans汉斯

# 基于多层次信息反馈的混合蛙跳算法

**刘华军**

景德镇陶瓷大学信息工程学院，江西 景德镇

收稿日期：2023年11月14日；录用日期：2023年12月20日；发布日期：2023年12月28日

## 摘　要

**基于多层次信息反馈的混合蛙跳算法(Shuffled Frog Leaping Algoriyhm based on the interation of Multi-level information, MSFLA)，吸收遗传算法的交叉算子及粒子群算法(PSO)的粒子进化方式，将整个优化过程划分为标准混合蛙跳优化层、青蛙进化与学习层、外部档案信息交换层。混合蛙跳优化层保证青蛙进行正常的局部搜索优化(蛙跳算法)；青蛙进化与学习层保证青蛙每次迭代结束时都能得到更好的自身位置(PSO粒子进化方式)；外部档案信息交换层可以保证青蛙种群获得最优解(交叉算子)。通过各层次之间的信息交流，提高算法的性能。从实验结果对比能够得出，改进后的MSFLA算法可以有效地改善早熟收敛问题，具有更好的收敛速度和更高的寻优精度。**

## 关键词

**混合蛙跳算法，遗传算法、交叉算子，粒子群优化算法**

# The Shuffled Frog Leaping Algorithm Bases on the Interation of Multi-Level Information

**Huajun Liu**

School of Information Engineering, Jingdezhen Ceramic University, Jingdezhen Jiangxi

Received: Nov. 14th, 2023; accepted: Dec. 20th, 2023; published: Dec. 28th, 2023

文章引用: 刘华军. 基于多层次信息反馈的混合蛙跳算法[J]. 软件工程与应用, 2023, 12(6): 949-957.
DOI: 10.12677/sea.2023.126093

## Abstract

**The Shuffled Frog Leaping Algorithm bases on the interation of multi-level information (MSFLA), which assimilates the Crossover of the Genetic Algorithm and the evolutional pattern of Particle Swarm Optimization (PSO). The whole optimization process includes three aspects—the optimizing layer of standardly blended frog leaping, the layer of flog evolution and learning, and the commutative layer of external file information. The first layer guarantees the frog to normally perform local optimal search (Fog Leaping Algorithm) and the second layer ensures that the frog can get a better self-position at the end of each iteration (the evolutional pattern of PSO). Moreover, the last layer can assure that the fog population gains the most optimal solution (the Crossover).Through the exchange of information between all levels, improve the performance of the algorithm. Experimental results show that the improved MSFLA can avoids premature convergence effectively, and has better convergence results, higher accuracy.**

## Keywords

**Shuffled Frog-Leaping Algorithm, Genetic Algorithm, Crossover, Particle Swarm Optimization, PSO**

Copyright © 2023 by author(s) and Hans Publishers Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
Open Access

## 1. 引言

在实际应用中，非线性、非连续、不可微、多变量甚至混杂系统出现的越来越多，经典优化方法已经不可能有效求解，为寻找复杂优化问题的解决方案，20 世纪 90 年代以来，学者们提出了一些新的迭代搜索算法——群智能优化算法。混合蛙跳算法(Shuffled Frog Leaping Algorithm, SFLA) [1]就是其中一种，它是 Eusuff 等人在 2003 年提出的一种元启发式协同搜索群智能算法。

SFLA 基于青蛙觅食的行为，结合遗传基因的模因演算算法和基于群体觅食行为的粒子群优化算法的优点，借鉴多种群混合进化的思想、将种群划分成若干个子群，所有子群独立地同时地进行局部模因进化，而后通过混合子群达到全局信息交换的目的，从而推进整个种群向最优的区域搜索。在局部迭代寻优的过程中，根据局部最优个体以及全局最优个体的差异对最差个体进行更新[2]。

## 2. 算法基本理论

### 2.1. 标准混合蛙跳算法

SFLA 算法是一种受青蛙群体寻找食物时按子群分类进行信息交换过程得到启发而产生的仿生智能优化算法。

### 2.2. 标准混合蛙跳算法的数学模型

对于一个 d 维的函数优化问题：

$$\begin{cases}\min f(x) \\ s.t.x \in [l,u]\end{cases} \tag{1}$$

其中 $[l,u] := \{x \in \Re^d \mid l_k \le x_k \le u_k, k = 1,2,...,d\}$ 。

利用 SFLA 算法求解此类问题时，主要有以下四个步骤：

1) 种群初始化：从可行域中随机生成 F 个青蛙作为初始种群，设第 $i$ 个青蛙表示为 $x(i) = (x_1^i, x_2^i, ..., x_d^i)$，则每个 $x(i)$ 为问题的一个解。

2) 子群划分：计算每个解的目标函数值 $f(i)$，对目标函数值进行排序，记种群中目标函数值最优的解为 $X_g$。然后将整个种群分为 $m$ 个子群。在每个子群中有 $n$ 个解(即 $F = m*n$)，记每个子群目标函数最好和最差的解分别为 $X_b$ 和 $X_w$。

3) 局部搜索：根据模因算法在各子群内部进行进化运算，每次迭代只改变子群中目标函数值最差的解 $X_w$，其更新公式为：

$$D_i = rand(\ ) * (x_b - x_w), i = 1,2,...,m \tag{2}$$

$$x'_w = x_w + D_i, |D_i| \le D_{\max}, i = 1,2,...,m \tag{3}$$

其中 $rand(\ )$ 是 0 到 1 之间的随机数，$D_i$ 为青蛙的移动步长，$D_{\max}$ 表示青蛙所允许改变位置的最大值。在经过更新后，如果得到的子群最差解 $X'_w$ 优于原来的最差解 $X_w$，则用 $X'_w$ 替换 $X_w$，如果 $X_w$ 没有得到改进，则用 $X_g$ 代替 $X_b$ 后利用公式(2)、(3)进行局部搜索，如仍没有改进，则随机生成一只新解直接替代 $X_w$，重复上述局部搜索直到达到子群迭代次数。

4) 全局混合：将所有子群内的蛙重新混合，更新整个种群中目标函数最优解 $X_g$，重新进行步骤 2 和 3，直到达到指定的迭代次数或满足定义的收敛条件。

### 2.3. 标准 SFLA 的局限性

由以上 4 步骤可见，SFLA 算法结合了局部进化更新和全局信息交换，在进行局部搜索的基础上更新全局最优解，具有参数少，全局寻优能力较强，且收敛速度较快等优点，故在实际中有很多的应用，可以用来优化各类连续或离散的问题[3] [4]，也可用于求组合优化和复杂函数的最优解[5]。

但由于标准 SFLA 更新方式较为单一，会造成种群多样性不强，收敛速度降低，而且容易陷入局部最优导致算法早熟的问题。并且在个体更新的过程中忽略了同组内其它个体，以及种群中其它个体的信息交流，可能会错过可能更优秀的解或组合[6]。

针对上述问题，本文通过引入粒子群算法(PSO) [7]的粒子进化方式以及遗传算法[8]的交叉算子来改善。提出了基于多层次信息反馈的混合蛙跳算法(MSFLA)。

## 3. 基于多层次信息反馈的混合蛙跳算法

在 MSFLA 算法中，整个算法分为标准混合蛙跳优化层、青蛙进化与学习层、外部档案信息交换层三个层次。

在每次迭代优化初始阶段中，通过标准蛙跳算法更新位置的方法获得青蛙的新位置，并且判断青蛙新旧位置的优劣情况。如果青蛙新位置没有优于旧位置，则进入青蛙进化与学习层。

在 MSFLA 算法中，通过引入粒子群算法中的粒子进化方式[9]来实现青蛙进化和青蛙对全局最优的学习，获取更优的新位置，从而增加青蛙的多样性。同时，利用遗传算法中交叉算子的功能，建立外部档案信息交换层，将每一次得到的全局最优解(全局最小适应度)相关信息记录进外部档案信息中，在优化

过程中，外部档案信息中的最优解以一定比例和优化层以及进化与学习层结合来产生新解，这样可以为青蛙寻优以及青蛙进化与学习提供更好的全局最优。

### 3.1. 青蛙进化与学习

蛙跳算法由于更新存在随机性，故对多模态等函数寻优时，往往使得 SFLA 后期搜索过程出现收敛速慢，解的精度较差等现象。我们将通过引入 PSO 算法中粒子的进化方式，用于改变这一现象。

在蛙群一次迭代完成后，判断青蛙的新位置 P 是否优于旧位置 OP，如果没有，那么将此青蛙进行进化与学习，具体步骤如下：

Step1：将青蛙移动位置后得到的新位置 P 与旧位置 OP 进行交叉操作，得到位置 P1。

Step2：计算青蛙位置 P、P1 的适应度，从中选出适应度更优的青蛙位置，记为 P2。

Step3：比较青蛙位置 P2 与 OP，如果 P2 优于 OP，则取 P2 作为青蛙的新位置以促进算法的收敛，并转至 Step7。否则，转至 Step4 进行青蛙旧位置 OP 对全局最小适应度(即最优解)的学习。

Step4：根据采取的交叉原则，将青蛙旧位置 OP 中的相应变量替换为全局最小适应度中的对应变量，得到青蛙位置 P3。

Step5：计算青蛙位置 P2 与 P3 的适应度，从中选出适应度最优的青蛙位置，记为 NP。

Step6：比较青蛙位置 NP 与 OP，如果 NP 优于 OP，则用 NP 作为青蛙的新位置，以促进算法的收敛；当 NP 和 OP 的适应度相等或无法比较时，NP 也被保存作为青蛙的新位置，这样可以提高算法的多样性；如果 NP 劣于 OP，则取 OP 作为青蛙的位置以防止最优解丢失。

Step7：青蛙进化和学习结束。

### 3.2. 外部档案信息交换

本文改进的 SFLA 算法采用遗传算法中交叉算子的功能用于实现外部档案信息交换。将全局最小适应度(即最优解)相关信息记录进外部档案信息中。在优化过程中，外部档案信息中的最优解以一定比例和混合蛙跳优化层以及青蛙进化与学习层结合以产生新解。外部档案信息交换层中的信息将在每次全局最小适应度更新后替换。具体交换方式有：

情形 1：如青蛙 $i$ 的适应度比外部档案中记录最优解的适应度差，则以一定比例和混合蛙跳优化层以及青蛙进化与学习层结合以产生新解；

情形 2：如适应度相等或无法比较，保留原青蛙位置；

情形 3：如适应度优于最优解适应度，则保留此青蛙位置。

### 3.3. 算法伪代码

```
初始化参数
F←m*n;//产生初始青娃
for i1 = 1 to F
给所有青蛙赋初始值
end
//全局迭代寻优
While ii = 1 to G
for i2 = 1 to F
计算所有青蛙的适应度
```

end
for i3 = 1 to F
排序
end
xg←全局最好的青蛙
//局部搜索过程//混合蛙跳优化层，提供新位置
for i4 = 1 to m
for j = 1 to N//每组青蛙迭代次数
记录组内最优 xb 与组内最差的青蛙 xw
使用 rand.*(xb-xw)进行组内最优更新操作
if fun(temp) > fun(xw)
使用 rand.*(xg-xw)进行采用全局最优更新操作
end
if fun(temp) > fun(xw)
使用 pmax*rands(1,d)进行随机更新操作
end
记录最差的青蛙更新后的位置为 P
If 青蛙新位置 p 比旧位置 op 差
Then 蛙群学习与进化
使用交叉算子对 P 和 op 进行交叉操作得到位置 P1
计算 P、P1 的适应度并比较，选择适应度更优的青蛙位置，记为 P2
If p2 比 op 差
Then 将记录的全局最优解 xg 记为 P3
选择 P2、p3 中的最优位置，记为 NP
If NP 优于或者等于 OP
Then 保留 Np 作为青蛙新位置
Elseif 保留 op 作为青蛙位置
Else 保留 p2 为青蛙的新位置
End
Elseif 青蛙新位置 P 优于旧位置
Then 保留 P
End
end　//结束 N
更新全局最优解
end　//结束 m
end　//结束 G

### 3.4. SFLA 算法流程图

MSFLA 算法流程图如下图 1。

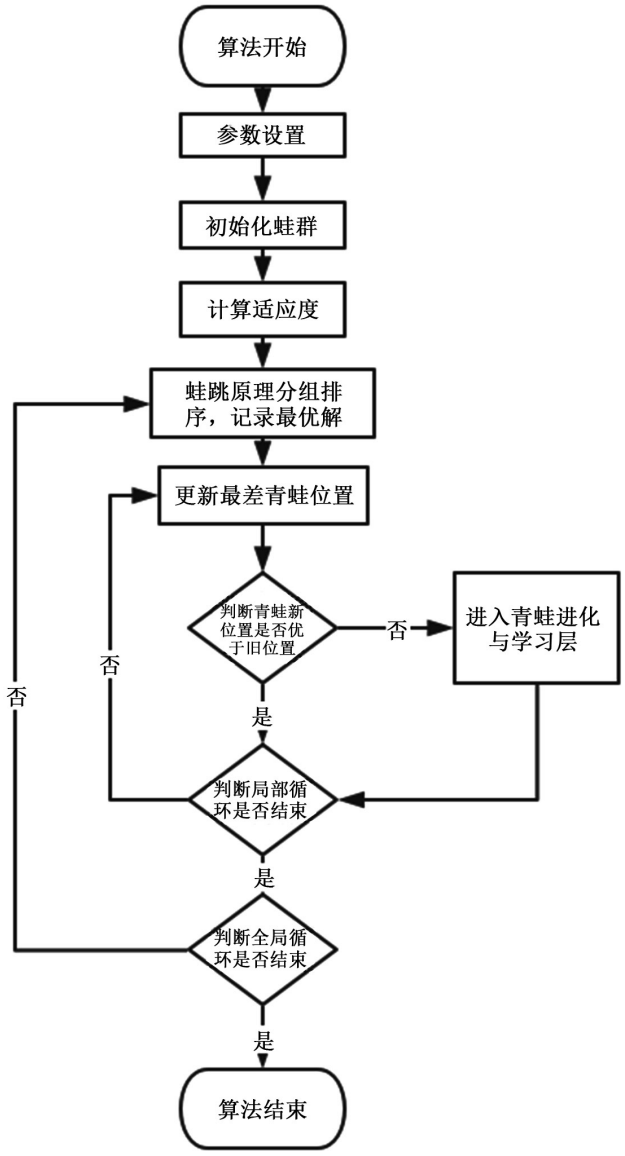


**Figure 1.** MSFLA algorithm flow chart
**图 1.** MSFLA 算法流程图

## 4. 仿真实验

为了验证 MSFLA 的收敛效果，根据文献[10]中的函数选择，选取了五个经典的测试函数进行仿真实验，将改进后的蛙跳算法与标准混合蛙跳算法进行对照实验。为了更好的进行 MSFLA 与 SFLA 算法的对比实验，本文进行实验时所有函数都采取一样的参数。参数设置如表 1 所示。

**Table 1.** Algorithm parameter settings
**表 1.** 算法参数设置

| 参数 | 数值 | 参数 | 数值 |
|---|---|---|---|
| 个群个数 m | 50 | 最大移动步长 smax | 10 |
| 个群中青蛙数 n | 35 | 种群最大迭代次数 MAXGEN | 200 |
| 子群内更新数 N | 25 | 优化函数维数 d | 25 |

表 2 列举了五个测试函数的表达式、范围、维数以及最优值。

**Table 2.** Test function
**表 2.** 测试函数

| 函数名 | 测试函数 | 维数 | 范围 | 最优值 |
|---|---|---|---|---|
| Sphere | $f_1(x)=\sum_{i=1}^{n}x_i^2$ | 25 | [−100, 100] | 0 |
| Rosenbrock | $f_2(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(1-x_i\right)^2\right]$ | 25 | [−1024, 1024] | 0 |
| Rastrigin | $f_3(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos\left(2\pi x_i\right)+10\right]$ | 25 | [−5.12, 5.12] | 0 |
| Griewank | $f_4(x)=\sum_{i=1}^{n}\frac{\left(x_i\right)^2}{4000}-\prod_{i=1}^{n}\cos\left(x_i/\sqrt{i}\right)+1$ | 25 | [−600, 600] | 0 |
| Schafferf7 | $f_5(x)=\sum_{i=1}^{n-1}\left(x_i^2+x_{i+1}^2\right)^{0.25}\left[\sin^2\left(50\left(x_i^2+x_{i+1}^2\right)^{0.1}\right)+1\right]$ | 25 | [−10, 10] | 0 |

在固定总迭代次数时，标准 SFLA 和 MSFLA 寻优精度的仿真结果如表 3 所示，从表中数值的对比结果可看出，在相同进化代数的前提下，MSFLA 对于上述测试函数无论是最优值、最差值还是平均值的结果均优于标准的 SFLA，故 MSFLA 可有效地提高算法的全局收敛性。

**Table 3.** Simulation results of optimization accuracy under fixed number of iterations
**表 3.** 固定迭代次数下寻优精度的仿真结果

| 函数名 | 算法 | 最优值 | 最差值 | 平均值 |
|---|---|---|---|---|
| Sphere | SFLA | 0.3396 | 0.4248 | 0.3846 |
| | MSFLA | 0.1471 | 0.2285 | 0.1823 |
| Rosenbrock | SFLA | 0.1070 | 0.3311 | 0.2191 |
| | MSFLA | 0.0505 | 0.1410 | 0.0958 |
| Rastrigin | SFLA | 44.3042 | 85.7696 | 64.9346 |
| | MSFLA | 14.5946 | 59.1526 | 36.8736 |
| Griewank | SFLA | 107.0871 | 138.7331 | 123.9108 |
| | MSFLA | 60.4489 | 113.0356 | 81.7423 |
| Schafferf7 | SFLA | 0.0892 | 0.1791 | 0.1307 |
| | MSFLA | 0 | 0 | 0 |

为了将 MSFLA 与标准 SFLA 进行对比，本文给出了两种算法下各测试函数的平均进化曲线图(图 2~6)。

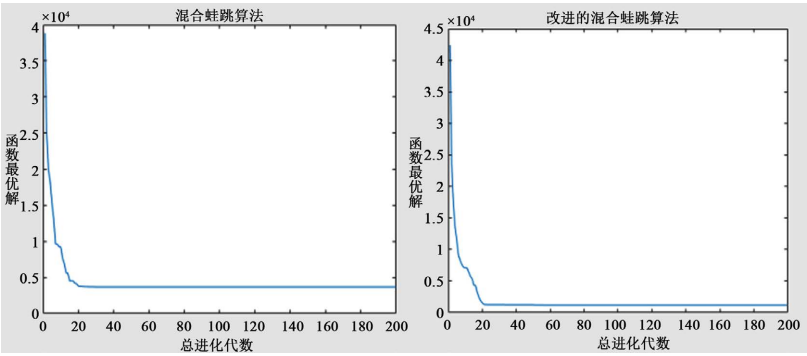


**Figure 2.** Sphere function
**图 2.** Sphere 函数

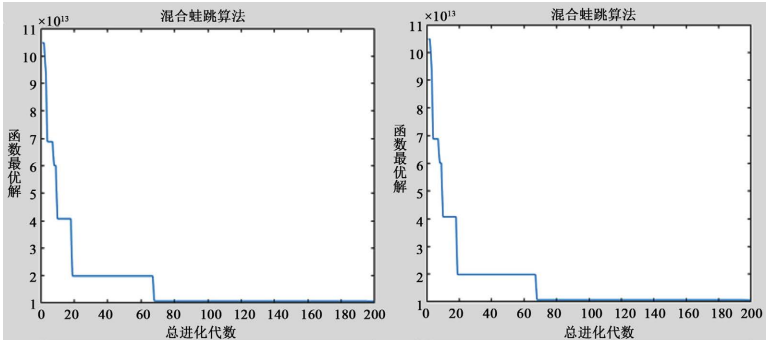


**Figure 3.** Rosenbrock function
**图 3.** Rosenbrock 函数

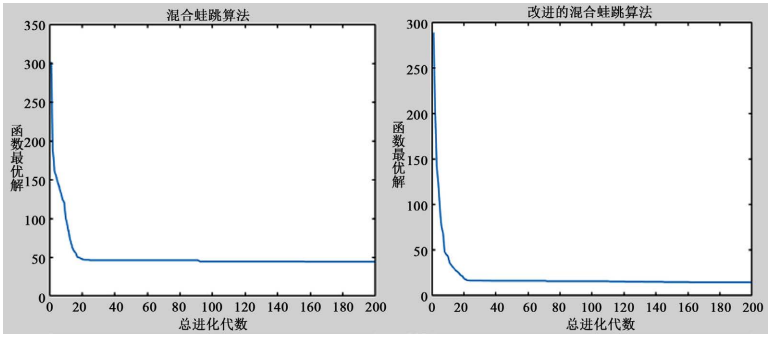


**Figure 4.** Rastrigin function
**图 4.** Rastrigin 函数

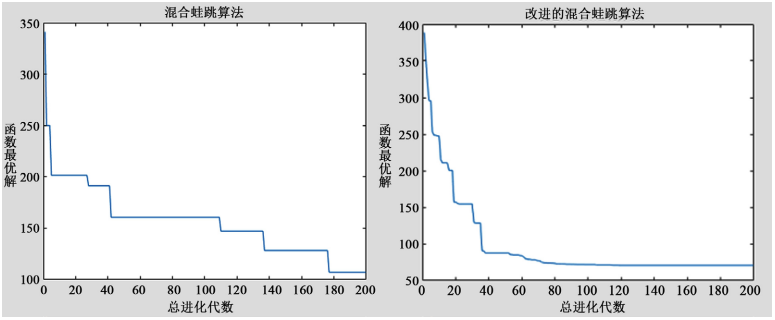


**Figure 5.** Griewank function
**图 5.** Griewank 函数

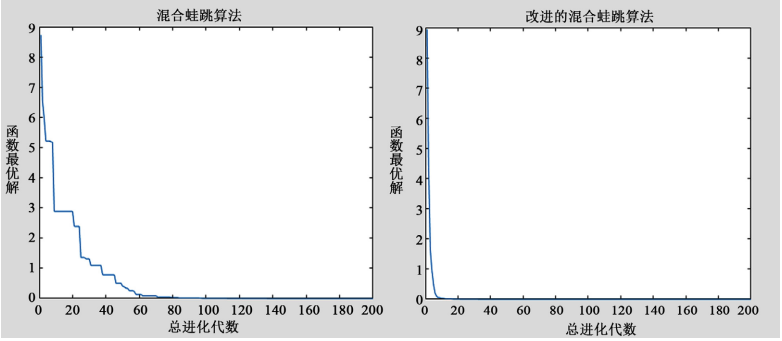


**Figure 6.** Schafferf7 function
**图 6.** Schafferf7 函数

由上述图可见，在迭代次数相同的情形下，MSFLA 相比于 SFLA 具有更高的寻优精度，在寻优精度相同的条件下，MSFLA 比 SFLA 需要的迭代次数更少。

## 5. 结论

针对标准 SFLA 算法更新方式单一，收敛速度慢，易陷入局部最优而导致算法早熟等问题，本文提出的 MSFLA 融合算法在保留混合蛙跳算法的优点的基础上，溶入了粒子群算法的粒子进化和遗传算法的交叉算子。MSFLA 算法通过粒子进化、学习和文件信息交换，提高了混合蛙跳算法的收敛性和多样性，通过新算法与 SFLA 算法的仿真实验可见，新算法在相比 SFLA 优化效果更好，具有更快的收敛速度、更高的寻优精度。

## 基金项目

江西省教育厅科学技术研究项目(编号：GJJ211329)。

## 参考文献

[1] Eusuff, M.M. and Lansey, K.E. (2003) Optimization of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm. *Water Resources Planning and Management*, **129**, 210-225. https://doi.org/10.1061/(ASCE)0733-9496(2003)129:3(210)

[2] 黄世贤, 何晓曦, 刘一明. 混合蛙跳算法研究综述[J]. 电子技术与软件工程, 2021(3): 130-132.

[3] 徐胜超. 基于混合蛙跳算法的容器云资源低能耗部署方法[J]. 重庆邮电大学学报(自然科学版), 2023, 35(5): 952-959.

[4] 张新明, 程金凤, 康强, 等. 改进的混合蛙跳算法及其在多阈值图像分割中的应用[J]. 计算机科学, 45(8): 54-62.

[5] 许健, 许峰. 基于函数变化率的自适应变异蛙跳算法[J]. 软件导刊, 2018, 17(9): 77-80,84.

[6] 孙宇贞, 郭皓文, 黄晓筱. 基于改进混合蛙跳算法的协调控制系统优化[J]. 热能动力工程, 2020, 35(6): 109-115.

[7] 周林, 陶冠宏, 王佩. 一种基于混合蛙跳和粒子群融合的改进优化新算法[J]. 电子信息对抗技术, 2019, 34(6): 27-31.

[8] 孟庆莹, 王联国. 基于邻域正交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] 李俊, 孙辉, 史小露. 多种群粒子群算法与混合蛙跳算法融合的研究[J]. 小型微型计算机系统, 2013, 34(9): 2164-2168.

[10] 孟凯露, 尚俊娜, 岳克强. 混合蛙跳算法的最优参数研究[J]. 计算机应用研究, 2019, 36(11): 3321-3324.