改进的混合蛙跳算法及其在多阈值图像分割中的应用

张新明[1,2] 程金凤[1] 康 强[1] 王 霞[1]
(河南师范大学计算机与信息工程学院 河南 新乡 453007)[1]
(河南省高校计算智能与数据挖掘工程技术研究中心 河南 新乡 453007)[2]

摘 要 针对混合蛙跳算法(Shuffled Frog Leaping Algorithm,SFLA)存在的计算复杂度高、优化效率不理想等问题,提出了一种改进的混合蛙跳算法(Improved Shuffled Frog Leaping Algorithm,ISFLA)。在原始SFLA的基础上进行如下改进:首先,将其中每次只更新组内最差青蛙的方式改为更新组内所有青蛙的方式,这既增大了获得优质解的概率,又省去了调整组内迭代次数的步骤,从而提升了优化效率和可操作性;其次,将基于局部最优更新的方法和基于全局最优更新的方法融合为一种混合扰动更新方法,从而避免了复杂条件的选择步骤,进一步提升了优化效率;最后,去掉随机更新方式,以免优质解被破坏,从而提高了整体的优化性能。将ISFLA用于CEC2005和CEC2015连续基准函数的优化测试和基于Renyi熵的灰度和彩色图像分割的多阈值选择实验中,结果表明,与SFLA和state-of-the-art的LSFLA相比,ISFLA具有更高的优化效率,更适用于多阈值图像分割的阈值选择。

关键词 智能优化算法,混合蛙跳算法,图像分割,多阈值图像分割,Renyi熵
中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.08.010

Improved Shuffled Frog Leaping Algorithm and Its Application in Multi-threshold Image Segmentation

ZHANG Xin-ming[1,2] CHENG Jin-feng[1] KANG Qiang[1] WANG Xia[1]
(College of Computer and Information Engineering,Henan Normal University,Xinxiang,Henan 453007,China)[1]
(Engineering Technology Research Center for Computing Intelligence & Data Mining of Henan Province,Xinxiang,Henan 453007,China)[2]

Abstract Aiming at the disadvantages of shuffled frog leaping algorithm (SFLA),such as high computational complexity and poor optimization efficiency,an improved shuffled frog leaping algorithm (ISFLA) was proposed in this paper. The following improvements have been made on the basis of SFLA. Firstly,the method which only updates the worst frog in SFLA is replaced by the method which updates all frogs in each group. This replacement can increase the probability of obtaining the high quality solutions,omit the steps of setting the number of iterations in the group and then improve the optimization efficiency and operability. Secondly,the method based on local optimum updating and the method based on global optimum updating are combined into a hybrid disturbance updating method,which avoids the tedious condition selection steps and further improves the optimization efficiency. Finally,the random updating method is removed to avoid destroying the superior solutions and further enhance the overall performance optimization. ISFLA was tested on the benchmark functions from CEC2005 and CEC2015,and was applied to the multi-threshold gray and color images segmentation based on Renyi entropy. The experimental results show that,ISFLA obtains higher optimization efficiency and is more suitable for threshold selection of multi-threshold image segmentation compared with SFLA and the state-of-the-art LSFLA.

Keywords Intelligent optimization algorithm,Shuffled frog leaping algorithm,Image segmentation,Multi-threshold image segmentation,Renyi entropy

1 引言

混合蛙跳算法是Eusuff和Lansey[1]于2003年提出的一种仿生群智能优化算法。该算法模拟了青蛙族群寻找食物的过程,通过将青蛙族群分组并对组内适应度值最差的青蛙的位置进行更新,来达到搜索优化问题的最优解的目的,已经在诸多领域得到成功应用[2-5]。SFLA模型虽然简单、容易实现,但也存在更新步骤繁琐、计算复杂度高及优化性能不足等缺陷。

到稿日期:2017-10-24 返修日期:2017-12-28 本文受河南省重点科技攻关项目(132102110209),河南省高等学校重点科研项目(19A520026)资助。

张新明(1963-),男,教授,CCF会员,主要研究方向为模式识别、数字图像处理和智能优化算法等,E-mail:xinmingzhang@126.com(通信作者);程金凤(1990-),女,硕士生,主要研究方向为数字图像处理;康 强(1989-),男,硕士生,主要研究方向为数字图像处理和智能优化算法;王 霞(1993-),女,硕士生,主要研究方向为数字图像处理和智能优化算法。

为了提升 SFLA 的优化性能，国内外学者进行了大量的研究。文献[6]提出了一种基于新搜索策略的混合蛙跳算法，定义了新的青蛙分类标准，并针对每个青蛙动态调整权重，避免了算法搜索的盲目性。文献[7]提出了一种自适应分组混沌云模型蛙跳算法，通过结合云模型在定性与定量之间的相互转换，避免算法陷入局部最优位置。文献[8]将组内更新方式改为更新最优青蛙的惯性矩，使算法的收敛能力得到了提高。虽然这些研究在一定程度上提升了 SFLA 的优化性能，但随着更多复杂优化问题和严格实时性要求的提出，亟需性能更好和效率更高的 SFLA，因此 SFLA 依然有着较大的改进空间。

阈值分割是图像分割方法之一，常用的阈值法有最小交叉熵法[9]、最大熵法[10]和 Renyi 熵法[11]等。最大 Renyi 熵准则是 Sahoo 等提出的一种图像阈值分割准则，可以较好地分离图像的目标区域和背景区域。阈值分割的关键在于寻找合适的阈值或阈值向量；穷举方法是寻找阈值或阈值向量的传统方法，能找到合适的阈值或阈值向量，但计算复杂度高，尤其是在多阈值图像分割中，随着阈值数的增加，计算复杂度呈指数倍增加。因此，许多学者通过引入智能优化算法来寻找合适的阈值或阈值向量，从而大幅度加快了分割速度。但目前，在处理多阈值图像分割时，大多算法仍存在阈值向量选择不理想等问题。鉴于此，获取一种高性能的多阈值优化选择算法十分必要。

针对 SFLA 存在计算复杂度高、优化效率不理想等问题，本文对其进行了改进，提出了一种改进的 SFLA（Improved Shuffled Frog Leaping Algorithm，ISFLA），使得原算法的优化效率和可操作性得到大幅度提升。

2　混合蛙跳算法

SFLA 模仿青蛙在寻找食物过程中族群内部的交流机制，通过群体间的信息共享来寻找有最多食物的地方，从而得到优化问题的最优解。

设在 D 维搜索空间中，首先随机生成 N 只青蛙（候选解向量）组成初始青蛙族群，然后根据优化问题的目标函数评价每只青蛙的适应度值，并根据适应度值将所有青蛙由优至劣排序，最后将有序的青蛙分成 m 个组（$m<N$）。分组标准为：排序第一的青蛙分在第一组，排序第二的青蛙分在第二组，排序第 m 的青蛙分在第 m 组，排序第 $m+1$ 的青蛙再次分在第一组，以此类推[7]。设组内适应度值最优和最差的青蛙分别用 X_b 和 X_w 表示，整个青蛙族群中适应度值最优的青蛙用 X_g 表示。

对于每个组，通过组内迭代，每次只对 X_w 进行更新，具体更新步骤如下。

首先，受组内最优青蛙 X_b 的引导，通过基于局部最优的更新方法对 X_w 进行更新，如式(1)和式(2)所示：

$$D_i = rand * (X_b - X_w) \tag{1}$$

$$X_w' = X_w + D_i \tag{2}$$

其中，D_i 表示蛙跳步长，$rand$ 是均匀分布在 0～1 之间的随机实数，X_w' 是更新后的青蛙。

评价 X_w' 的适应度值，并将其与 X_w 的适应度值进行对比，若 X_w' 的适应度值更优，则用 X_w' 取代 X_w；否则，受青蛙族群中最优青蛙 X_g 的引导，通过基于全局最优的更新方法对 X_w 进行更新，如式(3)和式(4)所示：

$$D_i = rand * (X_g - X_w) \tag{3}$$

$$X_w' = X_w + D_i \tag{4}$$

评价 X_w' 的适应度值，并将其与 X_w 的适应度值进行对比，若 X_w' 的适应度值更优，则用 X_w' 替代 X_w；否则，用解空间中随机生成的一只青蛙取代 X_w，如式(5)所示：

$$X_w' = s_{\min} + rand * (s_{\max} - s_{\min}) \tag{5}$$

其中，$s_{\max}$ 为解空间定义域的上界向量，$s_{\min}$ 为解空间定义域的下界向量。

上述步骤完成了对 X_w 的一次组内更新，然后进行组内排序。对组内更新 X_w 和组内排序进行迭代，直到达到最大组内迭代次数 $L_{\max}$，然后再对下一个组进行同样的组内迭代更新和排序，直至所有的组都完成组内迭代更新时，将所有组的青蛙重新混合组成新的青蛙族群并进行组外排序，即完成了一次算法迭代。在下一次算法迭代中，将新的青蛙族群按原方法重新分组，重复上述步骤，直至满足算法停止条件。此时，青蛙族群中适应度值最优的青蛙 X_g 即为搜索到的最优解。SFLA 的总流程如图 1 所示。

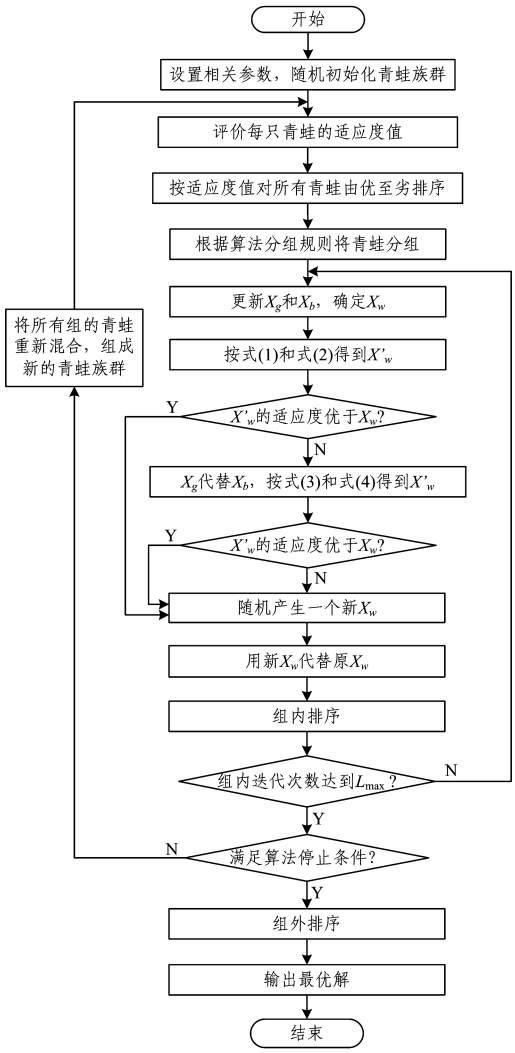


图 1　SFLA 总流程

Fig. 1　General flowchart of SFLA

3 改进的混合蛙跳算法

SFLA 虽然具有原理简单、容易实现等优点，但也存在不足。1)该算法在每次组内迭代中只更新一只青蛙即最差青蛙，若组内迭代次数设置得较少(可降低计算复杂度)，则有一部分青蛙可能得不到更新，从而降低了获取优质解的概率；若组内迭代次数设置得较多，则有可能更新到组内所有青蛙，这虽然能提高获取优质解的概率，但需要更大的计算量。2)青蛙更新需要通过条件选择采用 3 种不同的更新方式，更新步骤繁琐。3)由于采用随机更新方式(见式(5))，因此有可能破坏优质解。以上几种原因导致 SFLA 的优化效率不高。为了提升 SFLA 的优化效率，提出改进的 SFLA，其具体描述如下。

3.1 组内扰动更新

将 SFLA 中每次组内迭代只更新适应度值最差的青蛙改为更新组内所有青蛙，从而使每只青蛙的适应度值都有机会变得更优，这样既增大了获得优质解的概率，又省去了调整组内迭代次数的麻烦，提高了算法的可操作性。

当 SFLA 对最差青蛙进行更新时，效率最高的情况是直接通过式(1)和式(2)的计算得到适应度值更优的青蛙；否则，就需要更多的条件选择步骤对该青蛙进行式(3)和式(4)或式(5)的计算。但由于式(5)得到的是解空间内随机生成的结果，并不能保证更新会向着更优的方向发展，从而可能破坏优质解，导致青蛙族群退化，影响算法的优化效率。因此，去掉式(5)的随机更新方式，以避免优质解被破坏，同时将基于局部最优的更新和基于全局最优的更新整合为一种混合扰动更新方法，从而解决青蛙更新中条件选择和不同的更新方式带来的更新步骤繁琐的问题及多次计算适应度值带来的耗时问题。混合扰动更新方法由差分扰动更新方法和直接扰动更新方法组成，如式(6)和式(7)所示：

$$X_i = X_i + rand * (X_g - X_i + X_{rn1} - X_{rn2}) \tag{6}$$

$$X_i = X_i + (rand_{1\ to\ d} - 0.5) * (X_g - X_i + X_{SI} - X_i) \tag{7}$$

其中，X_i 是被选中的待更新的青蛙，$i=1,2,\cdots,n$，$n = N/m$；X_g 是青蛙族群中最优的青蛙；X_{rn1} 和 X_{rn2} 是从当前待更新的组内随机选择的 2 只青蛙，满足 $i \neq rn1 \neq rn2$；X_{SI} 是通过榜样学习方案(详见 3.2 节)从当前组内选出的榜样青蛙；$rand_{1\ to\ d}$ 指一个随机的 d 维向量，即该向量中的每一维都是均匀分布在 0～1 之间的随机数。

式(6)是差分扰动更新方法，只有组内适应度值最优的青蛙才采用此方法更新其位置。从式(6)可以看出，待更新的青蛙 X_i 受到自身、青蛙族群中适应度最优的青蛙 X_g 及当前组内另外 2 只随机选择的不同青蛙 X_{rn1} 和 X_{rn2} 的共同影响。对 4 只青蛙进行加权差分计算，并将计算结果赋值给 X_i，使其获得更多样化的信息。这种在解空间内的扰动搜索，有利于算法的全局搜索。

式(7)是直接扰动更新方法，是原算法中基于局部最优的更新和基于全局最优的更新的融合，用于更新组内的其他青蛙。从式(7)可以看出，待更新的青蛙 X_i 受到自身、青蛙族群中适应度最优的青蛙 X_g 和一个当前组内选出的榜样青蛙 X_{SI} 的共同影响。通过 X_g 和 X_{SI} 对 X_i 的直接加权计算，使其在解空间中 X_g 及 X_{SI} 附近的局部范围内扰动搜索，增强了局部搜索能力。榜样学习方案的使用，既保证了 X_i 能够得到比自己更优的青蛙的信息，又可以对不同的青蛙更新选出不同的榜样青蛙，从而避免了 X_i 只能单一地接收 X_b(局部最优位置)或者 X_g(全局最优位置)的信息，提高了种群多样性，有利于算法的全局搜索。此外，$(rand_{1\ to\ d} - 0.5)$ 可以得到一个所有维度取值均在 -0.5 至 0.5 之间的随机向量，将该值作为直接扰动更新方法中的权值也增加了青蛙更新的随机性，使青蛙每次更新都能够在解空间中搜索到更多的位置。

3.2 榜样学习

榜样学习是一种新颖的学习方案[12]。设种群中有若干个个体，其中，将比第 i 个个体的适应度更优的所有个体均称为第 i 个个体的榜样，这些榜样组成了第 i 个个体的榜样群组。由此也可以推断出，该种群中最优的个体没有榜样，次优个体的榜样群组中只有 1 个榜样，而最差的个体不可能成为榜样。从一个个体 X_i 的榜样群组中随机选出一个榜样，并对 X_i 的更新产生影响，即榜样学习法。榜样的选择如式(8)所示：

$$SI = \mathrm{ceil}(rand * (i-1)) \tag{8}$$

其中，ceil()为向上取整操作。

当采用混合扰动更新方法更新青蛙时，对于组内适应度值最优的青蛙，没有榜样可供学习，因此通过式(6)的差分扰动更新方法进行更新。对于组内的其他青蛙，则通过式(7)的直接扰动更新方法进行更新。两种更新方法混合使用可以使算法的全局搜索和局部搜索都有所增强。

3.3 ISFLA 总流程

综合上述所有的改进，便形成了改进的 SFLA，即 ISFLA，其伪代码如算法 1 所示，其中，$MaxDT$ 为算法的最大迭代次数，m 为分组数，组内青蛙数 $n = N/m$，N 是青蛙族群中的青蛙总数。从算法 1 可以看出，只要确定了 m，就确定了组内的青蛙数 n，也即确定了组内迭代次数，因此无需再设置组内迭代次数。

算法 1 ISFLA

```
1. 初始化参数，并随机初始化青蛙族群
2. 评价青蛙族群中每只青蛙的适应度值，并按其值从优至劣进行排序
3. for t=1 to MaxDT do
4. 根据 SFLA 的分组规则将所有青蛙分成 m 组
5.   for g=1 to m
6.     for i=1 to n
7.       通过式(6)更新组内适应度值最优的青蛙
8.     通过式(8)选择榜样青蛙，并通过式(7)更新组内其他的青蛙
9.     end for
10.    对每只青蛙进行越界限制
11.      评价每只青蛙的适应度值
12. 采用贪婪选择法替换组内青蛙
13. 将该组青蛙并入到青蛙族群中
14. end for
15. 对青蛙族群中的所有青蛙按适应度值由优至劣排序
16. end for
17. 输出最优解
```

对比图 1 和算法 1 可以看出：ISFLA 弥补了 SFLA 每次组内迭代只更新适应度值最差青蛙的缺陷并解决了青蛙更新步骤繁琐的问题，增大了获得优质解的概率，增强了全局搜索和局部搜索，从而提升了算法的优化性能；同时，由于 ISFLA 无须设置组内迭代次数，也无须对组内青蛙排序，因此降低了算法的计算复杂度，也提高了算法的可操作性。优化性能的提升和时间复杂度的降低使算法能够在更短的时间内获得更优的结果，从而提升了算法的优化效率。

4　用于 Renyi 熵多阈值图像分割的 ISFLA

4.1　基于 Renyi 熵的多阈值选择

假设有一幅灰度级为 L 的图像，其灰度出现的概率分布为 $P=\{p_i \mid i=0,1,\cdots,L-1\}$，图像大小为 $N*M$，则该图像中灰度级 i 的概率可用直方图估计为 $p_i=n_i/(N*M)$，其中 n_i 表示第 i 级灰度在图像中出现的像素个数。

图像单阈值分割 Renyi 熵法请参见文献[11]。将 Renyi 熵阈值分割扩展到图像多阈值分割问题时，考虑有 n 个阈值 $(t_1,t_2,\cdots,t_n)$把图像分割成 $n+1$ 类不同的区域。假设 $t_0=1$，$t_{n+1}=L-1$，则每类的先验 Renyi 熵定义为：

$$H_\alpha^c=\frac{1}{1-\alpha}\ln\sum_{i=t_{c-1}+1}^{t_c}(p_i/p_c)^\alpha \tag{9}$$

其中，$\alpha>0$，常取值为 0.8；c 表示图像区域的个数，$c=1,2,\cdots,n+1$。因此，总熵为：

$$H_\alpha^1+H_\alpha^2+\cdots+H_\alpha^{n+1}=\sum_{n+1}^{c=1}\frac{1}{1-\alpha}\ln\sum_{i=t_{c-1}+1}^{t_c}(p_i/p_c)^\alpha \tag{10}$$

根据最大 Renyi 熵准则，基于 Renyi 熵的多阈值分割图像所需要的最佳阈值为 $H_\alpha^1+H_\alpha^2+\cdots+H_\alpha^{n+1}$ 取得最大值时的阈值组合，即：

$$\{t_1^*,t_2^*,\cdots,t_n^*\}=\arg\max[H_\alpha^1+H_\alpha^2+\cdots+H_\alpha^{n+1}] \tag{11}$$

4.2　用于多阈值图像分割的 ISFLA

因为多阈值分割问题主要是依据某种分割准则来获取最佳阈值向量，是一个离散的优化问题，即灰度向量的每个分量必须取整数，所以将 ISFLA 用于多阈值分割需要做如下修改：

1)所有的决策参数在更新过程中需要离散化，本文通过对参量进行 *round* 取整使其变成整数来构成不同的灰度级别；

2)灰度级别更新的最小单位是一个灰度级，故在优化算法中添加微调算子，即正负 1 算子来加强局部搜索，提高搜索精度。

假设对图像进行 d 阈值分割，则阈值向量 $x=[x_1,x_2,\cdots,x_d]$，其取值均为正整数，并且满足 $0<x_1<x_2<\cdots<x_d<L-1$。以最大 Renyi 熵作为分割标准，在解空间的 L 个灰度级之间采用 ISFLA 算法进行优化选择，使式(10)取得最大值的一组解向量即为最优阈值向量。

ISFLA 用于多阈值图像分割时的步骤如下：

Step 1　输入待分割图片，初始化最大迭代次数 *MaxDT*、种群数量 N 和阈值个数 d 等参数；

Step 2　将式(10)作为评估青蛙适应度值的目标函数，用 ISFLA 搜索图像的最优阈值向量；

Step 3　用求得的最优阈值向量对图像进行分割，输出分割后的图像。

需要说明的是，对于 RGB 彩色图像分割，须采用优化算法分别对其 R，G，B 3 个颜色分量进行最优阈值向量的选取，再根据 ISFLA 获取的阈值向量对彩色图像进行分割。

5　实验结果与分析

为了验证 ISFLA 的有效性，将其用于 CEC2005[13] 和 CEC2015[14] 连续基准函数的寻优和基于 Renyi 熵的灰度和彩色图像的多阈值分割。本文选取 SFLA 和 state-of-the-art 的 LSFLA(Levy flight-based SFLA)作为对比算法，其中 LSFLA 是 Tang 等于 2016 年提出的一种新颖 SFLA 改进算法[5]。文献[5]已经表明，该算法在性能上优于许多其他改进算法，具有很好的优化性能。而 ISFLA 也为 SFLA 的改进算法，因此这两种算法具有很强的可对比性。

所有实验均在 Windows 7 操作系统、主频 3.10 GHz CPU 和 4 GB 内存的 PC 上进行，编程语言采用 Matlab R2014a。

5.1　在 CEC2005 和 CEC2015 基准函数上的实验

本组实验将 3 种算法在 CEC2005 的 10 维 F_1-F_{14}(包括单峰函数 F_1-F_5、多峰函数 F_6-F_{14})及 CEC2015[14] 的 10 维 F_1-F_{15}(包括单峰函数 F_1 和 F_2、旋转平移函数 F_3-F_9、混合函数 $F_{10}-F_{12}$ 及复合函数 $F_{13}-F_{15}$)上进行测试。

为公平起见，将 3 种算法的最大函数评价次数(Maximum Number of Function Evaluation，MNFE)均设置为 50000。ISFLA 参数依据大量的实验结果，设置为：分组数 $m=5$，种群大小 $N=20$，最大迭代次数 $MaxDT=2500$，每个组的青蛙数 $n=N/m=4$；依据文献[5]，LSFLA 的最佳参数设置为：分组数 $m=5$，组内迭代次数 $L_{\max}=9$，参数 $beta=0.6$；设置 SFLA 的 $m=5$，$L_{\max}=5$，组内青蛙数 $n=N/m=4$。LSFLA 和 SFLA 的其他参数设置同其相应的参考文献。

3 种算法在每个基准函数上独立运行 30 次，计算得到平均值(Mean)和标准差值(Std)。由于它们在不同的基准函数上获得的结果各有优势，为直观地展示对比结果，将 3 种算法的结果进行统计排名(Rank)。排名标准为：比较算法获得的 Mean 值，Mean 值越优则排名越好，若几种算法获得了相同的 Mean 值，再比较它们获得的 Std 值，Std 值越优则排名越好，若它们的 Std 值也相等，则排名并列。3 种算法的结果对比如表 1－表 4 所列，其中最优结果用粗体表示。

表 1 列出了 3 种算法在 CEC2005 的 F_1-F_{14} 上获得的结果(表 1 中 Mean 值是算法所得数值减去函数理论最小值的平均结果)，表 2 列出了对表 1 结果进行的排名统计。从表 2 中可以看出，ISFLA 获得了 11 次排名第一，2 次排名第二，1 次排名第三，获得排名第一的次数是 3 种算法中最多的。SFLA 和 LSFLA 分别获得了 2 次排名第一和 1 次排名第一，相差不大；但 LSFLA 获得了 10 次排名第二，而 SFLA 只获得了 2 次排名第二。ISFLA 的平均排名为 1.29，明显优于 LSFLA 的 2.14 和 SFLA 的 2.57。

表 1 各算法在 CEC2005 基准函数上的结果对比

Table 1 Result comparison of three algorithms on CEC2005 benchmark functions

函数	算法	Mean	Std	Rank
F_1	ISFLA	**3.4106e-14**	**5.0842e-14**	1
	SFLA	2.9731e+02	1.8069e+02	3
	LSFLA	1.0736e-03	6.7825e-04	2
F_2	ISFLA	**2.6527e-14**	**3.5744e-14**	1
	SFLA	1.0561e+03	3.6302e+02	3
	LSFLA	2.0429e-03	1.2390e-03	2
F_3	ISFLA	4.6952e+04	3.8142e+04	2
	SFLA	1.6510e+06	1.0092e+06	3
	LSFLA	**2.0197e+01**	**1.6432e+01**	1
F_4	ISFLA	**3.4106e-14**	**3.5326e-14**	1
	SFLA	1.1494e+03	2.9359e+02	3
	LSFLA	7.0518e-01	4.4117e-01	2
F_5	ISFLA	**1.1734e-01**	**2.2859e+00**	1
	SFLA	9.2506e+03	1.1431e+03	3
	LSFLA	7.1562e+00	5.9817e+00	2
F_6	ISFLA	**1.6129e+01**	**3.4999e+01**	1
	SFLA	2.7252e+06	4.2654e+06	3
	LSFLA	3.0904e+02	2.3314e+02	2
F_7	ISFLA	**2.0432e-01**	1.1437e-01	1
	SFLA	2.1132e+03	1.2274e+02	3
	LSFLA	4.0452e-01	**8.0965e-02**	2
F_8	ISFLA	**2.0383e+01**	8.8836e-02	1
	SFLA	2.0403e+01	**6.1185e-02**	3
	LSFLA	2.0384e+01	7.9594e-02	2
F_9	ISFLA	**3.7248e+00**	**2.1277e+00**	1
	SFLA	2.0787e+01	6.7957e+00	2
	LSFLA	2.1415e+01	4.4525e+00	3
F_{10}	ISFLA	**1.5317e+01**	**4.4153e+00**	1
	SFLA	3.5699e+01	1.2062e+01	3
	LSFLA	2.8537e+01	5.9998e+00	2
F_{11}	ISFLA	**5.1965e+00**	1.6237e+00	1
	SFLA	6.8860e+00	1.4753e+00	3
	LSFLA	6.3505e+00	**6.3841e-01**	2
F_{12}	ISFLA	1.9022e+04	6.9830e+03	3
	SFLA	**2.7148e+03**	**3.5220e+03**	1
	LSFLA	1.7868e+04	7.6328e+03	2
F_{13}	ISFLA	1.1741e+00	**2.3612e-01**	2
	SFLA	**1.0889e+00**	3.9971e-01	1
	LSFLA	1.7419e+00	4.3926e-01	3
F_{14}	ISFLA	**3.2906e+01**	3.9432e-01	1
	SFLA	3.3249e+01	**2.2823e-01**	2
	LSFLA	3.3287e+01	2.4263e-01	3

表 2 各算法在 CEC2005 基准函数上的对比排名统计

Table 2 Comparative ranking statistics of three algorithms on CEC2005 benchmark functions

算法	排名统计(次数)			
	排名第一	排名第二	排名第三	平均排名
ISFLA	11	2	1	1.29
SFLA	2	2	10	2.57
LSFLA	1	10	3	2.14

表 3 列出了 3 种算法在 CEC2015 的 F_1-F_{15} 上获得的结果，表 4 列出了根据表 3 的结果进行的排名统计。从表 4 中可以看出，ISFLA 获得了 8 次排名第一、7 次排名第二，没有排名第三的情况，获得排名第一的次数依然是 3 种算法中最多的。LSFLA 也获得了 6 次排名第一，而 SFLA 只获得了 1 次排名第一。从平均排名看，排名最好的是 ISFLA(1.47)，其次是 LSFLA(1.80)，最差的是 SFLA(2.73)。

表 3 各算法在 CEC2015 基准函数上的结果对比

Table 3 Result comparison of three algorithms on CEC2015 benchmark functions

函数	算法	Mean	Std	Rank
F_1	ISFLA	5.5692e+04	5.1442e+04	2
	SFLA	3.7454e+06	2.0601e+06	3
	LSFLA	**1.1285e+01**	**7.0283e+00**	1
F_2	ISFLA	4.3707e+03	5.5878e+03	2
	SFLA	8.1566e+07	1.9966e+08	3
	LSFLA	**1.6119e+03**	**8.9637e+02**	1
F_3	ISFLA	2.0358e+01	8.4673e-02	2
	SFLA	**1.9618e+01**	2.2473e+00	1
	LSFLA	2.0363e+01	**5.7876e-02**	3
F_4	ISFLA	**1.0350e+01**	**4.1581e+00**	1
	SFLA	2.9404e+01	9.8325e+00	3
	LSFLA	2.3700e+01	5.1830e+00	2
F_5	ISFLA	**6.2018e+02**	2.3570e+02	1
	SFLA	6.8526e+02	2.7382e+02	2
	LSFLA	6.9275e+02	**1.5083e+02**	3
F_6	ISFLA	3.5250e+02	2.1166e+02	2
	SFLA	6.2947e+05	3.2523e+06	3
	LSFLA	**1.4548e+02**	**4.9628e+01**	1
F_7	ISFLA	**6.5489e-01**	3.3523e-01	1
	SFLA	3.6603e+00	1.0383e+00	3
	LSFLA	2.0594e+00	**1.7604e-01**	2
F_8	ISFLA	1.1879e+02	8.2334e+01	2
	SFLA	3.1064e+03	4.8952e+03	3
	LSFLA	**5.5410e+01**	**2.6146e+01**	1
F_9	ISFLA	**1.0024e+02**	8.0075e-02	1
	SFLA	1.0123e+02	4.2961e-01	3
	LSFLA	1.0042e+02	**7.3515e-02**	2
F_{10}	ISFLA	3.4732e+02	1.1855e+02	2
	SFLA	4.2665e+03	2.7668e+03	3
	LSFLA	**2.2729e+02**	**6.1636e+00**	1
F_{11}	ISFLA	1.6211e+02	1.5011e+02	1
	SFLA	1.6906e+02	1.4440e+02	2
	LSFLA	2.2291e+02	**1.3010e+02**	3
F_{12}	ISFLA	**1.0213e+02**	**7.2324e-01**	1
	SFLA	1.0939e+02	3.0434e+00	3
	LSFLA	1.0459e+02	8.4567e-01	2
F_{13}	ISFLA	**3.3711e+01**	2.6596e+00	1
	SFLA	4.6884e+01	5.2237e+00	3
	LSFLA	3.8271e+01	**1.7478e+00**	2
F_{14}	ISFLA	3.3440e+03	2.8455e+03	2
	SFLA	5.3112e+03	3.4943e+03	3
	LSFLA	**1.4373e+03**	**1.1483e+03**	1
F_{15}	ISFLA	**1.0000e+02**	**3.8697e-13**	1
	SFLA	1.0300e+02	3.8474e+00	3
	LSFLA	1.0004e+02	1.0228e-02	2

表 4 各算法在 CEC2015 基准函数上的对比排名统计

Table 4 Comparative ranking statistics of three algorithms on CEC2015 benchmark functions

算法	排名统计(次数)			
	排名第一	排名第二	排名第三	平均排名
ISFLA	8	7	0	1.47
SFLA	1	2	12	2.73
LSFLA	6	6	3	1.80

从 3 种算法的 Mean 值和 Std 值的对比中可以看出，在 CEC2005 的结果对比中，ISFLA 获得的整体结果明显优于另外两种算法；在处理 CEC2015 等更为复杂的基准函数时，虽然 ISFLA 与 LSFLA 之间的差距被缩小，相互之间的竞争更为激烈，但 ISFLA 获得的结果依然是整体最优的。以上结果对比表明，ISFLA 具有较好的优化性能，从而证明了本文的

改进是有效且可行的。

本组实验过程还记录了 3 种算法的运行时间，它们在 CEC2015 基准函数上的运行时间对比如表 5 所列。

表 5　各算法在 CEC2015 基准函数上的运行时间对比

Table 5　Runtime comparison of three algorithms on CEC2015 benchmark function

（单位：s）

函数	ISFLA	SFLA	LSFLA
F_1	0.8958	2.0913	1.6421
F_2	0.7899	2.0736	1.6165
F_3	0.9966	1.5290	1.6346
F_4	0.8239	2.0964	1.6310
F_5	0.9916	2.0678	1.6913
F_6	0.9212	2.1690	1.6825
F_7	1.0098	2.2867	1.8112
F_8	0.9411	2.1255	1.6980
F_9	0.9612	2.3339	1.8420
F_{10}	1.2433	2.4394	2.0167
F_{11}	1.8526	3.0579	2.6848
F_{12}	1.0553	2.4566	1.9486
F_{13}	1.2108	2.4390	1.9990
F_{14}	1.2213	2.4416	2.0396
F_{15}	2.1929	3.5570	3.1043
平均时间	1.1405	2.3443	1.9361

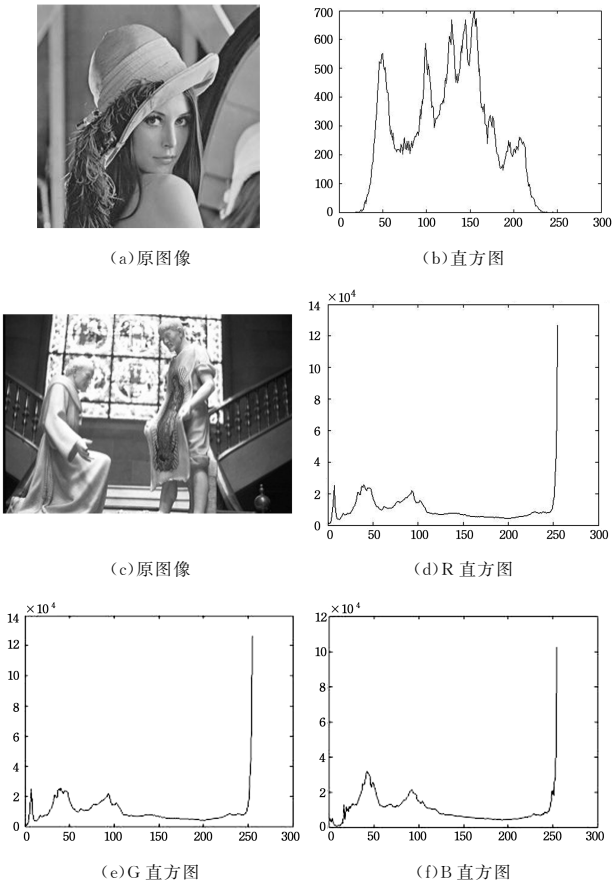
从表 5 可以看出，对于所有的函数，ISFLA 的耗时总是最少的，且优势明显。在平均耗时上，LSFLA（1.9361 s）和 SFLA（2.3443 s）分别达到了 ISFLA（1.1405 s）的 1.6 倍和 2 倍以上。3 种算法在 CEC2005 基准函数上的运行时间对比与在 CEC2015 基准函数上的结果类似，受篇幅限制，不再赘述。以上对比表明，ISFLA 具有较快的运行速度。由于 ISFLA 是 SFLA 的改进算法，因此以 SFLA 为参考，对 ISFLA 的计算复杂度进行分析。影响算法计算复杂度的因素主要有两个：1）算法的函数评价次数，本组实验设置算法的 MNFE 相等，故该因素对算法计算复杂度的影响相同；2）算法自身的流程复杂程度，显然，算法流程中的步骤越繁琐，计算复杂度越高。

对比图 1 和算法 1 可以发现，ISFLA 和 SFLA 的主要区别在于对青蛙族群分组之后的组内更新机制。SFLA 通过 $L_{\max}$ 次组内迭代，对最差的青蛙进行更新，每次迭代的更新次数最少为 1，最多为 3，平均更新次数为 $L_{\max} * 2$。此外，还需要组内排序步骤，按常用的排序方法，其计算复杂度为 $O(n^2)$，n 为组内青蛙个数。而 ISFLA 对组内每个青蛙进行更新，其更新次数固定为 n，根据大量相关文献，n 通常小于 $L_{\max} * 2$，且 ISFLA 没有组内排序的复杂度为 $O(n^2)$。因此，ISFLA 的计算复杂度明显低于 SFLA。另外，SFLA 采用多条件选择，因此只能对目标函数进行串行计算，而 ISFLA 采用并行运算方式计算适应度值。这些都印证了表 5 的运行时间对比结果。

综上所述，ISFLA 展现了较好的优化性能和较快的运行速度，整体结果是 3 种算法中最优的，从而验证了其较高的优化效率。

5.2　多阈值图像分割实验

本组实验将 ISFLA 算法用于多阈值图像分割，以验证其在图像分割应用中的有效性。采用 ISFLA 在大量图像上进行基于 Renyi 熵的图像分割实验，受篇幅所限，本文仅选取分割实验常用的灰度图像 Lena.jpg 及彩色图像 24077.jpg 作为示例来说明算法在 2～5 阈值图像分割的阈值向量搜索效率。两幅实验用图的原图像分别如图 2(a)和图 2(c)所示，灰度图像 Lena.jpg 的直方图如图 2(b)所示，彩色图像 24077.jpg 的 3 个分量的直方图分别如图 2(d)－图 2(f)所示，其中，横坐标表示灰度级，取值范围为 0～255，纵坐标表示图像中具有某一灰度级别的像素的个数。展示直方图的目的主要是便于观察各种算法获取的阈值是否在直方图的谷处附近。



(a)原图像　(b)直方图　(c)原图像　(d)R 直方图　(e)G 直方图　(f)B 直方图

图 2　两幅图像及其对应的直方图

Fig. 2　Two images and their histograms

为公平起见，设置 ISFLA 的种群大小 $N=30$，最大迭代次数 $MaxDT$ 根据阈值数的不同动态调整，在 2 阈值、3 阈值、4 阈值和 5 阈值图像分割实验中设置，分别为 40，70，104，142。另外，保证 3 种算法的 MNFE 均为 $N * MaxDT$。其他参数：设置 ISFLA 的分组数 $m=6$，每个组的青蛙数 $n=5$；设置 LSFLA 的 $m=5$，$L_{\max}=9$，$beta=0.6$；对 SFLA 做如下设置：$m=6$，$L_{\max}=5$，$n=5$。通过这些设置保证 LSFLA 和 SFLA 的性能最佳，它们的其他参数设置同其相应的参考文献。

目前，对于图像的多阈值分割还没有一个公认且已确定的最优阈值向量作为参考标准。为保证数据的可靠性，实验将 3 种算法在每次图像分割中分别独立运行 30 次，并选其中所得最大的（最优的）Renyi 熵值（V_{opt}）和其对应的最优阈值向量（optimal thresholds）作为参考标准，如表 6 所列。

表 6 最优 Renyi 熵值及其对应的最优阈值向量

Table 6 Optimal Renyi entropy values and their corresponding optimal threshold vectors

图像	阈值数	Optimal thresholds	V_{opt}
Lena	2	93,159	12.0583
	3	73,118,165	21.9349
	4	72,117,162,211	38.5510
	5	65,105,143,177,213	66.1370
24077 (R)	2	75,145	12.0513
	3	62,117,175	22.7257
	4	54,104,154,204	41.3322
	5	40,80,121,166,211	72.3850
24077 (G)	2	96,168	12.9164
	3	63,121,184	24.6380
	4	57,108,157,207	45.2946
	5	51,96,137,176,215	78.6498
24077 (B)	2	98,169	12.8674
	3	68,128,189	24.5536
	4	59,109,158,207	44.7089
	5	54,97,135,174,213	77.8534

取 3 种算法得到的平均值 *Mean*、标准差值 *Std*、最大值 *Max*、最小值 *Min* 及成功次数 *NS*(number of success,即将每次独立运行搜索到的 Renyi 熵与表 6 的参考标准值作对比,若得到参考标准值,则记成功 1 次)进行对比。

3 种算法的多阈值向量搜索结果对比如表 7 所列。利用 ISFLA 对 Lena.jpg 进行 3 阈值和 4 阈值分割的结果如图 3 所示。从表 7 中 *Max* 值一栏可以看出,对于 ISFLA 和 SFLA,所有情况下都能得到最大的 Renyi 熵值;LSFLA 在相对低维的 2 阈值和 3 阈值图像分割中能够得到最大的 Renyi 熵值,但是在相对高维的 4 阈值图像分割中的某些情况下不能得到最大的 Renyi 熵值,在相对更高维的 5 阈值图像分割中都没有获得最大的 Renyi 熵值。从表 7 中 *Min* 值一栏可以看出,多数情况下,ISFLA 的最差值(*Min* 值)也达到了最大的 Renyi 熵值,而其他两种算法均没有达到。从 *Mean* 值一栏可以看出,多数情况下,ISFLA 的 *Mean* 值都达到了最大的 Renyi 熵值,而两种对比算法的 *Mean* 值偶尔能够达到最大的 Renyi 熵值。

表 7 3 种算法的搜索 Renyi 熵值的结果对比

Table 7 Renyi entropy comparison of three algorithms

图像	阈值数	算法	*Mean*	*Std*	*Min*	*Max*	*NS*
Lena	2	ISFLA	**12.0583**	**1.8067e−15**	**12.0583**	**12.0583**	**30**
		SFLA	**12.0583**	2.4849e−04	12.0570	**12.0583**	28
		LSFLA	12.0582	3.9242e−04	12.0567	**12.0583**	24
	3	ISFLA	**21.9349**	**1.0840e−14**	**21.9349**	**21.9349**	**30**
		SFLA	21.9118	5.1439e−02	21.7770	**21.9349**	15
		LSFLA	21.9336	1.7448e−03	21.9271	**21.9349**	11
	4	ISFLA	**38.5510**	**7.2269e−15**	**38.5510**	**38.5510**	**30**
		SFLA	38.4748	1.2109e−01	38.2259	**38.5510**	11
		LSFLA	38.5127	4.5883e−02	38.3341	38.5505	0
	5	ISFLA	**66.1370**	**2.8908e−14**	**66.1370**	**66.1370**	**30**
		SFLA	65.9384	2.4107e−01	65.2011	**66.1370**	2
		LSFLA	66.0594	5.2320e−02	65.8764	66.1302	0
24077 (R)	2	ISFLA	**12.0513**	**1.8067e−15**	**12.0513**	**12.0513**	**30**
		SFLA	**12.0513**	1.4138e−04	12.0505	**12.0513**	29
		LSFLA	**12.0513**	7.7520e−05	12.0510	**12.0513**	26
	3	ISFLA	**22.7257**	**3.6134e−15**	**22.7257**	**22.7257**	**30**
		SFLA	22.7219	7.4058e−03	22.6959	**22.7257**	14
		LSFLA	22.7244	1.2753e−03	22.7207	**22.7257**	9
	4	ISFLA	**41.3320**	**6.5073e−04**	**41.3286**	**41.3322**	**29**
		SFLA	41.3216	2.2856e−02	41.2324	**41.3322**	14
		LSFLA	41.3261	6.0769e−03	41.3037	**41.3322**	4
	5	ISFLA	**72.3769**	4.4324e−02	**72.1422**	**72.3850**	**29**
		SFLA	72.2957	1.1407e−01	72.0848	**72.3850**	7
		LSFLA	72.3530	**2.1097e−02**	72.3003	72.3814	0
24077 (G)	2	ISFLA	**12.9164**	**3.6134e−15**	**12.9164**	**12.9164**	**30**
		SFLA	12.9163	4.8690e−04	12.9138	**12.9164**	27
		LSFLA	12.9163	2.5974e−04	12.9152	**12.9164**	22
	3	ISFLA	**24.6380**	**1.4454e−14**	**24.6380**	**24.6380**	**30**
		SFLA	24.6352	9.0823e−03	24.5894	**24.6380**	18
		LSFLA	24.6372	9.1812e−04	24.6336	**24.6380**	11
	4	ISFLA	**45.2946**	**2.1681e−14**	**45.2946**	**45.2946**	**30**
		SFLA	45.2819	3.1006e−02	45.1708	**45.2946**	21
		LSFLA	45.2768	1.0469e−02	45.2498	**45.2946**	1
	5	ISFLA	**78.6387**	**1.8727e−02**	**78.6082**	**78.6498**	**22**
		SFLA	78.5891	5.8496e−02	78.4599	**78.6498**	6
		LSFLA	78.5900	3.8390e−02	78.4744	78.6462	0

（续表）

图像	阈值数	算法	*Mean*	*Std*	*Min*	*Max*	*NS*
24077 (B)	2	ISFLA	**12.8674**	**5.4202e−15**	**12.8674**	**12.8674**	**30**
		SFLA	12.8673	1.9962e−04	12.8664	**12.8674**	28
		LSFLA	12.8672	4.8657e−04	12.8647	**12.8674**	21
	3	ISFLA	**24.5536**	**1.0840e−14**	**24.5536**	**24.5536**	**30**
		SFLA	**24.5536**	1.6650e−04	24.5528	**24.5536**	22
		LSFLA	24.5531	8.7326e−04	24.5502	**24.5536**	7
	4	ISFLA	**44.7089**	**1.4454e−14**	**44.7089**	**44.7089**	**30**
		SFLA	44.6938	4.5348e−02	44.4671	**44.7089**	12
		LSFLA	44.6991	5.3075e−03	44.6865	**44.7089**	1
	5	ISFLA	**77.8534**	**0**	**77.8534**	**77.8534**	**30**
		SFLA	77.7425	1.3113e−01	77.4089	**77.8534**	2
		LSFLA	77.8121	1.8875e−02	77.7598	77.8421	0

(a)3 阈值分割结果

(b)4 阈值分割结果

图 3　Lena 图像的多阈值分割结果

Fig. 3　Multi-thresholded results for Lena image

以上对比结果表明，ISFLA 在基于 Renyi 熵的多阈值图像分割过程中，有能力搜索到相对最优的阈值向量，得到最大的 Renyi 熵值，展现了较好的优化能力，优于两种对比算法。另外，从 *Std* 值一栏可以看出，在所有情况下，ISFLA 获得的结果都是最好的，表明了在基于 Renyi 熵的多阈值图像分割中 ISFLA 的稳定性是 3 种算法中最强的。从而不难推导出在 *NS* 一栏的对比中，ISFLA 的成功率总是最高的，甚至在一些情况下达到了 100%。对于另外两种算法，相对而言，SFLA 获得的结果优于 LSFLA，其成功率也明显高于 LSFLA，从而表明 LSFLA 虽然在连续的基准函数优化问题上展现出了较强的竞争力，但在处理基于 Renyi 熵的多阈值图像分割时性能不佳，不适合处理离散的多阈值选择优化问题。

本组实验过程依然记录了 3 种算法的运行时间，如表 8 所列，其中，时间单位为秒(s)。

表 8　3 种算法对于多阈值图像分割的运行时间对比

Table 8　Runtime comparison of three algorithms for multi threshold segmentation

图像	阈值数	ISFLA	SFLA	LSFLA
Lena	2	**0.0428**	0.0618	0.0636
	3	**0.0687**	0.1081	0.1133
	4	**0.1097**	0.1607	0.1693
	5	**0.1307**	0.2202	0.2414
24077 (R)	2	**0.0446**	0.0619	0.0627
	3	**0.0651**	0.1079	0.1122
	4	**0.1003**	0.1603	0.1674
	5	**0.1521**	0.2208	0.2357
24077 (G)	2	**0.0443**	0.0621	0.0632
	3	**0.0719**	0.1084	0.1122
	4	**0.0746**	0.1599	0.1684
	5	**0.1509**	0.2208	0.2359
24077 (B)	2	**0.0442**	0.0618	0.0631
	3	**0.0782**	0.1085	0.1124
	4	**0.1055**	0.1605	0.1676
	5	**0.1498**	0.2224	0.2370
平均时间		**0.0896**	0.1379	0.1453

从表 8 中可以看出，ISFLA 的运行时间总是少于另外两种算法，其平均运行时间(0.0896 s)只有 SFLA(0.1379 s)和 LSFLA(0.1453 s)的 64.97%和 61.67%，从而表明了 ISFLA 具有较快的运行速度。

综上所述，ISFLA 在基于 Renyi 熵的多阈值图像分割的阈值向量搜索过程中展现了较好的优化性能和较快的运行速度，能够搜索到相对最优的阈值向量，得到了最大的 Renyi 熵值，获得了最高的成功率，展现了较好的优化效率，从而验证了其用于多阈值图像分割的有效性。

结束语　为了解决 SFLA 存在的问题，文中提出了一种改进的 SFLA：去掉 SFLA 中的随机更新方式，将每次组内只更新最差青蛙的方式改为更新组内所有青蛙的方式，将基于局部最优更新的方法和基于全局最优更新的方法融合为一种混合扰动更新方法。这些改进提高了算法的优化性能和运行速度。基准函数和图像分割实验结果都表明，ISFLA 很好地解决了 SFLA 的优化效率不理想、计算复杂度高等问题，更适合于多阈值分割中的阈值优化选择。

下一步的研究重点是进一步改进 ISFLA，并尝试采用该算法处理更多工程领域的优化问题。

参 考 文 献

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

[2] JI C M, LI J W, ZHANG X M, et al. Application of immune-shuffled frog-leaping algorithm to optimized operation of cascade hydropower stations for short-term power generation[J]. Journal of Hydroelectric Engineering, 2015, 34(1): 29-36. (in Chinese)
纪昌明，李继伟，张新明，等. 梯级水电站短期发电优化调度的免疫蛙跳算法应用研究[J]. 水力发电学报，2015，34(1)：29-36.

[3] ZHENG S L, YANG X N. Group initialization technique of hybrid frog leap algorithm for cognitive radio cooperative spectrum sensing [J]. Acta Physica Sinica, 2013, 62(7): 492-497. (in Chinese)
郑仕链，杨小牛. 用于认知无线电协作频谱感知的混合蛙跳算法群体初始化技术[J]. 物理学报，2013，62(7)：492-497.

[4] LIU Z Z, WANG F B. Improvement of discrete shuffled frog-Leaping algorithm and application in compressed sensing reconstruction [J]. Journal of Jilin University(Engineering and Technology Edition), 2016, 46(4): 1261-1268. (in Chinese)

刘洲洲,王福豹.改进的离散混合蛙跳算法压缩感知信号重构及应用[J].吉林大学学报(工学版),2016,46(4):1261-1268.

[5] TANG D Y,YANG J,DONG S B,et al. A lévy flight-Based shuffled frog-leaping algorithm and its applications for continuous optimization problems[J]. Applied Soft Computing,2016,49:641-662.

[6] ZHAO F,ZHANG G Z. Shuffled frog leaping algorithm based on new search strategy [J]. Computer Applications and Software,2015,32(8):224-228. (in Chinese)
赵芳,张桂珠.基于新搜索策略的混合蛙跳算法[J].计算机应用与软件,2015,32(8):224-228.

[7] ZHANG Q,LI P C. Adaptive grouping chaotic cloud model shuffled frog leaping algorithm for continuous space optimization problems [J]. Control and Decision,2015,30(5):923-928. (in Chinese)
张强,李盼池.自适应分组混沌云模型蛙跳算法求解连续空间优化问题[J].控制与决策,2015,30(5):923-928.

[8] KAUR P,MEHTA S. Resource provisioning and work flow scheduling in clouds using augmented shuffled frog leaping algorithm[J]. Journal of Parallel & Distributed Computing,2016,101:41-50.

[9] HORNG M H,LIOU R J. Multilevel minimum cross entropy threshold selection based on the firefly algorithm[J]. Expert Systems with Application,2011,38(12):14805-14811.

[10] ZHANG X M,TU Q,KANG Q,et al. Grey wolf optimization algorithm with double-hunting modes and its application to multi-threshold image segmentation [J]. Journal of Shanxi University (Natural Science Edition),2016,39(3):378-385. (in Chinese)
张新明,涂强,康强,等.双模狩猎的灰狼优化算法在多阈值图像分割中应用[J].山西大学学报(自然科学版),2016,39(3):378-385.

[11] SAHOO P,WILKINS C,YEAGER J. Threshold selection using Renyi's entropy[J]. Pattern Recognition,1997,30(1):71-84.

[12] ZHANG X M,YIN X X,TU Q. High-dimensional multilevel thresholding based on BBO with dynamic migration and salt & pepper mutation [J]. Optics and Precision Engineering,2015,23(10):2943-2951. (in Chinese)
张新明,尹欣欣,涂强.动态迁移和椒盐变异融合生物地理学优化算法的高维多阈值分割 [J].光学精密工程,2015,23(10):2943-2951.

[13] SUGANTHAN P N,HANSEN N,LIANG J J,et al. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization:Technical Report,KanGAL Report # 2005005[R]. Singapore:Kanpur Genetic Algorithms Laboratory,Nanyang Technological University,2005.

[14] LIANG J J,QU B Y,SUGANTHAN P N,et al. Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization:Technical Report 201411A[R]. Computational Intelligence Laboratory,Zhengzhou University,Zhengzhou China And Technical Report,Nanyang Technological University,Singapore,2014.

(上接第 53 页)

除了不含字幕的帧,字幕检测效率约提高 11%。

结束语 本文提出的两阶段的视频字幕检测和提取算法能准确判断出字幕位置且贴合字幕边缘进行截取,有较高的查全率、查准率以及较广泛的实用性,适用于不同语言的字幕。所提算法能解决复杂背景的问题,对字幕的位置、大小也没有要求,但是对字幕边缘有一定的要求,即需要较为规整的字幕边缘;否则会将不规则的字幕如艺术字的边缘截取掉或者检测不出该字幕,从而导致字幕残缺过多信息而无法提取的问题,具有一定的局限性。如何进一步研究不规则边缘字幕的特征,更加完整地截取边缘不规则字幕,将是我们下一步的研究重点。

参 考 文 献

[1] WANG G H,WANG Z,YANG Y M,et al. Detection and positioning of video captions based on ICA algorithm [J]. Journal of Xi'an Shiyou University (Natural Science Edition),2011,26(3):100-103. (in Chinese)
王国红,王喆,杨永民,等.基于 ICA 算法的视频字幕检测与定位[J].西安石油大学学报(自然科学版),2011,26(3):100-103.

[2] WANG R R,JIN W J,WU L D. A new algorithm for detecting video caption by using multi-frame combination [J]. Journal of Computer Research and Development,2005,42(7):1191-1197. (in Chinese)
王蓉蓉,金万军,吴立德.一种新的利用多帧结合检测视频标题文字的算法[J].计算机研究与发展,2005,42(7):1191-1197.

[3] ZHAO X,LIN K H,HU Y X,et al. Caption form corners:A novel approach to detect caption and caption in videos [J]. IEEE Transactions on Circuits and Systems for Video Technology,2005,15(2):243-255.

[4] SATO T,KANADE T,HUGHES E K,et al. Video OCR:indexing digital news libraries by recognition of superimposed captions[J]. Multimedia Systems,1999,7(5):385-395.

[5] ANGADI S A,KODABAGI M M. Text region extraction from low resolution natural scene images using texture features[C]// 2010 IEEE 2nd International Advance Computing Conference (IACC). 2010.

[6] OUYANG P R,ZHANG W J,GUPTA M M. An adaptive switching learning control method for trajectory tracking of robot manipulators[J]. Mechatronics,2006,16(1):51-61.

[7] SU Y X,PARRA-VEGA V . Global asymptotic saturated output feedback control of robot manipulators[C]// Proceedings of the 7th World Congress on Intelligent Control and Automation. 2008.

[8] ZHAO X,LIN K H,FU Y,et al. Text from corners:A novel approach to detect text and caption in videos[J]. IEEE Transactions on Image Processing,2011,20(3):790-799.

[9] LYU M R,SONG J Q,CAI M. A comprehensive method for multilingual video caption detection,localization and extraction [J]. IEEE Transactions on Circuits and Systems for Video Technology,2005,15(2):243-255.

[10] LANG Y,ZHENG D. An Improved Sobel Edge Detection Operator[C]// IEEE International Conference on Computer Science and Information Technology. IEEE,2010:67-71.