基于维度加权的改进萤火虫算法

臧 睿 李 晶
(东北林业大学理学院 哈尔滨 150040)

摘 要 萤火虫算法是一种基于生物群智能的仿生优化算法,具有概念简明、需要设置的参数少、容易实现等特点。但标准萤火虫算法容易陷入局部最优,尤其是针对高维优化函数时更甚。文献[1]提出了一种基于对偶和维度的改进算法,在种群初始化和算法迭代等方面给出了改进。基于维度加权的方法对文献[1]中提出的算法给出新的改进。算法综合考虑了当前最优萤火虫信息和部分萤火虫信息。实验结果的比较表明,改进后的算法体现了较为突出的优越性。

关键词 萤火虫算法,加权,全局最优,维度
中图法分类号 TP18 **文献标识码** A

Improved Firefly Algorithm Based on Weighted Dimension

ZANG Rui LI Jing
(College of Science, Northeast Forestry University, Harbin 150040, China)

Abstract Firefly algorithm is a bionic optimization algorithm based on biological swarm Intelligence which has the advantages of simple concept, few parameters to adjust and easy to realize. However, it can easily get trapped in the local optima especially for high-dimensional optimization function. In literature [1], an improved algorithm based on opposition and dimension was proposed, which is improved in population initialization and algorithm iteration. In this paper, we proposed a new algorithm based on the dimension-weighted method. The algorithm takes into account the current optimal firefly information and part of firefly information. Through the comparison of the experimental results, the improved algorithm embodies the superiority.

Keywords Firefly algorithm, Weight, Global optimum, Dimension

1 引言

标准萤火虫算法(Firefly Algorithm, FA)由 Yang[2] 于 2008 年提出,它来源于对萤火虫群体行为的简化和模拟,是一种高级启发式算法。该算法通过萤火虫个体之间的相互吸引达到寻优的目的,是一种基于群体搜索的随机优化算法[3]。由于操作简单、需要调整的参数少且易于实现,标准萤火虫算法已经成功应用于诸多领域来求解优化问题[4-8]。

与其他仿生优化算法类似,标准萤火虫算法也存在若干缺陷:容易陷入局部最优、收敛速度慢等。国内外学者在萤火虫位置初始化、算法随机部分、萤火虫的位置更新公式[9]、算法参数设置[10]等方面进行了改进。通过改进,萤火虫算法在无约束优化、多约束优化、线性规划和非线性规划等诸多方面更加适用。文献[1]提出基于对偶和维度的萤火虫算法,对标准萤火虫算法在种群初始化和算法迭代等方面给出改进。本文针对迭代过程提出基于维度加权的改进方法,该方法既能保留原算法的进化优势,又能引入新萤火虫信息,从而提高算法的搜索性能。通过对文献[1]中涉及的 7 个高维优化函数的性能测试表明,改进算法取得了较为满意的效果。

2 标准萤火虫算法

2.1 生物学原理

萤火虫算法是近几十年发展起来的仿生模拟进化算法,它通过模拟自然界萤火虫的运动达到寻优效果。萤火虫算法有 3 点理想化准则[3]:

(1)在算法中,所有的萤火虫不分雌雄,且萤火虫总会向着亮度较大的萤火虫移动,亮度最大的萤火虫则在搜索空间中自由移动;

(2)萤火虫的吸引力与它们的亮度成正比关系,萤火虫的相对亮度与它们的距离成反比关系,即随着萤火虫之间距离的增加,它们的相对亮度逐渐减小,吸引力也逐渐减小;

(3)在优化问题中,萤火虫的绝对亮度表示待优化问题的目标函数值,每个萤火虫的位置表示目标函数的一组解,因此亮度最大的萤火虫的位置即为目标函数的最优解。

2.2 相关定义与记号

考虑优化问题 $\min\limits_{x\in\Omega} f(x)$,其中 $\Omega=\{(r_1,r_2,\cdots,r_n)\in R^d, a_i\leqslant r_i\leqslant b_i, i=1,2,\cdots,d\}$ 为可行域。设初始种群数为 N,记 x_i 为第 $i(i=1,2,\cdots,N)$ 个萤火虫在可行域中的位置。

定义 1 萤火虫 i 的绝对亮度 I_i 与 x_i 处的目标函数值相等,即

$$I_i=f(x_i) \tag{1}$$

定义 2 萤火虫 i 对萤火虫 j 的相对亮度为:

$$I_{ij}(r_{ij})=I_i e^{-\gamma r_{ij}^2} \tag{2}$$

其中,γ 为光吸收系数,r_{ij} 是萤火虫 i 到萤火虫 j 的笛卡儿距离。

$$r_{ij}=\|\vec{x_i}-\vec{x_j}\|=\sqrt{\sum_{k=1}^{d}(x_{i,k}-x_{j,k})^2} \tag{3}$$

定义 3 萤火虫 i 对萤火虫 j 的吸引力为:

本文受中央高校基本科研业务费专项资金(DL09BB40)资助。

臧 睿(1977-),男,博士,副教授,主要研究方向为优化理论与数值计算;李 晶(1992-),女,硕士生,主要研究方向为优化理论与数值计算。

$$\beta_{ij}(r_{ij})=\beta_0 e^{-\gamma r_{ij}^2} \tag{4}$$

其中，β_0 是 $r=0$ 时萤火虫的吸引力。

定义 4 萤火虫根据吸引力移动更新自己的位置，萤火虫 j 的位置更新公式为：

$$\vec{x_j}(t+1)=\vec{x_j}(t)+\beta_{ij}(r_{ij})(\vec{x_i}(t)-\vec{x_j}(t))+\alpha(rand-0.5) \tag{5}$$

其中，t 为算法迭代次数；α 为常数，一般可取 $\alpha\in[0,1]$；$rand$ 是[0,1]上服从均匀分布或其他分布的随机向量。

3 基于对偶和维度的萤火虫算法

标准萤火虫在求解高维优化问题时容易陷入局部最优，因为在求解最优解时，标准萤火虫算法不能充分考虑每个萤火虫每一维的最优值，有些维的值会趋于较好的解，有些维则会变差，全局最优解不理想。基于对偶和维度的萤火虫算法有助于标准萤火虫算法跳出局部最优。其改进如下：

(1)用对偶方法生成算法初始种群。首先随机初始 N 个萤火虫位置 x，然后计算每个萤火虫的对偶位置 $\tilde{x}$，$\tilde{x}_{ij}=a_j+b_j-x_{ij}$，其中 $i=1,2,\cdots,N$，$j=1,2,\cdots,d$。最后在 $x\cup\tilde{x}$ 中取 N 个较好的萤火虫作为初始种群 N。利用对偶方法生成算法初始种群可以加快收敛速度，从而缩短运行时间。

(2)用维度基方法确定当前迭代的最优萤火虫。首先用 *Gbest* 表示标准萤火虫算法当前迭代的最优萤火虫，再用 *Gbest* 替换其他萤火虫相应维度的值后继续迭代更新。每次迭代只替换某个萤火虫某一维的值，直到所有维都达到最优则迭代结束。如图 1 所示，Y 是一个向量，表示初始 *Gbest* 的位置，用第 $i(i=1,2,\cdots,N)$ 个萤火虫的第 $j(j=1,2,\cdots,d)$ 维的值替换 Y 的第 j 维的值，并比较两个目标函数值。若 $f(Y)$ 优于 *Gbest*，则最优萤火虫位置更新，否则不变。此迭代方法可以独立考虑最优萤火虫每一维的最优值，可有效避免算法陷入局部最优。最后，在萤火虫位置更新公式中，所有的萤火虫都向着当前最优萤火虫 *Gbest* 移动，即

$$\vec{x_j}(t+1)=\vec{x_j}(t)+\beta_{ij}(r_{ij})(Gbestpos-\vec{x_j}(t))+\alpha(rand-0.5)$$

相较于标准萤火虫算法中萤火虫向着较亮的萤火虫移动，基于对偶和维度的萤火虫算法效率更高。

维度基方法伪代码如图 1 所示。

```
/*update Gbest by Dimensional-Based approach*/
for each firefly i=1 to N
    for each dimension j=1 to d
            Y:=Gbestpos
            Y:=x(i,j)
    if f(Y)<Gbest
            Gbest:=f(Y)
            Gbestpos:=Y
    end
    end
end
```

图 1 维度基方法伪代码

4 改进萤火虫算法

鉴于标准萤火虫算法在求解高维优化函数时容易陷入局部最优，文献[1]采用维度基方法，用其他萤火虫每一维的值逐一替换标准萤火虫算法当前迭代产生的最优萤火虫位置 *Gbestpos* 的对应维的值。考虑到 *Gbestpos* 是经过标准萤火虫算法得到的当前最优位置，一定程度上优于其他萤火虫位置，将其直接替换后可能会使亮度变得很不理想。基于以上判断，我们希望在替换过程中既保留标准萤火虫算法当前迭代产生的最优萤火虫的信息，也考虑其他萤火虫的维度信息。因此，本文在替换过程中引入了基于维度加权的改进方案。改进的迭代过程如图 2 所示。

```
Y=Gbestpos
for each firefly i=1 to N
    for each dimension j=1 to d
            YY=Y
            Y(1,j)=a*x(i,j)+(1-a)*YY(1,j)
    if f(Y)<Gbest
            Gbest=f(Y)
            Gbestpos=Y
    else
            Gbestpos=YY
    end
    end
end
```

图 2 改进的迭代过程的伪代码

经过实验验证，对于多数优化函数，当 $a=0.5$ 时，可得到较好的最优解。

5 实验仿真及分析

本文针对文献[1]中涉及的 7 个高维优化函数做数值模拟。参数设置与文献[1]保持一致：$N=50$，$MaxGeneration=200$，$\alpha=0.2$，$\gamma=0.001$；对于 Schwefel's 和 Griewank's 函数，取 $N=200$，$MaxGeneration=500$。每个函数均独立运行 50 次且实验结果表中显示的是函数值的中位数。

(1)Ackley 函数

$$f_1(x)=-20e^{-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}}-e^{\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)}+20+e,\quad x_i\in[-32.768,32.768]$$

(2)De Jong 函数

$$f_2(x)=\sum_{i=1}^{d}x_i^2,\ x_i\in[-5.12,5.12]$$

(3)Rosenbrock 函数

$$f_3(x)=\sum_{i=1}^{d-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2],\quad x_i\in[-2.048,2.048]$$

(4)Griewank 函数

$$f_4(x)=\sum_{i=1}^{d}\frac{x_i^2}{4000}-\prod_{i=1}^{d}\cos(\frac{x_i}{\sqrt{i}})+1,x_i\in[-600,600]$$

(5)Rastrigin 函数

$$f_5(x)=10d+\sum_{i=1}^{d}[x_i^2-10\cos(2\pi x_i)],\quad x_i\in[-5.12,5.12]$$

(6)Michalewicz 函数

$$f_6(x)=-\sum_{i=1}^{d}\sin(x_i)\sin^{20}(\frac{ix_i^2}{\pi}),x_i\in[0,\pi]$$

(7)Schwefel 函数

$$f_7(x)=418.9829d-\sum_{i=1}^{d}x_i\sin(\sqrt{|x_i|}),\quad x_i\in[-500,500]$$

表 1 列出了优化函数 f_1-f_7 通过原算法和改进算法($a=0.5$)得到的优化函数分别在 10 维、20 维、30 维的最优值。

表 1　$a=0.5$ 时两种算法实验结果的对比

函数	算法	$d=10$	$d=20$	$d=30$
f_1	原算法	$6.1698*10^{-5}$	$7.4302*10^{-5}$	$1.2596*10^{-4}$
	改进算法	$3.8524*10^{-5}$	$3.9227*10^{-5}$	$3.9542*10^{-5}$
f_2	原算法	$8.9059*10^{-11}$	$1.7910*10^{-10}$	$3.3602*10^{-10}$
	改进算法	$4.2239*10^{-12}$	$4.7687*10^{-11}$	$8.1150*10^{-11}$
f_3	原算法	0.2318	0.7239	0.8273
	改进算法	0.8843	0.9775	8.4607
f_4	原算法	0.0172	$1.0259*10^{-8}$	$2.7066*10^{-9}$
	改进算法	0.0271	$1.3551*10^{-9}$	1.2529810^{-9}
f_5	原算法	$1.1096*10^{-8}$	$5.1128*10^{-8}$	$8.7898*10^{-8}$
	改进算法	$1.5285*10^{-9}$	$1.7319*10^{-9}$	$2.0827*10^{-8}$
f_6	原算法	−9.0019	−18.8389	−28.9458
	改进算法	−9.6176	−19.4823	−29.4008
f_7	原算法	$2.0557*10^{-4}$	$2.5456*10^{-4}$	$3.8185*10^{-4}$
	改进算法	2170.2	2783.3	4540.2

从表 1 可以看出，对于函数 f_1，f_2，f_4，f_5，f_6 改进后的算法表现出较好的寻优性能，尤其是对于高维函数。对于函数 f_3，f_7 改进后的算法寻优效果不理想，f_7 的搜索区间很大，改进后的算法出现不收敛的情况。

图 2－图 7 示出了优化函数在 30 维时的收敛曲线（图中虚线表示原算法，实线表示改进算法）。从图 2－图 7 可以看出，相比于原算法，改进后的算法的收敛速度更快，收敛精度更高；且改进后的算法克服了标准萤火虫算法对于高维函数容易陷入局部最优的缺点，整体表现出较理想的效果。

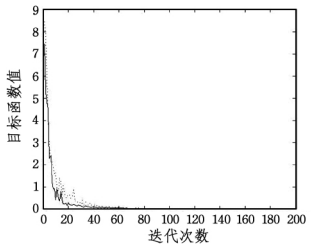

图 2　Ackley 函数的收敛曲线

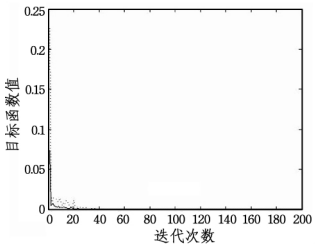

图 3　De Jong 函数的收敛曲线

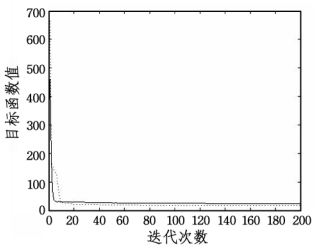

图 4　Rosenbrock 函数的收敛曲线

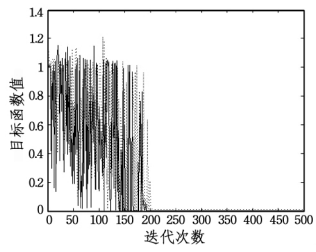

图 5　Griewank 函数的收敛曲线

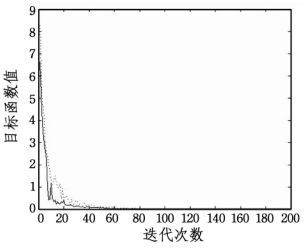

图 6　Rastrigin 函数的收敛曲线

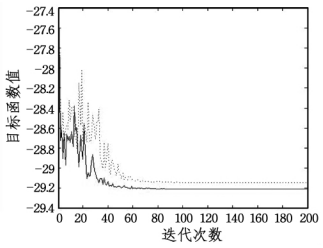

图 7　Michalewicz 函数的收敛曲线

结束语　通过比较高维优化函数的仿真结果，可以看出，改进后的算法对于函数 f_1，f_2，f_4，f_5，f_6 都有较好的效果，即可使高维优化函数更好地收敛到全局最优且收敛速度更快，说明本文对算法的改进有一定的可行性和有效性。

本文对文献[1]提出的基于对偶和维度的萤火虫算法做出了进一步改进，即在算法迭代的寻优过程中的替换部分引入了加权，取部分萤火虫信息和部分全局最优萤火虫信息，保证替换过程不丢失标准萤火虫算法产生的当前最优萤火虫的信息。最后，用测试函数证明了本文改进的有效性及可行性。

参 考 文 献

[1] VERMA O P，AGGARWAL D，PATODI T. Opposition and dimensional based modified firefly algorithm[J]. Expert Systems with Applications，2016(44)：168-176.

[2] YANG X S. Nature-Inspired Metaheuristic Algorithms [M]. Frome：Luniver Press，2008：83-96.

[3] 赵玉新，YANG X S，刘利强. 新兴元启发式优化方法[M]. 北京：科学出版社，2013：148-170.

[4] 李瑞青. 改进的萤火虫算法及应用[D]. 长春：吉林大学，2015.

[5] 王沈娟，高晓智. 萤火虫算法研究综述[J]. 微型机与应用，2015，34(8)：8-11.

[6] SENTHILNATH J，OMKAR S N，MANI V. Clustering Using-firefly Algorithm：Performance Study[J]. Swarm & volutionary EComputation，2011，1(3)：164-171.

[7] ZAMAN MA，MATIN A. Nonuniformly Spaced Linear Antenna Array Design Using Firefly Algorithm[J]. International Journal of Microwave Science and Technology，2012，8(36)：40-48.

[8] 王吉权，王福林. 萤火虫算法的改进分析及应用[J]. 计算机应用，2014，34(9)：2552-2556.

[9] YU S H，SU S B，LU Q P，et al. A novel wise step strategy for firefly Algorithm[J]. International Journal of Computer Mathematics，2014，91(12)：2507-2513.

[10] 李一玄. 萤火虫算法参数研究[J]. 物流工程与管理，2015，(9)：195-197.

（上接第 111 页）

[10] 陈静杰，邹迎欢. 油耗预测中显著影响因素参数提取方法仿真[J]. 计算机仿真，2013，30(6)：55-58.

[11] 童先群，周忠眉. 基于属性值信息熵的 KNN 改进算法[J]. 计算机工程与应用，2010，46(3)：115-117.

[12] 肖辉辉，段艳明. 基于属性值相关距离的 KNN 算法的改进研究[J]. 计算机科学，2013，40(11A)：157-159，187.

[13] 王凤梅，胡丽霞. 一种基于近邻规则的缺失数据填补方法[J]. 计算机工程，2012，38(21)：53-55，62.

[14] HUANG C C，LEE H M. A grey-based nearest neighbor approach for missing attribute value prediction[J]. Applied Intelligence，2004，20(3)：239-252.

[15] MEESAD P，HENGPRAPROHM K. Combination of KNN-Based Feature Selection and KNN-Based Missing-Value Imputation of Microarry Data[C]// The 3rd International Conference on Innovative Computing Information and Control. Washington，USA：IEEE Computer Society，2008：18-20.

[16] MOORTHY K，MOHAMAD M S，DERIS S. A Review on Missing Value Imputation Algorithms for Microarray Gene Expression Data[J]. Current Bioinformatics，2014，9(1)：18-22.