

DOI:10.11918/j.issn.0367-6234.201811027

Tent 混沌和模拟退火改进的飞蛾扑火优化算法

岳龙飞¹, 杨任农¹, 张一杰², 于洋³, 张振兴¹

(1. 空军工程大学 空管领航学院, 西安 710051; 2. 西北工业大学 电子信息学院, 西安 710129; 3. 中国人民解放军 95810 部队, 北京 100076)

摘要: 针对标准飞蛾扑火优化算法存在的易陷入局部最优陷阱、全局寻优能力不足的问题, 借鉴混沌序列、模拟退火算法和遗传算法, 提出 Tent 混沌和模拟退火改进的飞蛾扑火优化算法。首先, 通过 Tent 混沌序列初始化种群, 增加种群多样性; 然后对当前最优解增加扰动产生新解, 并与当前最优解按比例杂交相加, 根据模拟退火算法中的 Metropolis 准则判断是否接受杂交后的新解, 最终获得最优解。分别使用复杂高维基准函数和航迹规划问题测试算法性能。其中, 6 个复杂基准函数寻优测试结果表明, 对于 10 维基准函数, 该算法经过约 0.25 秒收敛到最优值; 对于 50 维基准函数, 该算法经过约 0.5 秒收敛到最优值。与标准飞蛾扑火优化算法和其它智能优化算法相比, 该算法能够有效跳出局部最优解, 寻优精度更高, 收敛速度更快。航迹规划仿真表明, 对有 4 个禁飞区和 2 个威胁源的空域环境, 该算法经过大约 100 次迭代可以得到最优航迹, 与标准飞蛾扑火优化算法相比精度更高, 具有实际应用价值。因此, 该算法具有更好的寻优性能。

关键词: 飞蛾扑火优化算法; Tent 混沌; 模拟退火; 智能优化算法; 混沌序列

中图分类号: V247

文献标志码: A

文章编号: 0367-6234(2019)05-0146-09

Tent chaos and simulated annealing improved moth-flame optimization algorithm

YUE Longfei¹, YANG Renrong¹, ZHANG Yijie², YU Yang³, ZHANG Zhenxing¹

(1. Air Traffic Control and Navigation College, Air Force Engineering University, Xi'an 710051, China; 2. School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710129, China; 3. Unit 95810 of the Chinese People's Liberation Army, Beijing 100076, China)

Abstract: Aiming at the problem that the moth-flame optimization algorithm is easy to fall into the local optimal trap and the global optimization ability is insufficient, chaotic sequences, simulated annealing algorithm, and genetic algorithm were utilized to propose a tent chaos and simulated annealing improved moth-flame optimization algorithm. First, the tent chaotic sequences were used to initialize the population and increase the diversity of the population. Then the new solution was obtained by adding disturbance to the current optimal solution which was further proportionally hybridized to generate a final new solution. According to the Metropolis criterion in the simulated annealing algorithm, the ultimate solution was finally obtained by determining whether the new hybridized solution was accepted. The performance of the proposed algorithm was tested through complex high-dimensional benchmark functions and trajectory planning problems. The results of six complex benchmark function optimization tests show that for the 10-dimensional benchmark function, the algorithm converged to the optimal value after about 0.25 seconds, and for 50 dimensions, it was about 0.5 seconds. Compared with the standard moth-flame optimization algorithm and other intelligent optimization algorithms, the proposed algorithm could effectively jump out of the local optimal solution in the discussed scenarios, and the accuracy precision and the convergence speed were improved in certain degree. The trajectory planning simulation results show that for an airspace environment with four no-fly zones and two threat sources, the proposed algorithm could obtain the optimal trajectory after about 100 iterations, which is more accurate than the standard moth-flame optimization algorithm. Therefore, it is valuable for engineering applications and has better performance.

Keywords: moth-flame optimization algorithm; tent chaos; simulated annealing; intelligent optimization algorithm; chaotic sequences

飞蛾扑火优化算法^[1] (Moth-flame optimization algorithm, MFO) 是澳大利亚 Seyedali Mirjalili 于

2015 年提出的一种新型智能优化算法^[2-3], 具有调节参数少、收敛较快、实现简单等优点, 因此常被用于参数优化^[4]、最优化计算^[5]。与改进遗传算法^[5]、粒子群算法^[6]、蝙蝠算法^[6]相比, MFO 收敛性能更优, 但其在寻优过程中也存在一些不足, 如每次迭代中都选取当前最优解导致容易陷入局部最优而早熟

收稿日期: 2018-11-04

基金项目: 国家自然科学基金资助项目 (61503409)

作者简介: 岳龙飞 (1995—), 男, 硕士研究生;

杨任农 (1969—), 男, 教授, 博士生导师

通信作者: 杨任农, ylf1172969690@163.com

收敛、由于种群多样性低导致全局寻优性能较差等. 针对此类问题, 罗钧等^[7]通过引入 Logistic 混沌序列初始化, 提高了解的多样性, 增强了算法的寻优性能; 匡芳君等^[8]结合混沌优化算法改进人工蜂群算法, 改进了算法的收敛性能, 避免陷入局部最优; 薛永生等^[9]提出用模拟退火算法改进粒子群算法, 能够有效跳出局部最优解, 得到近似全局最优解; 李志明等^[10]提出用 Lévy 飞行轨迹改进的飞蛾扑火优化算法, 提高了收敛速度和寻优精度.

针对 MFO 算法存在的缺陷, 本文借鉴 Tent 混沌优化、模拟退火算法和遗传算法中的算术交叉操作, 提出 Tent 混沌和模拟退火改进的飞蛾扑火优化算法 (Tent chaos and simulated annealing improved moth-flame optimization algorithm, TC-SA-MFO). 该算法首先通过 Tent 混沌映射初始化种群, 使初始种群尽可能地均匀分布于整个解空间, 增加解的多样性; 然后对每次迭代中当前最优解, 通过模拟退火机制产生新解, 并将当前最优解和新解按比例加权产生最终的新解, 然后以一定的概率接受新解, 提高其跳出局部最优的能力. 仿真结果表明, 该算法能够有效增加种群的多样性, 寻优时能够跳出局部最优陷阱, 与标准 MFO 相比, 收敛速度更快, 寻优精度更高.

1 飞蛾扑火优化算法

1.1 仿生原理

MFO 的灵感源于飞蛾夜间飞行中的横向定位^[1]. 飞蛾飞行时常保持与月亮的夹角不变, 由于与月亮距离很远, 所以飞蛾与月亮不同时刻连线一直平行, 使飞蛾可以沿直线飞行, 见图 1 所示^[1].

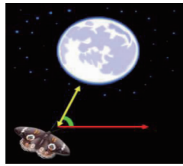


图 1 横向定位飞行
Fig. 1 Lateral positioning flight



图 2 飞蛾扑火
Fig. 2 Moth to the flame

然而在现实中, 飞蛾与火焰距离很近, 飞蛾仍然保持与火焰的夹角不变, 使得飞蛾沿着等角螺线向火焰飞行, 于是就有了“飞蛾扑火”, 见图 2^[1]所示.

Mirjalili 正是受到这个等角螺线函数的启发从而发明了 MFO 算法.

1.2 MFO 算法

MFO 算法中, M 为飞蛾矩阵, 即搜索解的智能体, Z_M 为飞蛾适应度值矩阵, F 为火焰位置矩阵, 即当前最优解, Z_F 为火焰适应度值矩阵, 如式 (1) 和 (2) 所示. 初始时刻火焰矩阵和飞蛾矩阵维度相同.

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,d} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,d} \\ \vdots & \vdots & & \vdots \\ m_{n,1} & m_{n,2} & \cdots & m_{n,d} \end{bmatrix}, Z_M = \begin{bmatrix} z_{m1} \\ z_{m2} \\ \vdots \\ z_{mn} \end{bmatrix}; \quad (1)$$

$$F = \begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,d} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,d} \\ \vdots & \vdots & & \vdots \\ f_{n,1} & f_{n,2} & \cdots & f_{n,d} \end{bmatrix}, Z_F = \begin{bmatrix} z_{f1} \\ z_{f2} \\ \vdots \\ z_{fn} \end{bmatrix}. \quad (2)$$

式中: n 为飞蛾的个数, d 为维度即待求变量的个数.

火焰位置根据适应度值从小到大排序, 飞蛾分别围绕排序后的火焰, 沿式 (3) 等角螺线飞行, 并通过改变参数 t 更新其位置.

$$M_i = D_i e^{bt} \cos(2\pi t) + F_j. \quad (3)$$

式中: M_i 为第 i 只飞蛾新的位置, $D_i = |M_i - F_j|$ 为第 i 只飞蛾与第 j 个火焰的距离, b 为等角螺线参数, 决定等角螺线的形状, t 为 $[-1, 1]$ 之间的随机数, 控制飞蛾与火焰的距离, 见图 3 所示. t 越小, 飞蛾离火焰越近, 通过改变 t , 飞蛾可以到达火焰周围的位置, 增强了算法的局部寻优能力.

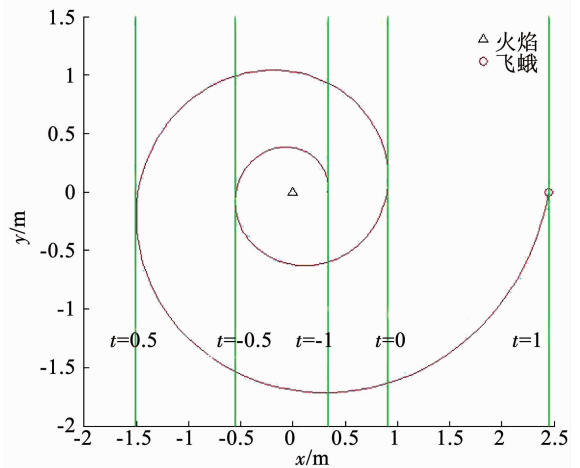


图 3 飞蛾位置随 t 变化
Fig. 3 Variation of the position of moth with t

为提高算法的搜索效率, MFO 采用火焰自适应减少机制, 使得后期收敛更快, 而不用在劣解周围继续寻优. 火焰数量自适应减少公式为

$$N_F = \text{round}(N_{\max} - I \times \frac{N_{\max} - 1}{I_{\max}}). \quad (4)$$

式中: N_F 为当前火焰数量, $N_{\max}$ 为最大火焰数量, I 为当前迭代次数, $I_{\max}$ 为最大迭代次数. 火焰数量随迭代次数减少变化见图 4 所示.

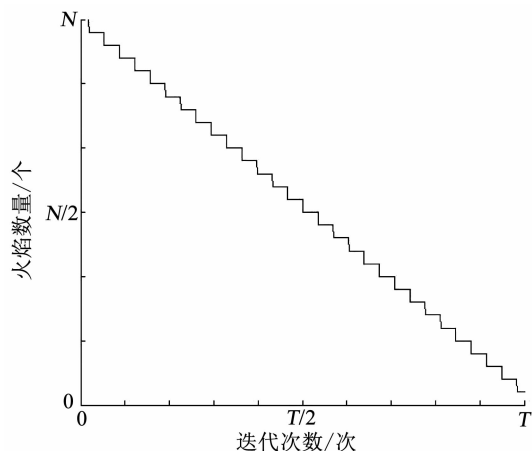


图 4 火焰数量自适应减少

Fig. 4 Decreases of the Number of flames with the number of iterations

当火焰数量少于飞蛾数量时, 飞蛾围绕适应度最差的火焰飞行. MFO 算法流程如下:

- 1) 参数初始化. 设置飞蛾数量为 N_M , 最大迭代次数为 $I_{\max}$, 求解问题维度为 d , 等角螺线参数为 b .
- 2) 飞蛾位置初始化, 即在搜索空间中随机生成飞蛾位置, 迭代次数 $I = 1$.
- 3) 将飞蛾位置按适应度值从小到大排序, 赋给火焰, 作为排序火焰位置.
- 4) 根据式(3), 未排序的飞蛾分别围绕排序后的火焰更新其位置.
- 5) 记录当前最优火焰适应度值.
- 6) 根据式(4)减少火焰数量, $I = I + 1$.
- 7) 判断是否达到最大迭代次数, 若达到则输出最优火焰位置 and 对应适应度值, 否则转到 3).

2 Tent 混沌和模拟退火改进的飞蛾扑火优化算法

2.1 Tent 混沌序列

混沌序列具有遍历性和随机性, 利用混沌序列可以使初始解分布更加均匀, 有利于找到全局最优解. 目前常用的混沌序列有 Logistic 映射和 Tent 映射两种. 单梁等^[11] 通过研究证明 Tent 映射较 Logistic 映射遍历性更好, 因此本文选用 Tent 映射优化初始种群. Tent 映射和 Logistic 映射分布见图 5 所示.

Tent 映射公式为

$$x_{i+1} = \begin{cases} 2x_i, & 0 \leq x_i \leq 0.5; \\ 2(1 - x_i), & 0.5 < x_i \leq 1. \end{cases} \quad (5)$$

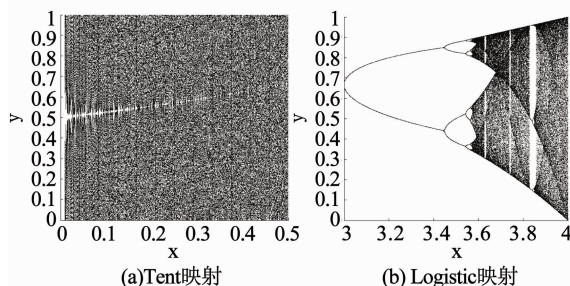


图 5 Tent 和 Logistic 映射分布对比

Fig. 5 Comparison of tent map and logistic map

2.2 模拟退火算法

模拟退火算法是 Metropolis 发现物理中固体物质的退火过程与组合优化问题的相似性而提出的一种优化算法. 物理退火过程根据温度变化可分为加温过程、等温过程和冷却过程 3 部分, 其中, 加温过程对应算法中的设置初始温度. 等温过程对应算法中的 Metropolis 抽样, 冷却过程对应减小算法的控制参数. 物质的能量对应目标函数, 能量最低态就是待求的目标函数最优值. Metropolis 准则是模拟退火算法的核心, 它以一定的概率接受当前的劣解, 从而使算法具有能够跳出局部最优, 收敛于全局最优解的能力.

模拟退火算法流程如下(以最小化问题为例):

- 1) 初始化: 设置初始温度为 T_0 , 令 $T = T_0$, 任取初始解 S_1 , 确定每个 T 时的迭代次数, 即 Metropolis 链长 L .
- 2) 对当前温度 T 和 $k = 1, 2, \dots, L$, 重复步骤 3) ~ 6).
- 3) 给当前解 S_1 一个随机扰动, 得到新解 S_2 ;
- 4) 计算增量, $df = f(S_2) - f(S_1)$ 为 S_1 的代价函数.

5) 根据 Metropolis 准则, 公式为

$$P = \begin{cases} 1, & df < 0; \\ e^{-\frac{df}{T}}, & df \geq 0. \end{cases} \quad (6)$$

如果 $df < 0$, 则接受新解; 否则, 以 $e^{-\frac{df}{T}}$ 的概率接受新解.

6) 判断是否达到终止条件, 如果是则输出当前解 S_1 为最优解, 然后结束; 否则减小温度 T 后返回步骤 2). 终止条件为: 在连续多个 Metropolis 链中, 都没有接受新解 S_2 , 或达到终止温度.

2.3 Tent 混沌和模拟退火改进的飞蛾扑火优化算法流程

TCSA-MFO 算法通过引入 Tent 混沌映射初始

化种群、按照模拟退火机制以一定概率接受劣解,借鉴遗传算法中的算术杂交操作,使算法具有一定的跳出局部最优的能力,从而避免早熟收敛.其中,杂交指对模拟退火产生的新解和旧解按比例杂交加权,得到最终新解.这样得到新解既有利于保留旧解的优点,又能减小扰动误差的影响.公式为

$$S_2 = a \times S_2 + (1 - a) \times S_1. \tag{7}$$

TCSA-MFO 算法流程图见图 6 所示,具体流程如下:

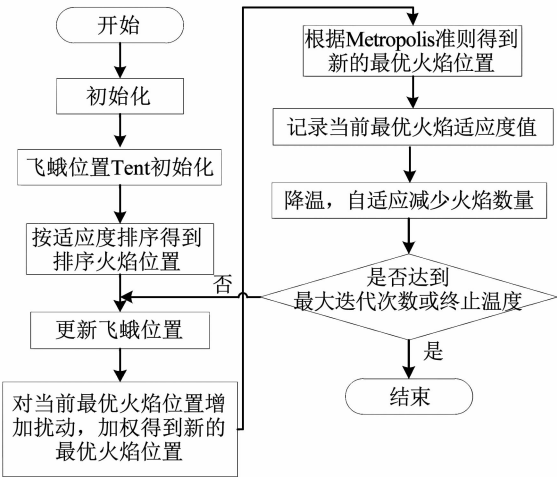


图 6 TCSA-MFO 算法流程图
Fig. 6 TCSA-MFO flow diagram

- 1) 参数初始化. 设置飞蛾数量为 N_M , 最大迭代次数为 $I_{\max}$, 求解问题维度为 d , 等角螺旋线参数为 b .
- 2) 飞蛾位置初始化, 即在搜索空间中随机生成飞蛾位置后, 根据式 (5) 进行 Tent 映射, 迭代次数 $I=1$, 设置模拟退火初始温度 T_0 , 降温速率 q .
- 3) 将飞蛾位置按适应度值从小到大排序赋给

- 火焰, 作为排序火焰位置.
- 4) 根据式 (3), 未排序的飞蛾分别围绕排序后的火焰更新其位置.
- 5) 对当前最优火焰位置 S_1 根据式 (8) 产生扰动, 即给当前最优火焰位置加上一个偏移量, 该偏移量为当前温度与随机数映射到 $(-1, 1)$ 区间的乘积, 得到新的火焰位置 S_2 , 并根据式 (7) 加权得到最终新的火焰位置 S_2 .

$$S_2 = S_1 + T \times \frac{\text{randn}(1, d)}{\text{norm}(\text{randn}(1, d))}. \tag{8}$$

- 6) 计算 $df=f(S_2)-f(S_1)$, 根据式 (6), 如果 df 小于 0, 则接受 S_2 , 否则以 $e^{-\frac{df}{T}}$ 的概率接受 S_2 .
- 7) 记录当前最优火焰适应度值.
- 8) 根据式 (4) 减少火焰数量, $T=q \times T, I=I+1$.
- 9) 判断是否达到最大迭代次数或终止温度, 若达到则输出最优火焰位置 and 对应适应度值, 否则转到 3).

3 仿真实验与结果分析

3.1 基准测试函数

本文选取 CEC2010 中的 6 个经典基准测试函数^[12]来测试算法寻优性能, 见图 7 和表 1 所示. 其中, Sphere 函数是平滑的单调函数, 只有一个全局最小值, 用于测试算法的收敛速度; Rosenbrock 函数是非凸单峰函数, 有多个局部极值, 主要用于测试算法的收敛速度和寻优精度; Schwefel2.26、Rastrigin、Ackley 和 Griewank 函数都是复杂的多峰函数, 有多个局部极小值, 主要用来测试算法的跳出局部极值、避免早熟收敛的全局寻优能力. 这些函数可以较为全面地考察算法的寻优性能.

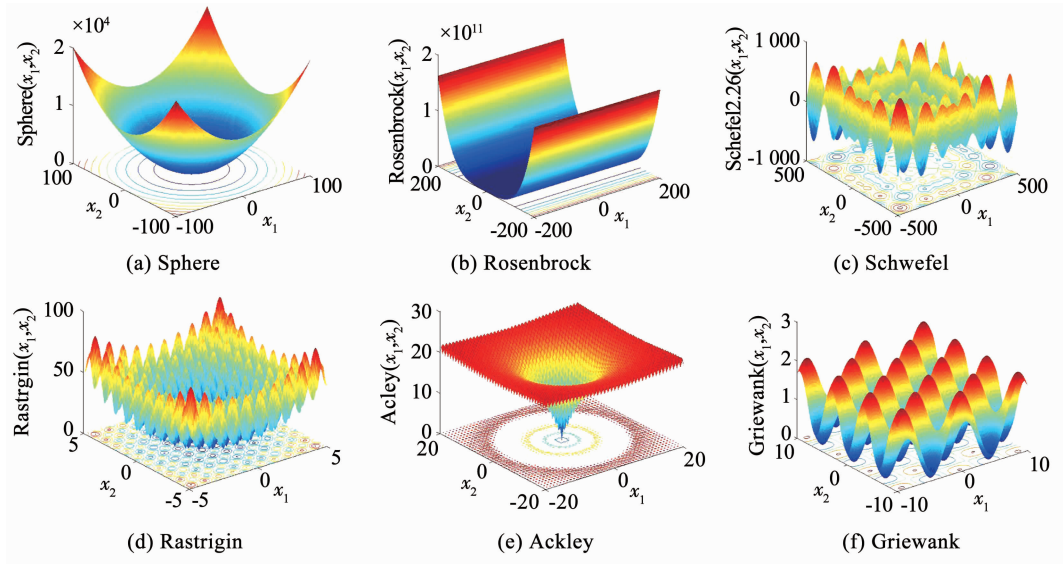


图 7 基准函数

Fig. 7 Benchmark functions

3.2 TCSA-MFO 算法的性能分析

3.2.1 基准函数寻优性能分析

为了评价 TCSA-MFO 算法的有效性,本文用上
述基准测试函数进行测试,并与 MFO 算法进行对
比.采用 MATLAB R2014a 进行仿真,参数设置为:
飞蛾种群数量为 30,最大迭代次数为 1 000,螺旋线
参数为 1,模拟退火初始温度为 100,终止温度为
0.001,退火系数为 0.99,新解与旧解加权系数为
0.5.为了减小误差,取连续运行 50 次得到的适应度
均值作为测试结果.

10 维基准函数测试的适应度收敛曲线见图 8 ~
13 所示;10 维和 50 维基准函数测试的适应度均值、
标准差、收敛时间和迭代次数见表 2 所示.

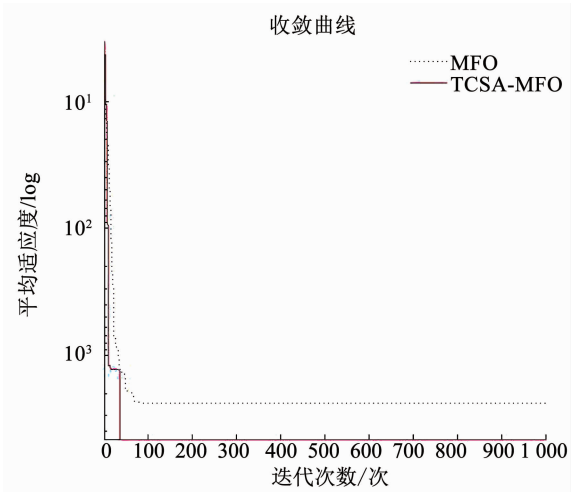


图 10 Schwefel 寻优收敛曲线(10 维)

Fig. 10 Convergence curves of Schwefel (10 dimensions)

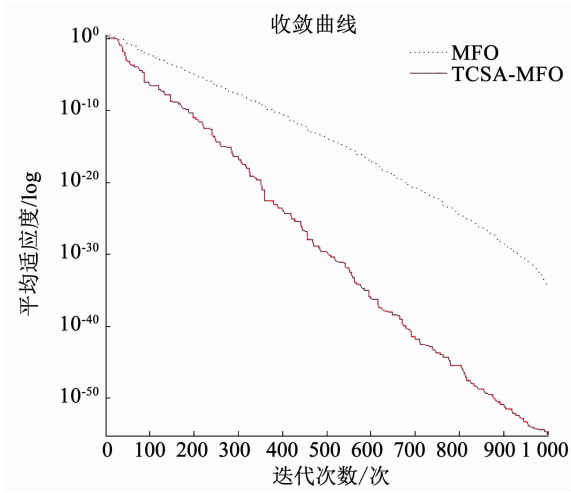


图 8 Sphere 寻优收敛曲线(10 维)

Fig. 8 Convergence curves of Sphere (10 dimensions)

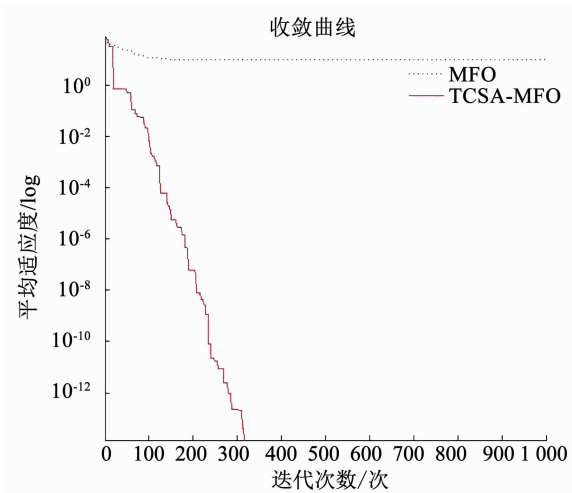


图 11 Rastrigin 寻优收敛曲线(10 维)

Fig. 11 Convergence curves of Rastrigin (10 dimensions)

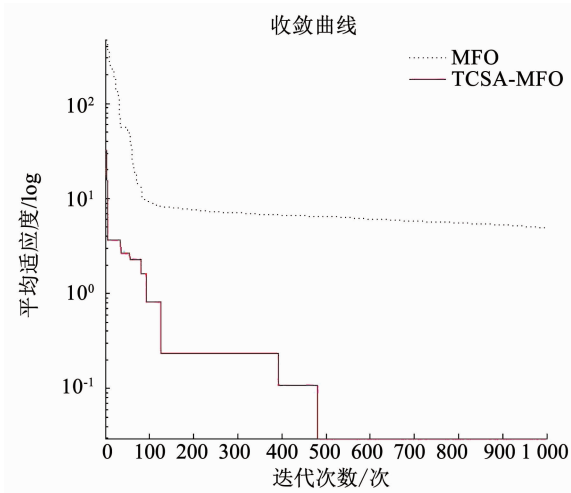


图 9 Rosenbrock 寻优收敛曲线(10 维)

Fig. 9 Convergence curves of Rosenbrock (10 dimensions)

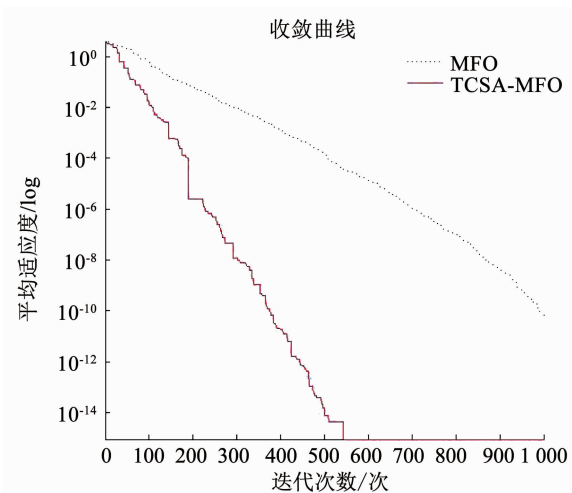


图 12 Ackley 寻优收敛曲线(10 维)

Fig. 12 Convergence curves of Ackley (10 dimensions)

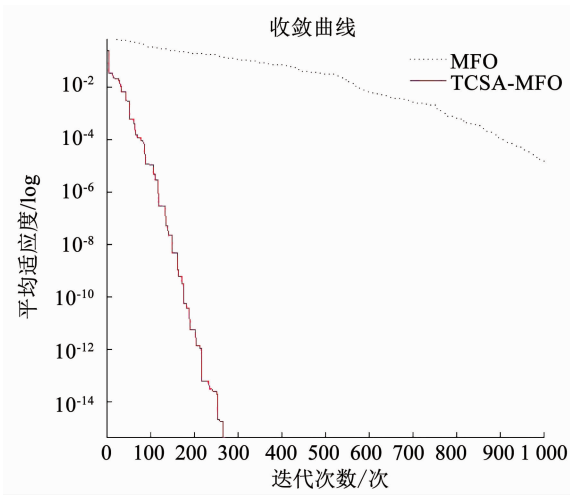


图 13 Griewank 寻优收敛曲线(10 维)

Fig. 13 Convergence curves of Griewank (10 dimensions)

由图 8 ~ 13 可知,对于单峰函数,TCSA-MFO 算

法比 MFO 算法收敛速度更快,寻优精度也更高;对于多峰函数,MFO 算法基本都陷入局部最优值,收敛速度慢,而 TCSA-MFO 算法可以跳出局部最优陷阱,一直搜索最优解,具有良好的全局寻优性能,收敛速度更快. 由表 2 可知,TCSA-MFO 算法的寻优的均值和标准差都比 MFO 算法的寻优均值和标准差小,证明了 TCSA-MFO 算法性能更优,且鲁棒性好. 尤其是对于 Schwefel 函数和 Rastrigin 函数,MFO 算法明显早熟收敛,提前停滞于局部最优,而 TCSA-MFO 几乎达到了全局最小值.

为进一步测试 TCSA-MFO 算法的性能,本文选取前文提到的改进遗传算法^[5] (EGA)、粒子群算法^[6] (PSO)、蝙蝠算法^[6] (BA) 进行对比,用 50 维 Sphere 和 Rastrigin 函数测试,种群数量为 30,最大迭代次数为 1 000. 4 种算法寻优时的收敛曲线见图 14和 15 所示.

表 1 基准测试函数

Tab. 1 Benchmark functions

函数名	表达式及取值范围	全局最小值	特性
Sphere	$f(X) = \sum_{i=1}^D x_i^2, x_i \in [-100, 100]$	$x_i = 0, f(X) = 0$	单峰
Rosenbrock	$f(X) = \sum_{i=1}^{D-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2), x_i \in [-30, 30]$	$x_i = 1, f(X) = 0$	单峰
Schwefel2.26	$f(X) = -\sum_{i=1}^D (x_i \sin \sqrt{ x_i }), x_i \in [-500, 500]$	$x_i = 420.968\ 7, f(X) = -418.982\ 9\ D$	多峰
Rastrigin	$f(X) = \sum_{i=1}^D (100(x_i^2 - 10\cos(2\pi x_i) + 10)), x_i \in [-5.12, 5.12]$	$x_i = 0, f(X) = 0$	多峰
Ackley	$f(X) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^D x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^D \cos(2\pi x_i) + 20 + e), x_i \in [-32, 32]$	$x_i = 0, f(X) = 0$	多峰
Griewank	$f(X) = \frac{1}{1\ 000}\sum_{i=1}^D x_i^2 - \prod_{i=1}^D \cos \frac{x_i}{\sqrt{i}} + 1, x_i \in [-600, 600]$	$x_i = 0, f(X) = 0$	多峰

表 2 基准测试函数优化结果对比

Tab. 2 Comparison of optimization results of benchmark functions

函数	维数	算法	均值	标准差	收敛时间/s	迭代次数/次
Sphere	10	MFO	8.738×10^{-35}	8.043×10^{-35}	0.25	396
		TCSA-MFO	6.544×10^{-55}	1.936×10^{-50}	0.21	211
	50	MFO	9.965×10^{-3}	4.507×10^{-2}	0.52	453
		TCSA-MFO	3.346×10^{-50}	1.672×10^{-48}	0.45	274
Rosenbrock	10	MFO	7.971	1.463	0.35	536
		TCSA-MFO	8.954×10^{-1}	3.041×10^{-1}	0.30	114
	50	MFO	4.873×10^1	1.676	0.62	757
		TCSA-MFO	9.580×10^{-1}	1.637	0.58	265

续表 2

函数	维数	算法	均值	标准差	收敛时间/s	迭代次数/次
Schwefel2.26	10	MFO	-0.999×10^3	1.649×10^3	0.26	77
		TCSA-MFO	-4.188×10^3	1.403	0.23	41
	50	MFO	-1.392×10^4	7.090×10^2	0.50	194
		TCSA-MFO	-2.094×10^4	5.981×10^{-2}	0.48	87
Rastrigin	10	MFO	5.597	6.085	0.38	142
		TCSA-MFO	4.967×10^{-16}	3.455×10^{-15}	0.30	113
	50	MFO	7.355×10^1	9.458	0.52	264
		TCSA-MFO	3.678×10^{-14}	5.279×10^{-11}	0.47	195
Ackley	10	MFO	4.209×10^{-15}	2.621×10^{-16}	0.35	315
		TCSA-MFO	8.419×10^{-16}	3.847×10^{-17}	0.31	248
	50	MFO	7.118×10^{-2}	2.937×10^{-1}	0.59	574
		TCSA-MFO	8.882×10^{-16}	5.580×10^{-18}	0.56	399
Griewank	10	MFO	1.314×10^{-4}	8.359×10^{-4}	0.45	412
		TCSA-MFO	3.177×10^{-17}	3.564×10^{-16}	0.31	157
	50	MFO	1.129×10^{-4}	9.423×10^{-4}	0.53	631
		TCSA-MFO	1.761×10^{-17}	2.965×10^{-15}	0.50	334

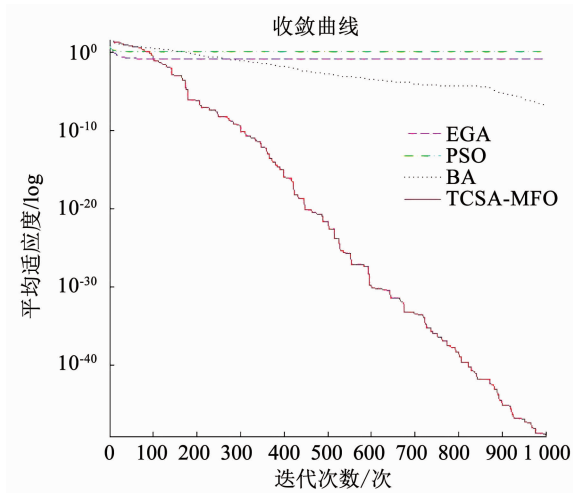


图 14 Sphere 寻优收敛曲线(50 维)

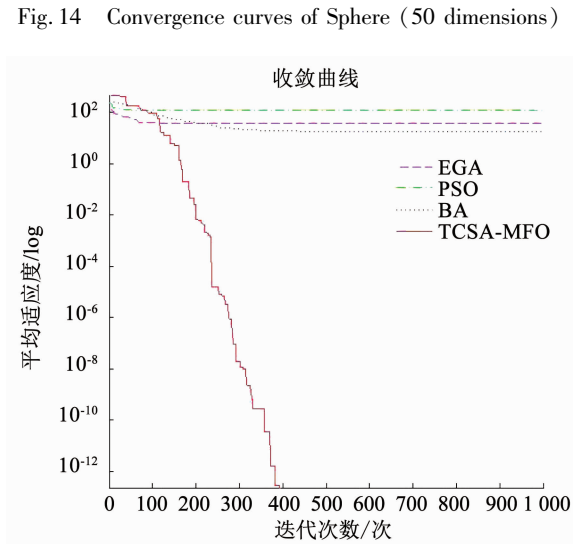


图 15 Rastrigin 寻优收敛曲线(50 维)

Fig. 15 Convergence curves of Rastrigin (50 dimensions)

由图 14 和 15 可知,对于 50 维 Sphere 函数和 Rastrigin 函数寻优,TCSA-MFO 相比其他三种智能优化算法,收敛速度更快,寻优精度更高.

综上可知,TCSA-MFO 算法由于引进 Tent 混沌序列初始化和模拟退火中的跳出机制,在函数寻优中具有较好的跳出局部最优的能力,全局寻优性能更优.

3.2.2 航迹规划问题性能分析

TCSA-MFO 作为一种智能优化算法,可以用于解决多种优化问题,比如航迹规划、目标分配等. 本文选择航迹规划问题对算法性能进一步测试.

航迹规划问题一般等效为在给定离散空间中,从起点到终点的一条可行路径的搜索问题. 借鉴向祖全等^[13]研究成果,建立航迹规划模型. 选择适应度函数为航迹长度,函数值越小航迹越优,当航迹与禁飞区等交叉时函数值为无穷大. 计算公式为

$$P = L_{SP_1} + \sum_{i=1}^{n-1} L_i L_{i+1} + L_{P_n G} = \sum_{i=0}^n L_{P_i} L_{P_{i+1}}, \quad (9)$$
$$F(i) = \begin{cases} L_{SG} = \sum_{i=0}^n L_{P_i} L_{P_{i+1}}, & L_{P_i} L_{P_{i+1}} \cap D_{NF} = \emptyset; \\ +\infty & L_{P_i} L_{P_{i+1}} \cap D_{NF} \neq \emptyset. \end{cases} \quad (10)$$

式中: S 为起点, G 为终点, $L_{P_i} L_{P_{i+1}}$ 为航迹点 i 和航迹点 $i+1$ 之间的距离, D_{NF} 为禁飞区和威胁源(当飞机进入威胁源范围内时容易被击落,所以认为威胁源为不可进入区域).

实验设置:以航迹起点为坐标原点,建立极坐标系,终点为(0.785 rad,50 km),禁飞区和威胁源位置参数见表3所示.种群数量为30,搜索维度为4,最大迭代次数为1 000.TASA-MFO 算法得到的最优航迹见图16和17所示.

表3 禁飞区和威胁源参数

Tab.3 Parameters of no-fly zone and threat source

极角/rad	极径/km	直径/km
0.785	10	5
0.4	20	8
1.0	30	15
0.5	40	10
0.85	20	10
0.6	30	8

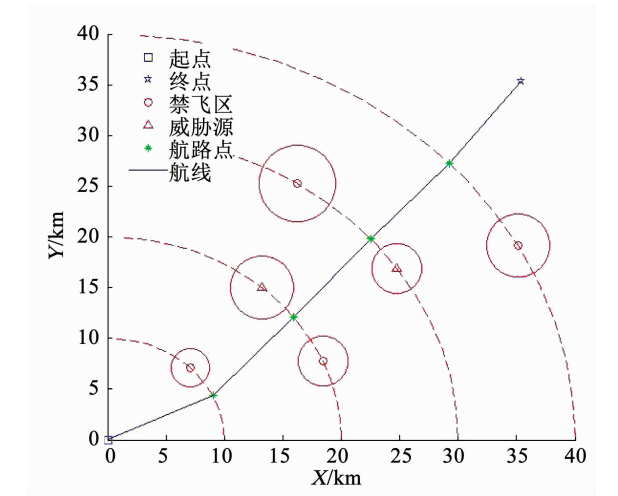


图16 TCSA-MFO 算法规划最优航迹

Fig.16 Optimal trajectory obtained by TCSA-MFO algorithm

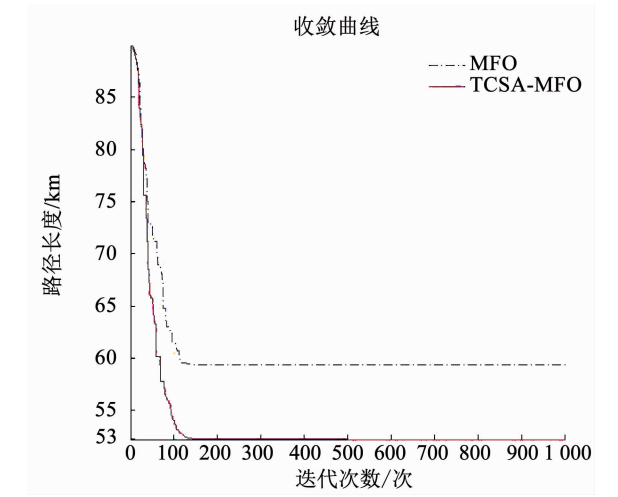


图17 航迹长度收敛曲线

Fig.17 Convergence curve of route length

仿真结果表明,与标准 MFO 算法相比,TASA-MFO 算法经过 100 代基本可以得到最优航迹长度

为 53 km,而 MFO 算法在 60 km 时基本陷入了局部最优值.因此,TCSA-MFO 算法可以用于求解航迹规划问题,搜索精度较高,航迹较优.

4 结论

针对标准 MFO 算法存在的问题,本文提出了一种改进的飞蛾扑火优化算法(TCSA-MFO),通过引入 Tent 混沌映射初始化和模拟退火中的 Metropolis 准则,改善了算法的寻优性能.该算法相比 MFO 算法主要有以下特点:

- 1)引入 Tent 混沌映射初始化种群,使种群尽可能均匀分布在整個解空间.
 - 2)引入模拟退火机制,对迭代中的当前最优解产生扰动,生成新解,然后根据 Metropolis 准则判断是否接受该新解,即以一定的概率接受劣解,增强算法跳出局部最优能力.
 - 3)对扰动生成的新解和当前最优解杂交相加,得到最终的新解,既保留当前最优解的优势,又增加局部寻优的能力.
 - 4)6 个基准函数寻优测试和航迹规划性能测试表明,TCSA-MFO 算法在寻优过程中,因 Tent 初始化和模拟退火改进能防止早熟收敛,具备良好的全局寻优能力,收敛速度较快,精度较高,鲁棒性较好.
- 下一步研究如何将 TCSA-MFO 算法应用到更复杂的优化问题中去,并学习借鉴其他智能优化算法,提出性能更优的全局优化算法.

参考文献

[1] MIRJALILI S. Moth-flame optimization algorithm; A novel nature-inspired heuristic paradigm[J]. Knowledge-e-Based Systems, 2015, 89: 228. DOI:10.1016/j.knosys.2015.07.006

[2] MIRJALILI S. The ant lion optimizer[J]. Advances in Engineering Software, 2015, 83 (C): 80. DOI:10.1016/j.advengsoft.2015.01.010

[3] MIRJALILI S, MIRJALILI S M, LEWIS A. Grey wolf optimizer[J]. Advances in Engineering Software, 2014, 69 (3): 46. DOI:10.1016/j.advengsoft.2013.12.007

[4] 包义钊,殷保群,曹杰,等.基于飞蛾-烛火优化算法的贝叶斯网络结构学习[J].计算机工程,2018,44(1):187. DOI:10.3969/j.issn.1000-3428.2018.01.032

BAO Yizhao, YIN Baoqun, CAO Jie, et al. Bayesian network structure learning based on moth-flame optimization algorithm[J]. Computer Engineering, 2018, 44(1): 187. DOI:10.3969/j.issn.1000-3428.2018.01.032

[5] 王子琪,陈金富,张国芳,等.基于飞蛾扑火优化算法的电力系统最优潮流计算[J].电网技术,2017,41(11):3641. DOI:10.13335/j.1000-3673.pst.2017.0657

WANG Ziqi, CHEN Jinfu, ZHANG Guofang, et al. Optimal power flow calculation with moth-flame optimization algorithm[J]. Power System Technology, 2017, 41 (11): 3641. DOI:10.13335/j.1000-

3673. pst. 2017. 0657
- [6] 李志明, 莫愿斌, 张森. 一种新颖的群智能优化算法: 飞蛾扑火优化算法[J]. 电脑知识与技术, 2016, 12(31): 172. DOI:10.14004/j.cnki.ckt.2016.4156
- LI Zhiming, MO Yuanbin, ZHANG Sen. A novel swarm intelligence optimization algorithm: Moth-flame optimization algorithm[J]. Computer Knowledge and Technology, 2016, 12(31): 172. DOI: 10.14004/j.cnki.ckt.2016.4156
- [7] 罗钧, 李研. 具有混沌搜索策略的蜂群优化算法[J]. 控制与决策, 2010, 25(12): 1913–1916. DOI:10.13195/j.cd.2010.12.156. luoj. 013
- LUO Jun, LI Yan. Artificial bee colony algorithm with chaotic-search strategy[J]. Control and Decision, 2010, 25(12): 1913. DOI:10.13195/j.cd.2010.12.156. luoj. 013
- [8] 匡芳君, 徐蔚鸿, 金忠. 自适应 Tent 混沌搜索的人工蜂群算法[J]. 控制理论与应用, 2014, 31(11): 1502. DOI:10.7641/CTA.2014.31114
- KUANG Fangjun, XU Weihong, JIN Zhong. Artificial bee colony algorithm based on self-adaptive tent chaos search[J]. Control Theory & Applications, 2014, 31(11): 1502. DOI:10.7641/CTA.2014.31114
- [9] 薛永生, 吴立尧. 基于模拟退火的改进粒子群算法研究及应用[J]. 海军航空工程学院学报, 2018, 33(2): 248. DOI:10.7682/j.issn.1673–1522.2018.02.012
- XUE Yongsheng, WU Liyao. Research and application of improved PSO algorithm based on simulated annealing[J]. Journal of Naval Aeronautical and Astronautical University, 2018, 33(2): 248. DOI:10.7682/j.issn.1673–1522.2018.02.012
- [10] 李志明, 莫愿斌. 基于 Lévy 飞行的飞蛾扑火优化算法[J]. 计算机工程与设计, 2017, 38(3): 807. DOI:10.16028/j.issn1000–7024.2017.03.046
- LI Zhiming, MO Yuanbin. Moth-flame optimization algorithm based on Lévy flights[J]. Computer Engineering and Design, 2017, 38(3): 807. DOI:10.16028/j.issn1000–7024.2017.03.046
- [11] 单梁, 强浩, 李军, 等. 基于 Tent 映射的混沌优化算法[J]. 控制与决策, 2005, 20(2): 179. DOI:10.3321/j.issn:1001–0920.2005.02.013
- SHAN Liang, QIANG Hao, LI Jun, et al. Chaotic optimization algorithm based on tent map[J]. Control and Decision, 2005, 20(2): 179. DOI:10.3321/j.issn:1001–0920.2005.02.013
- [12] LIAO X, ZHOU J Z, OUYANG S, et al. An adaptive chaotic artificial bee colony algorithm for short-term hydrothermal generation scheduling[J]. Electrical Power and Energy Systems, 2013, 53(1): 34. DOI:10.1016/j.ijepes.2013.04.004
- [13] 向祖权, 靳超, 杜开君, 等. 基于粒子群优化算法的水面无人艇分层局部路径规划[J]. 武汉理工大学学报, 2015, 37(7): 38. DOI:10.3963/j.issn.1671–4431.2015.07.008
- XIANG Zuquan, JIN Chao, DU Kaijun, et al. Local obstacle avoidance for unmanned surface vehicle using a hierarchical strategy based on particle swarm optimization[J]. Journal of Wuhan University of Technology, 2015, 37(7): 38. DOI:10.3963/j.issn.1671–4431.2015.07.008
- [14] PAN T S, DAO T K, PAN J S, et al. An unmanned aerial vehicle optimal route planning based on compact artificial bee colony[C]// Advances in Intelligent Information Hiding and Multimedia Signal Processing (IHMS). Kaohsiung, Taiwan: Springer, 2016: 361. DOI:10.1007/978–3–319–50212–0_43
- [15] KENNEDY J, EBERHART R. Particle swarm optimization[C]// Proceedings of the IEEE International Conference on Neural Networks (ICNN). Piscataway, NJ: IEEE Press, 1995: 1942. DOI:10.1109/ICNN.1995.488968
- [16] GAO W F, LIU S Y. A modified artificial bee colony algorithm[J]. Computer & Operations Research, 2012, 39(3): 687. DOI: 10.1016/j.cor.2011.06.007

(编辑 苗秀芝)