DOI:10.11918/j.issn.0367-6234.201706044

自适应 Tent 混沌搜索的蚁狮优化算法

张振兴, 杨任农, 房育寰, 赵克新

(空军工程大学 航空航天工程学院, 西安 710038)

摘 要: 针对蚁狮算法存在的早熟收敛和不易得到全局最优解等问题,借鉴混沌优化算法,提出了自适应 Tent 混沌搜索蚁狮算法. 该算法首先使用 Tent 混沌映射初始化种群,然后自适应调整混沌搜索空间得到最优解,改善适应度较差个体,提高种群整体的适应度和寻优效率,同时使用锦标赛策略选择蚁狮个体. 最后,利用混沌算子优化蚂蚁随机游走行为,与蚁狮觅食行为形成了全局、局部并行搜索模式. 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 个复杂高维基准函数的寻优测试实验表明,对于 30 维基准函数,该算法经过约 0.5 秒收敛到最优值;对于 50 维基准函数,约 2 秒收敛到最优值. 与标准蚁狮算法和其他优化算法相比,该算法具有较好的收敛速度和寻优精度,适合复杂高维函数寻优. 航迹规划实验表明,对于包含 7 个威胁源的空域环境,当搜索维度为 10 维时,该算法经过 0.939 秒,迭代 30 次基本可以达到航迹代价的全局最优值. 与标准蚁狮算法相比,能够更加快速准确地得到一条满足要求的航迹,具有实际应用价值.

关键词: 蚁狮优化算法;混沌理论;锦标赛选择策略;Tent 映射;Logistic 映射;航迹规划

中图分类号: V247 **文献标志码**: A **文章编号**: 0367-6234(2018)05-0152-08

Ant lion optimizatopm algorithm based on self-adaptive Tent chaos search

ZHANG Zhenxing, YANG Rennong, FANG Yuhuan, ZHAO Kexin

(College of Aeronautics and Aeronautics Engineering, Air Force Engineering University, Xi'an 710038, China)

Abstract: To overcome premature convergence and local minimum of Ant Lion Optimizer (ALO) algorithm, an innovative ALO algorithm based on self-adaptive chaos search which is combined with Tent chaos algorithm is proposed. Tent mapping is applied to initialize individuals in the search space. Self-adaptive adjustment of chaos search scopes is presented to get the better solution, perfect the fitness of the worse individuals and enhance the whole individuals' fitness and efficiency. At the same time, Tournament selection strategy is used to screen out ant lion individuals. At the end, ants' random walking behavior is optimized by chaos operator to form a parallel search mode with ant lion foraging behavior, which is beneficial to the overall optimization performance to obtain the optima. Algorithm's performance is tested through complex benchmark functions with high-dimension and path planning problem. Experimental results of the former further confirm that, as for benchmark functions with 30 dimensions, it costs about 0.5 s to converge to the optimum, and for 50 dimensions, it is about 2 s. Compared with ALO and other optimization algorithms, the improved algorithm not only accelerates the convergence rate and improves the precision, but it is also fit for complex high-dimensional functions optimization. Path planning test manifests that for airspace with 7 menaces, while the search dimension is 10, after 0.939 s, and 30 iterations, the global optimum of the path cost function can be got. Compared with ALO, it has advantage in speed and accuracy to get the specific path, and it is of great value in actual problems.

Keywords: ant lion optimization algorithm; chaos theory; tournament selection strategy; Tent mapping; Logistic mapping; path planning

蚁狮优化算法(ant lion optimizer algorithm, ALO)作为一种新的智能算法,于 2015 年由澳大利亚教授 Seyedali 提出. 它的优势在于调节参数少,全局寻优能力好、收敛速度快和易于实现等方面. 与粒子群算法、蝙蝠算法和花朵授粉算法等 7 种智能算法相比,ALO 算法具有更好的全局寻优能力和收敛速度[1]. 但在寻优进化过程中,算法也存在一些不足,比如:算法初期的轮盘赌搜索方法影响种群的寻优性能和收敛速度;在迭代过程中,蚂蚁可能向适应度较差的个体进行学习,使算法陷入局部最优,影响算法效率;算法后期,蚁狮多样性随搜索步长和搜索速度的下降而降低,从而导致算法后期易陷入局部最优解. 针对上述问题,众多学者进行了研究. 罗钧等[2]结合 Logistic 混沌方法,改善蜂群的种群多

收稿日期: 2017-06-08
基金项目: 航空科学基金资助项目(20155196022);国家自然科学基金青年基金资助项目(71501184);陕西省自然科学基金(2016JQ6050)
作者简介: 张振兴(1993—),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,2207621676@qq.com

样性和全局寻优能力；Gao 等[3]使用 Logistic 混沌和反向训练方法提高收敛速度和精度；匡芳君等[4-5]使用 Tent 混沌方法优化蜂群与粒子群混合算法，克服后期不易获得最优解的缺点；Akay 等[6]提出利用参数优化改进人工蜂群算法.

针对 ALO 算法缺陷，本文借鉴 Tent 混沌方法[4]、锦标赛选择策略[7]和 Logistic 混沌映射[3]，提出了自适应 Tent 混沌搜索的蚁狮优化算法（self-adaptive Tent chaos search ant lion optimizer algorithm，SATC-ALO）. 算法初期，使用 Tent 混沌策略优化算法，扩大种群搜索区域，保证其均匀性和多样性，同时采取自适应动态调整 Tent 混沌搜索空间，产生适应度较好的新解，克服算法无法获得全局最优解的缺陷；使用锦标赛选择策略替代轮盘赌法选择蚁狮，提高种群的优良性能和跳出局部最优的能力；算法后期，将混沌算子与蚂蚁随机游走的行为结合，生成具有混沌算子遍历性、随机性和规律性特点的混沌蚂蚁[8]，结合蚁狮的寻优能力，降低其收敛时间. 实验仿真结果表明，该算法优化种群的多样性和均匀性，并能改善 ALO 算法的收敛时间、收敛精度、搜索效率以及跳出局部最优的能力，在 6 个基准测试函数和航迹规划问题上均表现出较好的性能，具有实际应用价值.

1　蚁狮优化算法

蚁狮优化算法是受到 AntLion 捕捉蚂蚁的过程的启发：蚂蚁在地上无规律爬行，而 AntLion 事先在地下做好陷阱等待蚂蚁，当蚂蚁被 AntLion 捉到后，AntLion 会继续挖陷阱等待下一只蚂蚁.

ALO 算法即是通过仿照蚁狮捕捉蚂蚁的过程来解决实际问题：蚂蚁通过随意行为对解进行搜索；通过学习优秀蚁狮以确保取得全局最优值. 蚂蚁和蚁狮的位置表示优化问题的解，蚁狮通过不断捕食蚂蚁寻找问题的最优解. 蚁狮捕食蚂蚁的行为见图 1 所示.

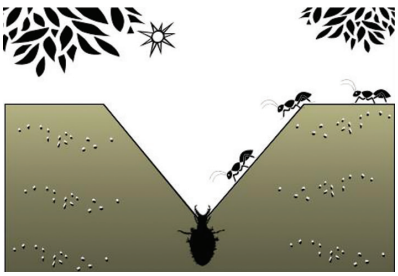

图 1　蚁狮的捕食行为

Fig.1　Hunting behaviour of antlions

寻求最优解时，需要应用以下条件：

1）蚂蚁随意走动受蚁狮陷阱的牵制；

2）蚁狮的适应度值越大，陷阱的范围越大，越容易抓到蚂蚁；

3）如果蚂蚁的适应度值优于蚁狮，表示被蚁狮抓到.

蚂蚁的随机行为表达式为

$$\boldsymbol{X}(t)=[0,\mathrm{cumsum}(2r(t_1)-1),\mathrm{cumsum}(2r(t_2)-1),\cdots,\mathrm{cumsum}(2r(t_n)-1)] \tag{1}$$

式中：cumsum 为计算数组累，n 为蚂蚁的数目，t 为目前的迭代次数，$r(t)$ 的表达式为

$$r(t)=\begin{cases}1, & \text{if } m>0.5;\\ 0, & \text{if } m\leqslant 0.5.\end{cases} \tag{2}$$

式中 m 为 0~1 之间的随机数.

蚂蚁的随机游走受到蚁狮的影响，数学表式为

$$\boldsymbol{c}_i^t=\boldsymbol{Y}_i^t+\boldsymbol{c}^t, \tag{3}$$

$$\boldsymbol{d}_i^t=\boldsymbol{Y}_i^t+\boldsymbol{d}^t. \tag{4}$$

式中：$\boldsymbol{c}^t$ 为所有变量第 t 次迭代的最小值，$\boldsymbol{d}^t$ 为所有变量第 t 次迭代的最大值，$\boldsymbol{Y}_i^t$ 为被选出的第 i 只蚁狮第 t 次迭代后的位置.

当蚂蚁在陷阱周围随机移动时，蚁狮会继续挖洞，使蚂蚁慢慢移向自己，数学表式为

$$\boldsymbol{c}^t=\frac{\boldsymbol{c}^t}{10^w\dfrac{t}{T_{\max}}}, \tag{5}$$

$$\boldsymbol{d}^t=\frac{\boldsymbol{d}^t}{10^w\dfrac{t}{T_{\max}}}. \tag{6}$$

式中：t 为当前迭代次数，$T_{\max}$ 为最大的迭代次数，w 为固定值，能够调整蚂蚁移向蚁狮的速度，数学表式为

$$w=\begin{cases}2, & t>0.1T;\\ 3, & t>0.5T;\\ 4, & t>0.75T;\\ 5, & t>0.9T;\\ 6, & t>0.95T.\end{cases} \tag{7}$$

为保证蚂蚁在搜索空间内随机游走，使用式(8)处理

$$\boldsymbol{X}_i^t=\frac{(\boldsymbol{X}_i^t-\boldsymbol{a}_i)\times(\boldsymbol{d}_i^t-\boldsymbol{c}_i^t)}{\boldsymbol{b}_i-\boldsymbol{a}_i}+\boldsymbol{c}_i^t. \tag{8}$$

式中：$\boldsymbol{a}_i$ 和 $\boldsymbol{b}_i$ 分别为第 i 维变量随意游走的最小值和最大值，$\boldsymbol{c}_i^t$ 和 $\boldsymbol{d}_i^t$ 分别为第 i 维变量第 t 次迭代的最小值和最大值.

蚂蚁当前的位置为

$$\boldsymbol{B}_i^t=\frac{\boldsymbol{A}^t+\boldsymbol{E}^t}{2}. \tag{9}$$

式中：$\boldsymbol{A}^t$ 为第 t 次迭代在通过轮盘赌方式选择的蚁狮周围随机游走，$\boldsymbol{E}^t$ 为第 t 次迭代在精英蚁狮周围随机游走.

蚁狮捕到 Ant 后,位置的数学表示

$$\boldsymbol{Y}_i^t = \boldsymbol{B}_i^t \text{ if } f(\boldsymbol{B}_i^t) > f(\boldsymbol{Y}_i^t). \tag{10}$$

ALO 算法步骤如下:

步骤 1 设置参数:最大迭代次数 $I_{\max}$、蚂蚁和蚁狮的数目 Num_{B}、Num_{Y}、适应度函数维数 dim 以及变量范围 u 和 l.

步骤 2 随机初始化蚂蚁和蚁狮的位置,计算相应的适应度值并排序,选出其中适应度值最大的蚁狮作为精英蚁狮个体 EAntlion, $I = 1$.

步骤 3 通过轮盘赌方式对蚁狮进行贪婪选择,利用式(3)和(4)更新 $\boldsymbol{c}_i^t$ 和 $\boldsymbol{d}_i^t$ 的值,实现蚂蚁在选择出的蚁狮和精英蚁狮周围随机游走,利用式(5)~(8)使蚂蚁不断靠近蚁狮,之后根据式(9)更新蚂蚁的位置.

步骤 4 游走后的蚂蚁根据式(10)与当前位置最好的蚁狮对比,重新调整最佳蚁狮的位置. 如果蚁狮的位置超出了最远边界 u 或者 l,则按照最远边界处理,$I = I + 1$.

步骤 5 判断是否达到算法的最大迭代次数或精度,若达到,则获得最佳蚁狮位置,否则转至步骤 3.

2 自适应 Tent 混沌搜索的蚁狮优化算法

2.1 Tent 混沌序列

Tent 混沌方法具备随机性、均匀性和有序性等优势,利用其寻求最优解,可提高种群多样性,取得全局最优解. 因此,本文使用 Tent 混沌序列初始化种群.

Tent 映射表达式如下:

$$\boldsymbol{x}_{t+1} = \begin{cases} 2\boldsymbol{x}_t, & 0 \leqslant \boldsymbol{x}_t \leqslant 0.5; \\ 2(1-\boldsymbol{x}_t), & 0.5 \leqslant \boldsymbol{x}_t \leqslant 1. \end{cases} \tag{11}$$

经过贝努力移位变换后为

$$\boldsymbol{x}_{t+1} = (2\boldsymbol{x}_t) \bmod 1. \tag{12}$$

在可行解中生成 Tent 混沌序列的方式为

步骤 1 产生(0,1)之间的随机数 x_0 作为初值(避免 x_0 在小周期内(0.2,0.4,0.6,0.8)),$z(1) = x_0$,$i = j = 1$;

步骤 2 根据式(12)迭代,生成 $\boldsymbol{x}(i+1)$,$i = i + 1$;

步骤 3 如果 $\boldsymbol{x}(i) = \{0, 0.25, 0.5, 0.75\}$,或 $\boldsymbol{x}(i) = \boldsymbol{x}(i-k)$,$k = \{0,1,2,3,4\}$,按式 $\boldsymbol{x}(i) = \boldsymbol{z}(j+1) = z(j) + \varepsilon$ 改变迭代初值,ε 为随机数,$j = j + 1$,否则转向步骤 2.

步骤 4 若到达最大迭代数,程序结束,保持 x 序列,否则转到步骤 2.

2.2 自适应 Tent 混沌搜索

使用自适应动态调整混沌搜索策略为适应度较差的蚂蚁和蚁狮个体产生适应度较好的新解,改善种群的整体性能. 即在每一次迭代中,分别求出种群第 j 维的最小值 $\boldsymbol{X}_{\min}^j$ 和最大值 $\boldsymbol{X}_{\max}^j$ 作为混沌搜索空间,以目前为止搜索到最优值为基础产生 Tent 混沌序列. 主要步骤如下:

步骤 1 $\boldsymbol{x}_k$ 归一化处理

$$\boldsymbol{z}_k^j(0) = \frac{(\boldsymbol{x}_k^j - \boldsymbol{X}_{\min}^j)}{(\boldsymbol{X}_{\max}^j - \boldsymbol{X}_{\min}^j)}. \tag{13}$$

式中 $k = 1, \cdots, n$, $j = 1, \cdots, D$.

步骤 2 利用式(12)生成混沌解 $\boldsymbol{z}_k^j(m)$, 同时利用式(14)使 $\boldsymbol{z}_k^j(m)$ 还原到解空间的邻域内

$$\boldsymbol{y}_k^j = \boldsymbol{x}_k^j + \frac{(\boldsymbol{X}_{\max}^j - \boldsymbol{X}_{\min}^j)}{2} \times (2\boldsymbol{z}_k^j(m) - 1). \tag{14}$$

步骤 3 计算新的适应度值和原值比对,保留更好的解.

步骤 4 如果到达最大迭代次数,程序终止,否则转向步骤 2.

2.3 锦标赛选择方法

基本 ALO 算法使用轮盘赌法将适应度值占总适应度值的比例作为选择概率选出被学习的蚁狮,使得蚂蚁向适应度高的蚁狮群体集中,破坏了种群多样性,使结果出现早熟收敛,得到局部极值. 锦标赛选择机制[7]是从局部竞争机制中产生的,通过随机选取 q 个个体进行比较,从中选取适应度值较大者作为最优个体. 本文采用锦标赛策略选取被学习蚁狮个体,并使 $q = 2$. 在每次选取的过程中,奖励适应度值大的 1 分,重复 n 次,其中得分最高的权重也最大. 由于锦标赛选择策略使用适应度的相对值作为选择标准,并且未对适应度值正负做要求,进而避免了超级个体对进化过程的影响. 适应度选择概率为

$$P_i = \frac{c_i(t)}{\sum_{i=1}^{N} c_i(t)}. \tag{15}$$

式中 c_i 为每个个体得分.

2.4 Logistic 混沌搜索

将典型的 Logistic 混沌算子与蚂蚁随意行为结合成混沌蚁群,使蚂蚁能够全局寻优,改善了算法取得全局最优值的性能. 混沌变量数学表达式为[9]

$$x_{n+1} = 4x_n(1 - x_n). \tag{16}$$

式中 x_n 是 0~1 之间的一个随机数.

经过改进,可将混沌蚂蚁的全局搜索能力和蚁狮的局部搜索能力结合,改善了算法的整体收敛时间和精度.

2.5 自适应 Tent 混沌搜索的蚁狮算法流程

算法的步骤如下:

步骤 1 设置参数:蚁狮算法的最大迭代次数

I_{max}、蚂蚁和蚁狮的数目 Num_B 和 Num_Y、适应度函数维数 dim、变量范围 u 和 l 以及混沌策略的最大迭代次数 m.

步骤 2　在搜索区域内，利用 Tent 混沌序列优化种群，分别生成 Num_B 和 Num_Y 个 dim 维向量 $\boldsymbol{X}_i$，I =1.

1.随机生成 Num_B 和 Num_Y 个(0,1)之间 dim 维向量 $\boldsymbol{Z}_i^D(t)$，根据 Tent 混沌序列可得相应的序列 $\boldsymbol{Z}_i^D(t)$.

2.利用式(17)将 $\boldsymbol{Z}_i^D(t)$ 载波到对应变量的取值范围内，即

$$\boldsymbol{X}_j^i(t) = \boldsymbol{X}_{\min}^j + (\boldsymbol{X}_{\max}^j - \boldsymbol{X}_{\min}^j)\, \boldsymbol{z}_i^j(t). \quad (17)$$

步骤 3　利用自适应 Tent 混沌搜索为种群中适应度较差蚂蚁和蚁狮产生新解.

1.设置参数：种群中较差个体比例 p_0，选出个数是 $p_0 \times$ Num，Tent 混沌搜索次数是 n.

2.自适应 Tent 动态调整混沌搜索空间方法为 $p_0 \times$ Num 个个体产生新解.

步骤 4　计算 $\boldsymbol{X}_i$ 适应度值并排序，从中选出值最大的个体作为精英个体 EAntlion.

步骤 5　通过锦标赛选择方式对蚁狮进行选择，并更新 $\boldsymbol{c}_i^t$ 和 $\boldsymbol{d}_i^t$ 的值.

步骤 6　将混沌算子与蚂蚁的随机游走相结合，利用式(3)和(4)实现蚂蚁在蚁狮周围混沌随机游走，利用式(5)~(8)使蚂蚁不断靠近蚁狮，之后根据式(9)更新蚂蚁的位置.

步骤 7　游走后的蚂蚁根据式(10)与当前位置最好的蚁狮对比，重新调整最佳蚁狮的位置. 如果蚁狮的位置超出了最远边界 u 或者 l，则按照最远边界处理，$I = I + 1$.

步骤 8　判断是否达到算法的最大迭代数，若达到，则输出最佳蚁狮位置，否则回到步骤 3.

3　仿真实验与结果分析

3.1　基准测试函数

为验证提出的 SATC-ALO 算法的特性，选取了 6 个 Benchmark 基准函数[10]和航迹规划问题对算法进行寻优测试. 其中，基准函数的不同性能可以充分检验算法的性能，见表 1 所示. 单峰函数主要检验其寻优性能和收敛速度，复杂非线性多峰函数主要检验算法的全局优化能力、跳出局部最优值等性能. 航迹规划问题可以检验 SATC-ALO 算法在具体问题上的实时性和准确性. 实验仿真环境为 Inter(R) Core(TM) i5-6500，CPU，3.20GHz，4GB RAM，Win7 操作系统，仿真软件为 MATLAB R2014a.

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	$f(\boldsymbol{X}) = \sum_{i=1}^{D} x_i^2,\ x_i \in [-100,100]$	$x_i = 0,\ f(\boldsymbol{X}) = 0$	单峰
Rosenbrock	$f(\boldsymbol{X}) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert,\ x_i \in [-10,10]$	$x_i = 0,\ f(\boldsymbol{X}) = 0$	单峰
Rastrigin	$f(\boldsymbol{X}) = \sum_{i=1}^{D} (100(x_i^2 - 10\cos(2\pi x_i) + 10)),\ x_i \in [-5.12, 5.12]$	$x_i = 0,\ f(\boldsymbol{X}) = 0$	多峰
Griewank	$f(\boldsymbol{X}) = \frac{1}{4\,000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\frac{x_i}{\sqrt{i}} + 1,\ x_i \in [-600, 600]$	$x_i = 0,\ f(\boldsymbol{X}) = 0$	多峰
Ackley	$f(\boldsymbol{X}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e,\ x_i \in [-32, 32]$	$x_i = 0,\ f(\boldsymbol{X}) = 0$	多峰
Schwefel	$f(\boldsymbol{X}) = -\sum_{i=1}^{D} (x_i \sin\sqrt{\lvert x_i \rvert}),\ x_i \in [-500, 500]$	$x_i = 420.968\,7,\ f(\boldsymbol{X}) = -418.982\,9$	多峰

3.2　SATC-ALO 算法的性能分析

3.2.1　基准函数寻优性能分析

为了评价算法效用性，采用 ALO 和 SATC-ALO 算法进行测试. 实验参数设置：蚂蚁和蚁狮的个数为 100 个，最大迭代次数为1 000 次，种群中较差个体比例 p_0 为 0.3，混沌搜索次数为 30 次. 通过对比每个测试函数运行 50 次得到的适应度平均值、方差等参数来评价各个算法的优化能力. 30 维基准测试函数进化曲线见图 2~7 所示，30 维和 50 维的测试结果见表 2 所示.

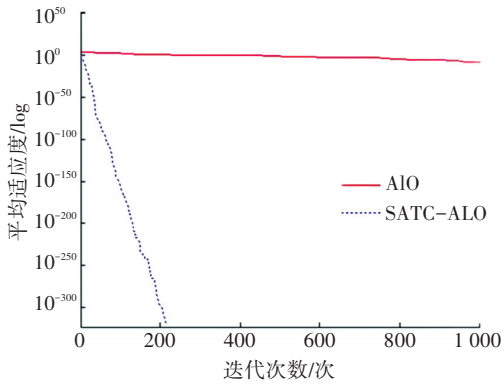

图 2 Sphere 函数动态寻优过程(30 维)

Fig.2 The progress curve of Sphere function (30 dimensions)

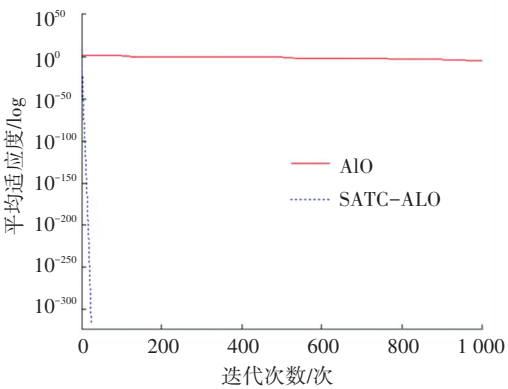

图 3 Rosenbrock 函数动态寻优过程(30 维)

Fig.3 The progress curve of Rosenbrock function (30 dimensions)

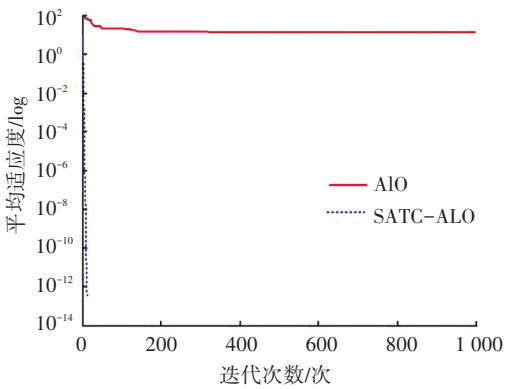

图 4 Rastrigin 函数动态寻优过程(30 维)

Fig.4 The progress curve of Rastrigin function (30 dimensions)

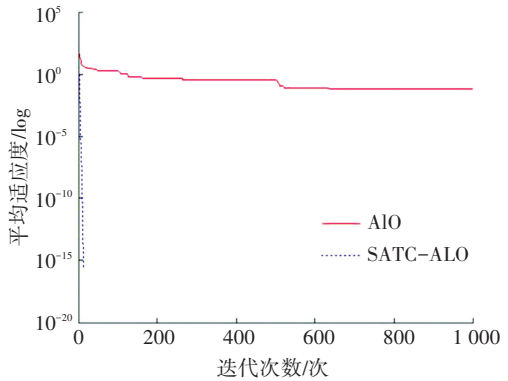

图 5 Griewank 函数动态寻优过程(30 维)

Fig.5 The progress curve of Griewank function (30 dimensions)

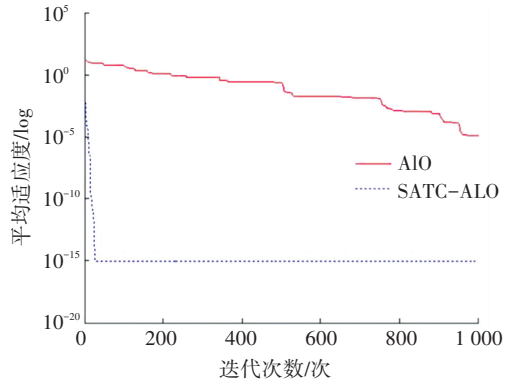

图 6 Ackley 函数动态寻优过程(30 维)

Fig.6 The progress curve of Ackley function (30 dimensions)

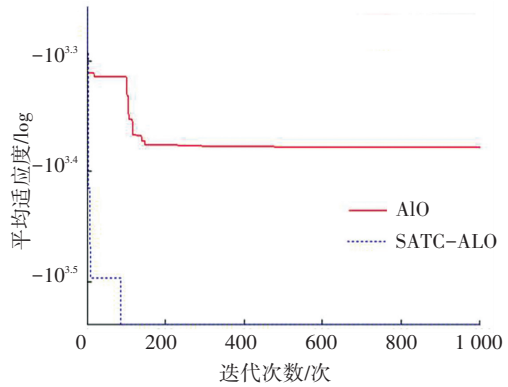

图 7 Schwefel 函数动态寻优过程(30 维)

Fig.7 The progress curve of Schwefel function (30 dimensions)

由图 2~6 和表 2 可知,无论是针对单峰函数还是针对多峰函数,SATC-ALO 算法的收敛时间均明显优于 ALO 算法,全局寻优效果好,可以连续、高效地搜寻到函数最小值. 由图 7 和表 2 可知,对多峰函数 Schwefel,取值范围为[-500,500],最优值在搜索空间边界附近,使算法极易陷入到某一个局部最优解之中,但 SATC-ALO 算法收敛值接近理论最优值,并且寻优速度也明显优于 ALO 算法.

进一步验证 SATC-ALO 算法的优越性,将其与智能算法中寻优能力较好的人工蜂群算法(ABC)[11],粒子群算法(PSO)[12],灰狼算法(GWO)[13] 进行了比较,选择 Sphere 函数和 Griewank 函数作为测试函数,设置函数维度为 50,最大迭代次数为 300 次. 4 种算法对两种测试函数寻优的进化过程见图 8~9 所示. 基准函数的搜索区域见图 10 所示.

由图 8~9 和表 2 可知,对于 Sphere 函数和 Griewank 函数,SATC-ALO 算法的收敛精度和时间均明显好于其他三种智能算法.

综上所述,在收敛速度和寻优精度上,SATC-ALO 算法较 ALO 算法和大部分智能算法均有明显的提升,具有较好的全局优化和跳出局部最优的能力,并且鲁棒性较好.

表 2　基准函数的优化结果比较

Tab.2　Optimization results comparison of benchmark functions

函数	维数	算法	均值	方差	最大值	最小值	收敛时间/s
f_1	50	SATC-ALO	0	0	0	0	1.49
		ALO	2.84×10^{-10}	1.28×10^{-10}	2.95×10^{-10}	2.68×10^{-10}	180.95
	30	SATC-ALO	0	0	0	0	0.57
		ALO	7.46×10^{-13}	2.79×10^{-12}	8.32×10^{-13}	6.58×10^{-13}	64.51
f_2	50	SATC-ALO	0	0	0	0	0.12
		ALO	5.16×10^{-6}	8.43×10^{-7}	6.35×10^{-6}	4.02×10^{-6}	183.85
	30	SATC-ALO	0	0	0	0	0.10
		ALO	6.92×10^{-7}	2.68×10^{-7}	8.34×10^{-7}	5.31×10^{-7}	64.42
f_3	50	SATC-ALO	0	0	0	0	1.70
		ALO	7.26×10^{-7}	5.34×10^{-7}	4.19×10^{-7}	1.28×10^{-7}	156.57
	30	SATC-ALO	0	0	0	0	0.61
		ALO	6.92×10^{-7}	2.68×10^{-7}	8.34×10^{-7}	5.31×10^{-7}	64.42
f_4	50	SATC-ALO	0	0	0	0	1.42
		ALO	1.86×10^{-2}	6.52×10^{-2}	2.25×10^{-2}	0.87×10^{-2}	181.09
	30	SATC-ALO	0	0	0	0	0.55
		ALO	8.49×10^{-3}	2.64×10^{-4}	9.64×10^{-3}	7.17×10^{-3}	54.21
f_5	50	SATC-ALO	8.88×10^{-16}	8.59×10^{-17}	7.83×10^{-16}	7.21×10^{-16}	3.27
		ALO	7.34×10^{-9}	5.94×10^{-10}	4.84×10^{-9}	6.29×10^{-9}	179.83
	30	SATC-ALO	0	0	0	0	1.32
		ALO	5.71×10^{-7}	2.48×10^{-8}	8.42×10^{-7}	6.27×10^{-7}	63.25
f_6	50	SATC-ALO	-2.09×10^{4}	3.85×10^{-9}	-2.09×10^{4}	-2.09×10^{4}	2.03
		ALO	-2.07×10^{4}	1.33×10^{1}	-2.02×10^{4}	-2.08×10^{4}	181.00
	30	SATC-ALO	-3.44×10^{3}	8.49×10^{-8}	-2.86×10^{3}	-3.50×10^{3}	3.33
		ALO	-2.39×10^{3}	9.21×10^{-6}	-2.24×10^{3}	-2.55×10^{3}	68.24

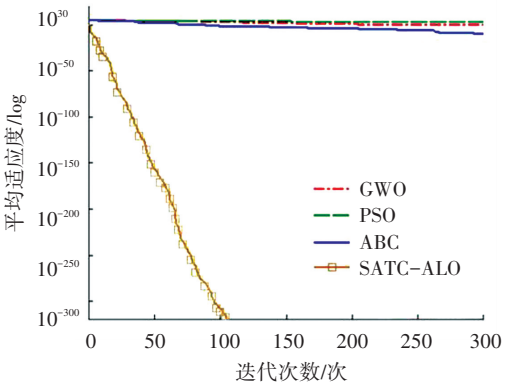

图 8　Sphere 函数动态寻优过程(50 维)

Fig.8　The progress curve of Sphere function (50 dimensions)

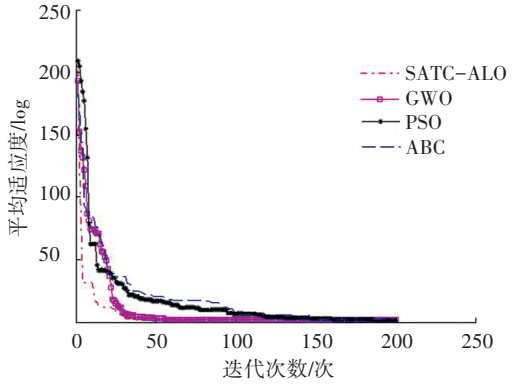

图 9　Griewank 函数动态寻优过程(50 维)

Fig.9　The progress curve of Griewank function (50 dimensions)

3.2.2　航迹规划问题性能分析

作为一种新型高效的群智能算法，SATC-ALO 算法可以应用在许多工程领域，比如航迹规划、火力分配问题以及其他寻优问题. 以无人机航行轨迹规划问题为例介绍 SATC-ALO 算法的具体用法并测试其应用性能.

航迹规划问题主要考虑油耗和威胁指标，表达式为

$$J_t = \int_0^L w_t \mathrm{d}l. \tag{18}$$

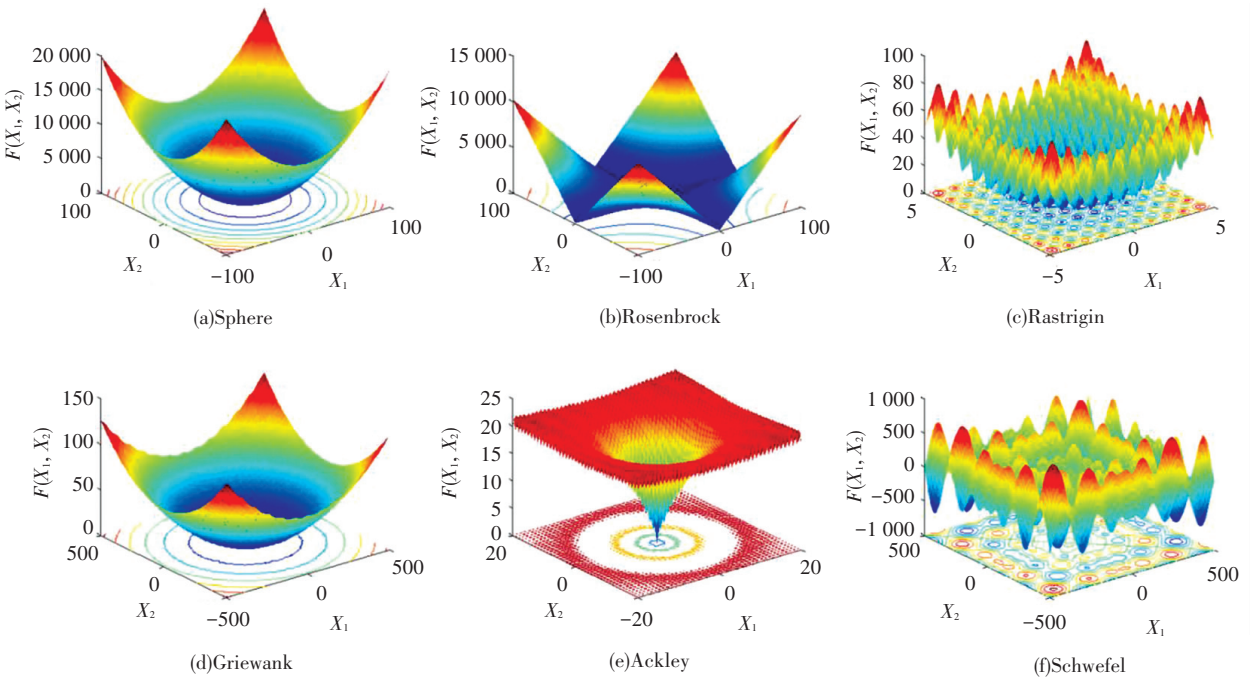

图 10 基准函数的搜索区域

Fig.10 Search space of benchmark functions

$$J_f = \int_0^L w_f \mathrm{d}l. \tag{19}$$

式中：w_t 和 w_f 分别为航迹点的威胁代价和油耗代价，L 为整段航迹的长度. 将整段航迹平均分成 10 段，计算奇数分点的评价指标. 因此，威胁代价为

$$w_{t,L_i} = \frac{L_i}{5} \cdot \sum_{k=1}^{N} t_k \cdot \left(\frac{1}{d_{0.1,i,k}^4} + \frac{1}{d_{0.3,i,k}^4} + \frac{1}{d_{0.5,i,k}^4} + \frac{1}{d_{0.7,i,k}^4} + \frac{1}{d_{0.9,i,k}^4}\right) \tag{20}$$

式中：N 为威胁源的数目；t_k 为常数，由具体威胁源决定；L_i 为第 i 段航迹的长度；$d_{0.1,i,k}^4$ 为 1/10 点与威胁源 k 之间的距离. 总的航迹代价为[8]

$$J = kJ_t + (1-k)J_f. \tag{21}$$

式中 k 为威胁权值，根据具体情况取值.

SATC-ALO 解决航迹规划问题的流程图见图 11.

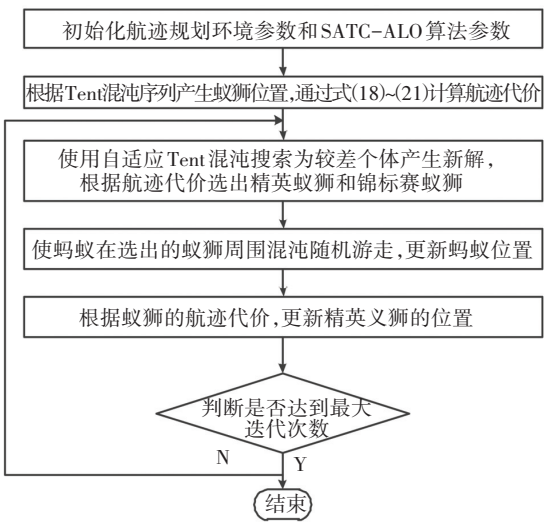

图 11 航迹规划算法流程图

Fig.11 The flow diagram of path planning

仿真实验：设置无人机的起点坐标为(10 km, 10 km)，目标点坐标为(100 km, 75 km)，具体的威胁源分布见表 3 所示. 设置种群规模为 15，$T_{\max}$ 为 150，D 为 10. SATC-ALO算法得到的航迹仿真结果见图 12、13.

仿真结果表明，SATC-ALO 算法经过 0.939 秒，迭代 30 次基本可以达到航迹代价最优值，满足实时性条件. 由图 12 和图 13 可知，在航迹规划问题上，SATC-ALO 算法的收敛速度和精度均优于 ALO 算法，可快速准确规划出一条满足要求的航迹，具有实际应用价值.

表 3 威胁源信息

Tab.3 Information of threat source

威胁源位置/km	威胁半径/km
[20,20]	8
[28,40]	15
[51,27]	8
[52,42]	17
[70,65]	10
[80,50]	10
[90,17]	10

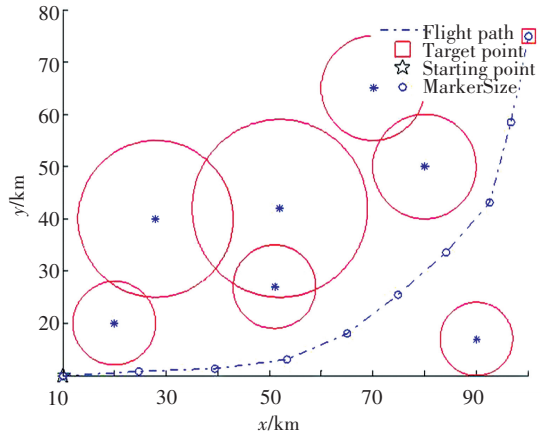

图 12　SATC-ALO 算法规划的航迹

Fig.12　Path planning result optimized by SATC-ALO algorithm

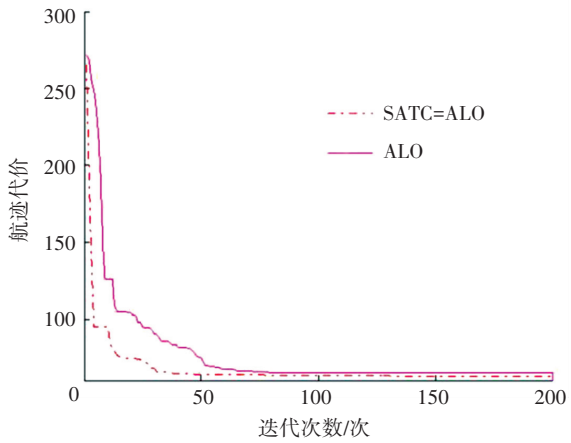

图 13　航迹代价收敛曲线

Fig.13　Convergence curves of algorithms

4　结　论

本文提出了自适应 Tent 混沌搜索的蚁狮优化算法(SATC-ALO). 主要从如下 4 个方面对 ALO 算法进行了优化:1)引入 Tent 混沌搜索优化种群,获得遍历均匀的个体,提升其多样性;2)自适应动态调整 Tent 混沌搜索空间,将混沌算子与蚂蚁的随机游走结合,使蚂蚁能够进行混沌搜索,提高算法跳出局部最优解的能力;3)利用 Tent 混沌搜索为种群中适应度较差蚂蚁和蚁狮产生新解,避免算法陷入局部最优值;4)使用锦标赛选择策略取代轮盘赌方式选择蚁狮,较为有效地规避了算法早熟收敛以及出现超级个体的现象.

对 6 个标准函数和航迹规划问题进行寻优测试. 实验结果表明,SATC-ALO 算法在保证种群均匀性和多样性的前提下,能够尽量避免早熟和陷入局部最优问题,具有优秀的收敛速度和寻优精度,并且算法相对稳定,鲁棒性较好.

参考文献

[1] MIRJALILI S. The ant lion optimizer [J]. Advances in Engineering Software, 2015, 83(C): 80-98. DOI:10.1016/ j.advengsoft.2015.01.010.

[2] 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- 1916. DOI: 10.13195/j.cd.2010.12.156.luoj.013.

[3] GAO W F, LIU S Y. A modified artificial bee colony algorithm [J]. Computer & Operations Research, 2012, 39(3): 687-697.

[4] 匡芳君, 金忠, 徐蔚鸿, 等. Tent 混沌人工蜂群与粒子群混合算法[J]. 控制与决策, 2015, 30(5): 839-846. DOI:10.13195/j.kzyjc.2014.0750.
KUANG Fangjun, JIN Zhong, XU Weihong, et al. Artificial bee colony algorithm based on self-adaptive tent chaos search [J]. Control Theory & Applications, 2014, 31(11): 1502-1508. DOI: 10.13195/j.kzyjc.2014.0750.

[5] 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): 1502-1509. DOI: 10.7641/CTA.2014.31114.
KUANG Fangjun, XU Weihong, JIN Zhong. Hybridization algorithm of Tent chaos artificial bee colony and particle swarm optimization [J]. Control and Decision, 2015, 30(5): 839-847. DOI: 10.7641/CTA.2014.31114.

[6] AKAY B, KARABOGA D. A modified artificial bee colony algorithm for real-parameter optimization [J]. Information Sciences, 2012, 192(6): 120-142. DOI: 10.1016/j.ins.2010.07. 015.

[7] BAO L, ZENG J C. Comparison and analysis of the selection mechanism in the artificial bee colony algorithm [C]//2009 9th International Conference on Hybrid Intelligent Systems. Los Alamitos, CA: IEEE, 2009: 411-416. DOI:10.1109/his.2009.319.

[8] XU C F, DUAN H B, LIU F. Chaotic artificial bee colony approach to uninhabited combat air vehicle (UCAV) path planning [J]. Aerospace Science and Technology, 2010, 14(8): 535-541. DOI:10.1016/j.ast.2010.04.008.

[9] ALATAS B. Chaotic bee colony algorithms for global numerical optimization [J]. Expert Systems with Applications, 2010, 37(8): 5682-5687. DOI:10.1016/j.eswa.2010.02.042.

[10] 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(12): 34-42. DOI:10.1016/j.ijepes. 2013.04.004.

[11] 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: 361-369. DOI:10.1007/978-3-319-50212-0_43.

[12] KENNEDY J, EBERHART R. Particle swarm optimization [C]// Proceedings of the IEEE International Conference on Neural Networks(ICNN). Piscataway, NJ: IEEE Press, 1995: 1942-1948.

[13] MIRJALILI S, MIRJALILI SM, LEWIS A. Grey wolf optimizer [J]. Advances in Engineering Software, 2014, 69: 46-61. DOI: 10.1016/j.ins.2014.09. 011.

(编辑　苗秀芝)