

基于组合变异的果蝇优化算法

王 颖, 蔡 忱

(华北电力大学 控制与计算机工程学院, 北京 102200)

摘 要: 针对原始果蝇优化算法在寻优过程中存在易于陷入局部极值的不足, 提出一种基于组合变异的果蝇优化算法. 在算法陷入局部极值时引入组合变异策略, 更新种群飞行方向, 增强种群逃逸能力, 使算法继续迭代寻找全局极值. 选取 6 个基准函数进行实验仿真, 实验结果表明: 基于组合变异的果蝇优化算法较好平衡了算法局部搜索能力和全局开采能力, 使算法在收敛速度和收敛精度方面均获得较大提高.

关键词: 果蝇优化算法; 组合变异; 函数优化

中图分类号: TP18

文献标识码: A

文章编号: 1000-7180(2018)05-0100-05

Fruit Fly Optimization Algorithm Based on Combined Mutation

WANG Ying, CAI Chen

(School of Control and Computer Engineering, North China Electric Power University, Beijing 102200, China)

Abstract: Aiming at solving the problems of trapping into local extremes easily in the process of optimization, there is a fruit fly optimization algorithm based on combinatorial variation proposed. In this paper the combined mutation strategy is introduced when the algorithm falls into the local extreme. In the basis of this, the flight direction is updated and the population escape ability is enhanced. Thus, the algorithm continues to iterate to find the global extreme value. The experimental results show that the fruit fly optimization algorithm based on combinatorial variation is a good way to balance the algorithm's local search ability and global mining ability, and the algorithm is improved in terms of convergence speed and convergence accuracy.

Key words: fruit fly optimization algorithm; combined mutation; benchmark function

1 引言

果蝇优化算法(Fruit Fly Optimization Algorithm, FOA)具有设计简单, 设置参数少, 寻优精度高的优点, 一经提出就广泛应用到 Z-SCORE 模型、广义回归神经网络参数的优化、灰色神经网络参数的优化和对支持向量机算法的优化, 并取得较好的效果^[1-2].

果蝇算法种群飞行方向迭代更新的公式较为简单, 但参考信息单一, 因此算法在获得较好执行效率的同时, 也出现搜索能力有限、易于早熟收敛的问题. 在实际应用过程中, FOA 易受到局部极值点的束缚, 易于出现局限于复杂解空间中的局部极值的现象, 从而导致收敛速度减慢、收敛精度降低. 针对

上述问题, 提出基于组合变异的果蝇优化算法(Fruit Fly Optimization Algorithm Based on Combined Mutation, CMFOA). 通过引入组合变异算子, 结合柯西变异与高斯变异各自优点, 平衡算法局部挖掘能力与全局开采能力, 提高种群个体多样性, 解决算法早熟收敛的问题, 提高算法收敛速度和收敛精度. 通过对不同类型测试函数进行寻优仿真实验, 验证出改进算法有效提高了 FOA 算法局部挖掘能力和全局开采能力, 在求解多维函数寻优问题方面有较好表现.

2 果蝇优化算法

果蝇优化算法是一种基于果蝇觅食行为推演出

寻求全局优化的优化方法^[1]. 果蝇的嗅觉器官能灵敏查询到漂浮在空气中的各种气味,然后飞近食物位置,进而使用敏锐的视觉发现食物与同伴聚集的位置,并且往该方向飞去. 根据果蝇的觅食特性,果蝇优化算法被分为以下几个步骤.

步骤 1:初始化算法参数,设置种群规模为 size-pop,最大的迭代次数 maxgen;

步骤 2:随机初始果蝇群体位置 x_0, y_0 ;

步骤 3:给每个果蝇个体利用嗅觉觅食的随机方向与距离;

$$\begin{aligned} x_i &= x_0 + \text{step} \times \text{rand} \\ y_i &= y_0 + \text{step} \times \text{rand} \end{aligned} \tag{1}$$

步骤 4:估计食物源与原点的距离 D_i ,计算食物源味道浓度判定值 S_i :

$$\begin{aligned} D_i &= \sqrt{x_i^2 + y_i^2} \\ S_i &= 1/D_i \end{aligned} \tag{2}$$

步骤 5:计算果蝇个体位置的味道浓度 Smell_i :

$$\text{Smell}_i = f(S_i) \tag{3}$$

步骤 6:保留最优味道浓度与该食物源坐标,果蝇群体向该食物源飞行:

$$\begin{aligned} (\text{bestSmell}, \text{bestIndex}) &= \text{argmax}(\text{Smell}) \\ x_0 &= x_{\text{bestIndex}} \\ y_0 &= y_{\text{bestIndex}} \end{aligned} \tag{4}$$

步骤 7:进入迭代寻优,重复执行步骤 3~5,判断最优的味道浓度是否优于前一代的味道浓度,是则执行步骤 6,否则继续迭代直到满足终止条件.

3 基于组合变异的果蝇优化算法

3.1 相关概率分布

高斯分布是概率论与数理统计中的一种常见分布^[3],其在数学、物理及工程领域被广泛应用. 一维高斯分布的密度函数为:

$$f(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, -\infty < x < \infty \tag{5}$$

期望 μ 决定了函数峰值的位置,标准差 σ 描述了分布的幅度,因其曲线呈钟形,因此人们又经常称之为钟形曲线. 当 $\sigma = 1, \mu = 0$ 时为标准高斯分布,记作 $X \sim N(0, 1)$.

柯西分布是概率统计学中的一个极具特色的连续型分布^[3],由于柯西分布具有许多独特的性质,比如它的期望、方差均不存在等,一直是众多统计学家的关注对象. 一维柯西分布概率密度函数为:

$$f(x, \sigma, \mu) = \frac{1}{\pi} \frac{\sigma^2}{\sigma^2 + (x - \mu)^2}, -\infty < x < \infty$$

(6)

当 $\sigma = 1, \mu = 0$ 时为标准柯西分布,记作 $Z \sim P(0, 1)$.

标准高斯分布和标准柯西分布的概率密度函数曲线如图 1 和图 2 所示. 从图中可以看出与高斯分布相比,二者曲线形态均呈钟性,整体变化趋势相同,但是柯西分布具有更大的可能性在分布的两端,曲线梯度下降更为缓慢,且在 $X = \mu$ 处,取得比高斯分布更大的峰值.

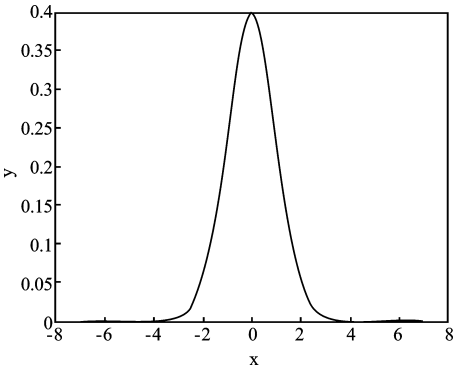


图 1 高斯分布概率密度曲线

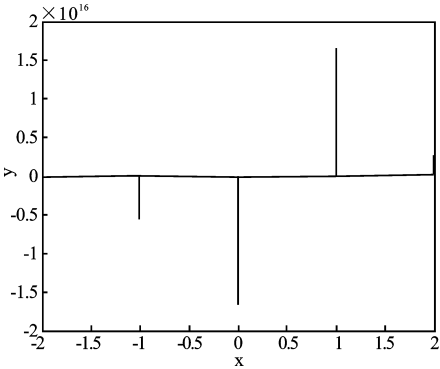


图 2 柯西分布概率密度曲线

3.2 基于组合变异的果蝇优化算法

Yao 等人说明了相比于高斯变异,根据柯西变异取样的变异步长可以赋予算法更好的全局搜索能力,这是由柯西分布本身的概率分布特性所决定的^[4]. 与此同时,果蝇算法在应用过程中存在易于陷入局部极值点和早熟收敛的问题,因此在算法迭代后期如果可以赋予算法较强的全局搜索能力,就有助于果蝇种群逃出局部极值点的束缚,增强算法的解空间勘探能力,解决算法早熟收敛的问题,从而进一步提高算法的寻优效率和精度. 将柯西分布和高斯进行自适应组合,在迭代初期着重利用高斯变异的局部挖掘能力,使果蝇个体快速锁定优质食物源,增强算法收敛速度;在迭代后期利用柯西变异强化果蝇个体全局勘探能力,降低局部极值点约束力,使

算法陷入局部极值时能迅速逃离,解决早熟收敛问题. 所以由自适应组合变异产生的扰动对果蝇优化算法具有较好的调节能力^[5]. 利用上述观点对果蝇位置更新方程进行改进,在原有果蝇个体的状态上引入服从组合分布的随机向量,如式(7)所示.

$$X_{t+1} = X_t + r_{\max} \times (\partial(t) + \beta(t))$$
$$\partial(t) = (1 - \frac{t}{T}) \times N(0,1)$$
$$\beta(t) = (\frac{t}{T}) \times C(0,1)$$

(7)

式中, X_t 是 t 时刻果蝇个体的解状态; T 是算法最大迭代次数; $r_{\max}$ 是种群最大搜索半径,该参数的取值范围受具体问题影响.

基于组合变异的果蝇优化算法的步骤如下.

步骤 1:初始化算法参数:设置种群规模为 size-

pop,最大的迭代次数 maxgen;
步骤 2:随机初始果蝇群体位置 x_0, y_0 ;
步骤 3:根据式(8),为每个果蝇个体利用嗅觉觅食的随机方向与距离;
步骤 4:根据式(2)估计食物源与原点的距离 D_i ,计算食物源味道浓度 S_i ;
步骤 5:根据式(3),计算果蝇个体位置的味道浓度 $Smell_i$;
步骤 6:根据式(4),保留最优味道浓度与该食物源坐标,果蝇群体飞向食物源;
步骤 7:进入迭代寻优,重复执行步骤 3~5,判断最优的味道浓度是否优于前一代的味道浓度,是则执行步骤 6,否则继续迭代直到满足终止条件.

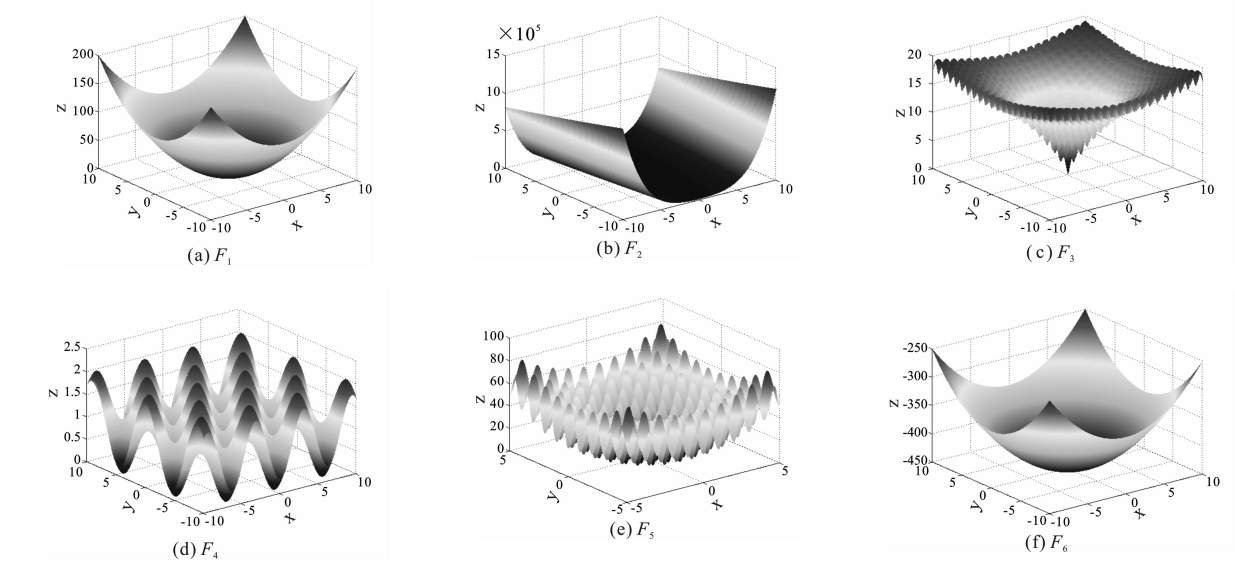


图 3 三维测试函数图像

表 1 标准函数详细信息

No.	Function	x^*	$f(x)$
F ₁	$f(x) = \sum_{i=1}^n x_i^2$	0	0
F ₂	$f(x) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i)^2 + (1 - x_i)^2)$	0	0
F ₃	$f(x) = -20 \exp(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}) - \exp(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)) + 20 + e$	0	0
F ₄	$f(x) = \frac{1}{400} \sum_{i=2}^n x_i^2 - \prod_{i=1}^n \cos(\frac{x_i}{\sqrt{i}}) + 1$	0	0
F ₅	$f(x) = \sum_{i=2}^n (x_i^2 - 10 \cos(2\pi x_i) + 10)$	0	0
F ₆	$f(x) = 418.9829n + \sum_{i=1}^n (x_i \sin(\sqrt{ x_i }))$	0	0

4 实验结果与分析

本文选择 5 个经典测试函数验证算法性能,函数具体信息见表 1,函数三维图见图 3. 选择果蝇优化算法(FOA)、改进 FOA(mFOA)^[5]和布谷鸟搜索算法(CS)^[6-7]进行对比测试. 所以仿真实验均在处理器为 Inter(R) i3-2330M、4 GB 内存的 PC 机上进行,编程软件为 MATLAB R2010a.

实验参数设置如下:四个算法的种群规模 size-pop=20,测试函数维数设置为 30,迭代次数设置为 1 000 次. 对 mFOA 和 CS 的其余参数参考相关文献进行设置.

4.1 寻优精度测试

表 2 和表 3 结果显示了 6 个测试函数在迭代次

数为 1 000 次、函数维度分别为 30 和 100 的条件下,完成 50 次寻优测试的实验结果. 实验结果表明:CMFOA 算法的寻优结果在 F_1, F_3, F_5 和 F_6 函数上均优于其余 3 种算法,但是在函数 F_2 和 F_4 上寻优精度略逊色于 mFOA 算法,这是由于函数 F_2 和 F_4 函数变化较为平缓,CMFOA 的优势得以完全发挥. 对于局部极值较多的多模态函数 F_1, F_3, F_5 和 F_6 来说,显著提高了寻优精度,并且在寻优稳定性上取得进步,寻优平均值与均方差也都优于另外 3 种算法. 通过比较发现在种群规模增大时,CMFOA 算法依然优于另外 3 种算法. 证明 CMFOA 算法拥有更好的寻优精度.

表 2 实验测试结果(种群规模=30)

	CMFOA		FOA		mFOA		CS	
	Median	Std	Median	Std	Median	Std	Median	Std
F_1	8.67×10^{-8}	5.67×10^{-8}	9.71×10^{-6}	5.50×10^{-6}	5.01×10^{-6}	1.30×10^{-12}	1.02×10^{-5}	4.08×10^{-5}
F_2	2.91×10^{-5}	8.07×10^{-6}	1.76×10^{-6}	7.85×10^{-7}	2.07×10^{-7}	2.44×10^{-13}	4.29×10^{-4}	1.78×10^{-4}
F_3	1.21×10^{-5}	5.16×10^{-5}	5.91×10^0	1.64×10^1	1.08×10^0	4.33×10^{-1}	1.40×10^1	6.63×10^0
F_4	3.18×10^{-5}	1.85×10^{-5}	1.37×10^{-6}	1.85×10^{-2}	9.27×10^{-7}	1.98×10^{-2}	2.06×10^{-4}	1.06×10^{-4}
F_5	4.71×10^{-5}	1.00×10^{-5}	2.63×10^0	1.97×10^{-2}	1.04×10^0	1.97×10^{-2}	8.78×10^0	1.16×10^1
F_6	5.21×10^{-2}	1.54×10^{-2}	6.50×10^{-1}	2.17×10^{-1}	-1.52×10^1	7.02×10^{-1}	-1.39×10^1	2.18×10^{-1}

表 3 实验测试结果(种群规模=100)

	CMFOA		FOA		mFOA		CS	
	Median	Std	Median	Std	Median	Std	Median	Std
F_1	1.57×10^{-9}	1.08×10^{-8}	8.05×10^{-6}	9.58×10^{-7}	4.22×10^{-6}	1.03×10^{-12}	1.00×10^{-5}	1.02×10^{-5}
F_2	5.01×10^{-6}	3.25×10^{-6}	2.07×10^{-6}	1.05×10^{-8}	1.33×10^{-7}	1.04×10^{-13}	5.00×10^{-4}	1.08×10^{-4}
F_3	1.01×10^{-5}	3.06×10^{-5}	4.23×10^0	1.04×10^1	1.00×10^0	5.20×10^{-1}	1.02×10^1	1.03×10^0
F_4	2.01×10^{-5}	1.58×10^{-5}	1.56×10^{-6}	1.85×10^{-2}	1.07×10^{-8}	1.08×10^{-2}	1.05×10^{-4}	9.01×10^{-5}
F_5	3.52×10^{-5}	1.03×10^{-5}	1.22×10^0	5.03×10^{-3}	1.00×10^0	1.91×10^{-2}	1.08×10^{-1}	9.06×10^0
F_6	3.20×10^{-2}	1.95×10^{-2}	5.01×10^{-1}	1.56×10^{-1}	-1.24×10^1	9.00×10^{-2}	-1.09×10^1	1.08×10^{-1}

表 4 收敛率测试结果

	CMFOA		FOA		mFOA		CS	
	Median	Rate	Median	Rate	Median	Rate	Median	Rate
F_1	32.90	1.00	286.15	0.80	90.50	1.00	685.52	0.80
F_2	51.10	1.00	958.10	0.05	71.65	1.00	∞	0.00
F_3	812.05	1.00	∞	0.00	∞	0.00	∞	0.00
F_4	56.25	1.00	549.90	0.45	94.05	1.00	921.12	0.15
F_5	791.20	1.00	∞	0.00	∞	0.00	∞	0.00
F_6	225.52	1.00	∞	0.00	878.55	1.00	∞	0.00

4.2 收敛性测试

表 4 显示了在目标精度为 1×10^{-6} 的情况下,6 个测试函数经过 50 次独立实验的平均迭代次数和成功率,其中平均迭代次数和成功率通过式(8)、式(9)计算,其中 iteration_t 表示第 t 次运行的迭代次数;表中的符号“ ∞ ”表示迭代次数超过了最大迭代次数,取 $\infty=1\ 000$.

$$\text{Median} = \frac{\sum_{t=1}^{50} \text{iteration}_t}{50},$$

(8)

$$\text{Rate} = \frac{\sum_{t=1}^{50} \alpha(\text{iteration}_t)}{50},$$

$$\begin{cases} \alpha(x) = 1, 0 < x \leqslant 1\ 000 \\ \alpha(x) = 0, x > 1\ 000 \end{cases}$$

(9)

实验结果表明:对于 6 个测试函数,CMFOA 算法平均迭代次数均明显少于 mFOA 算法、FOA 算

法和 CS 算法,寻优成功率也优于其他 3 种算法. 实验中 CMFOA 算法对测试函数的寻优成功率均达到了 100%,而 mFOA 算法虽然对于函数 F_1, F_2, F_4 和 F_6 的成功率也达到 100%,但是在函数 F_3 和 F_5 上寻优失败,没能在最大迭代次数内达到目标精度;FOA 只在函数 F_1, F_2 和 F_4 达到了目标精度,且收敛率较低,特别是在函数 F_2 上的成功率只有 5%;CS 算法的实验结果也不乐观,只在函数 F_1 和 F_4 上达到目标精度,且成功率均没有达到 100%. 在迭代次数方面,在函数 F_1, F_2, F_4 和 F_6 上,CMFOA 算法的平均迭代次数要明显少于 mFOA 算法,大大优于 FOA 和 CS 算法,证明 KM-DVFOA 算法拥有更快的收敛速度.

5 结束语

本文提出一种组合变异果蝇优化算法,该算法加入了基于柯西变异鱼高斯变异的自适应组合变异策略,在迭代初期着利用高斯变异,加强了算法局部搜索能力,增强了算法收敛速度,完成了对解空间的快速搜索;与此同时,CMFOA 算法在迭代后期强化柯西变异的作用,提高算法后期勘察能力,使算法获得变异能力,完成对解空间详尽搜索,提高了算法的逃逸能力和全局寻优能力. 对不同类型测试函数进行寻优仿真实验,并与其他改进 FOA 算法和其他演化算法进行比较,结果说明:改进算法都可以显著提高其搜索精度和收敛速度. 但是,CMFOA 算法在解决 FOA 算法早熟收敛问题的同时,也存在搜索稳定性不足和无法解决离散型寻优问题的问题,如何解决这些问题将成为下一阶段研究的重点.

参考文献:

- [1] Li H, Guo S, Li C, et al. A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm[J]. Knowl.-Based Syst., 2013, 37: 378-387.
- [2] Pan W T. Mixed modified fruit fly optimization algorithm with general regression neural network to build oil and gold prices forecasting model[J]. Kybernetes, 2014, 43(7):1053-1063.
- [3] 杜晓昕,张剑飞,郭媛,等. 基于柯西-高斯动态消减变异的果蝇优化算法研究[J]. 计算机工程与科学. 2016, 38 (6) :1171-1176.
- [4] Yao X, Liu Y. Fast evolutionary programming. [C]// Evolutionary Programming VI. Berlin: Springer, 1997: 151-161.
- [5] Pan Q K, Sang H Y, Duan J H, et al., An improved fruit fly optimization algorithm for continuous function optimization problems[J]. Knowledge-Based System, 2014, 62:69-83.
- [6] Pan W T, Pan, WT. A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example. Knowledge-Based Systems [J]. Knowledge-Based Systems, 2012, 26(2):69-74.
- [7] Li H, Guo S, Zhao H, et al. Annual electric load forecasting by a least squares support vector machine with a fruit fly optimization algorithm[J]. Energies, 2012, 5(12):4430-4445.

作者简介:

王 颖 女,(1954-),教授. 研究方向为数据库技术与网络安全、电力系统信息化、数据挖掘.

E-mail:327276364@qq. con.

蔡 忱 女,(1993-),硕士研究生. 研究方向为数据库技术与网络安全、数据挖掘.