高思慧,吴克晴,何斌.改进搜索机制的自适应*t*分布麻雀搜索算法[J].辽宁工程技术大学学报(自然科学版),2025,44(2):247-256.DOI:10.11956/j.issn.1008-0562.20240087
GAO Sihui,WU Keqing,HE Bin.Adaptive *t*-distribution sparrow search algorithm with improved search mechanism[J].Journal of Liaoning Technical University(Natural Science),2025,44(2):247-256.DOI:10.11956/j.issn.1008-0562.20240087

改进搜索机制的自适应*t*分布麻雀搜索算法

高思慧，吴克晴*，何　斌
（江西理工大学 理学院,江西 赣州 341000）

摘　要：针对麻雀搜索算法在寻优过程中容易陷入局部最优、依赖种群初始化等缺陷，提出一种改进搜索机制的自适应*t*分布麻雀搜索算法（ATSSA）。引入Bernoulli混沌映射来获得高质量的初始种群；受鱼鹰优化算法中鱼鹰捕鱼方式的启发，改进发现者搜索机制，使发现者在寻优过程中表现出更大的灵活性，从而增强算法的勘探能力；根据概率引入自适应*t*分布算子进行扰动，以提升算法的收敛速度；采用黄金正弦策略来改变警觉者位置，提高算法的收敛能力。在14个基准函数上进行测试并进行Wilcoxon秩和检验来验证算法的性能。研究结果表明，ATSSA具有良好的寻优效果和鲁棒性。

关键词：麻雀搜索算法；Bernoulli混沌映射；鱼鹰优化算法；自适应*t*分布；黄金正弦算法
中图分类号：TP301.6　　**文献标志码：**A　　**文章编号：**1008-0562(2025)02-0247-10

Adaptive *t*-distribution sparrow search algorithm with improved search mechanism

GAO Sihui, WU Keqing*, HE Bin
(School of Science, Jiangxi University of Science and Technology, Ganzhou 341000, China)

Abstract: A self-adaptive *t*-distribution sparrow search algorithm with improved search mechanism (ATSSA) is proposed to address the shortcomings of the sparrow search algorithm, which is prone to falling into local optima and relies on population initialization during the optimization process. Bernoulli chaotic mapping is introduced to obtain a high-quality initial population; inspired by the fishing method of the osprey optimization algorithm, the discoverer search mechanism is improved to enable the discoverer to exhibit greater flexibility in the optimization process, thereby enhancing the exploration ability of the algorithm; an adaptive *t*-distribution operator is introduced based on probability to perturb and improve the convergence speed of the algorithm; the golden sine strategy is adopted to change the position of the alerter and improve the convergence ability of the algorithm. The performance of the algorithm was validated through testing on 14 benchmark functions and Wilcoxon rank sum test. The research results show that ATSSA has good optimization performance and robustness.

Keywords: sparrow search algorithm (SSA); Bernoulli chaotic mapping; osprey optimization algorithm; adaptive *t*-distribution; golden sine algorithm

0　引言

针对实际优化问题，牛顿法[1]、共轭梯度法[2]等传统优化方法在解决复杂问题、高维非线性问题时存在局限性，容易陷入局部最优[3]。因此，学者们采用元启发式优化算法来寻找问题的最优解[4]。其中，群智能算法以其简单、灵活、高效的特点得到广泛应用，如蜻蜓优化算法[5]、海鸥优化算法[6]、蜣螂优化算法[7]等。麻雀搜索算法（sparrow search algorithm，SSA）[8]是一种新型群智能算法，灵感来源于麻雀的捕食和反捕食行为。与其他算法相比，该算法具有结构简单、调整参数少以及搜索性能较好的特点，并在多个领域得到应用，如旅行商问题[9]、路径规划[10]、图像分割[11]等。

麻雀算法存在初始种群分布不均匀、收敛精度低以及易陷入局部最优的缺点[12]。因此，很多学者提出了改进方法。TANG等[13]利用鸟群算法的思想来更新发现者和加入者位置，然后引入交叉变

收稿日期：2024-04-10　**修回日期：**2024-05-24　**接受日期：**2024-06-13　**责任编辑：**张雯
基金项目：国家自然科学基金项目（12461010）
作者简介：高思慧（2000-），女，江西 吉安人，硕士研究生，主要从事智能优化、计算机应用技术方面的研究。E-mail：2893462499@qq.com
通信作者：吴克晴（1972-），男，江西 鹰潭人，博士，副教授，主要从事智能优化、泛函分析、运筹学等方面的研究。E-mail：wkq622@126.com

异算子进行扰动，降低了算法陷入局部最优的概率。CHEN 等[14]采用基于阶跃函数的混合搜索策略和基于 Logistic 模型的食物搜索策略，分别更新发现者和加入者位置，提高了算法的收敛速度。马卫等[15]采用 Sin 混沌映射初始化种群，随后引入 Levy 飞行来增强 SSA 的寻优性能。宋立钦等[16]利用 Circle 映射初始化种群，结合樽海鞘群算法更新发现者位置，并使用镜像选择提升麻雀个体质量，提高了算法的寻优精度和寻优速度。

已有的改进算法虽然在一定程度上提升了算法性能，但部分研究仅从单一角度对算法进行优化，比如只关注提高全局搜索能力，而忽略了局部搜索精度的提升，未能充分平衡全局与局部搜索，使得算法在迭代初期收敛较快，很快陷入局部最优，难以在复杂的搜索空间中持续寻优，导致最终结果精度受限。为进一步提升麻雀算法的性能，本文在已有研究的基础上提出一种改进搜索机制的自适应 t 分布麻雀搜索算法。首先，在初始化阶段引入 Bernoulli 混沌映射，使种群分布更加均匀；其次，结合鱼鹰优化算法来改变发现者的搜索机制，增加算法的全局搜索能力；再次，在加入者位置更新阶段引入自适应 t 分布算子，使算法的全局搜索能力与局部开发能力达到平衡；最后，采用黄金正弦策略更新警觉者位置，避免陷入局部最优。通过与其他 7 种算法在 14 个基准测试函数上进行测试并进行 Wilcoxon 秩和检验，验证改进算法的有效性。

1 麻雀搜索算法

在捕食过程中，麻雀个体分为发现者、加入者和警觉者。发现者具有较高的能量储备，引导种群中其余个体获取食物。当发现者寻找到食物后，加入者会与发现者抢夺食物。当种群面临威胁或者意识到危险时，警觉者会发出警报。

在 SSA 中，假设在一个 d 维搜索空间中，存在 n 只麻雀，则第 i 只麻雀在 d 维空间的位置为 $\boldsymbol{X}_i=\left(x_{i,1},x_{i,2},\cdots,x_{i,d}\right)$，其中，$i=1,2,\cdots,n$。麻雀种群的适应度为 $\boldsymbol{F}=\left(f(\boldsymbol{X}_1),f(\boldsymbol{X}_2),\cdots,f(\boldsymbol{X}_n)\right)$。

根据麻雀的适应度值来判断麻雀在种群中充当的角色，一般选择适应度值较小的前 10% ~ 20% 作为种群的发现者，其位置更新计算式为

$$\boldsymbol{X}_{i,j}^{I+1}=\begin{cases}\boldsymbol{X}_{i,j}^{I}\exp\left(-\dfrac{i}{\alpha M}\right) & R_2<S\\ \boldsymbol{X}_{i,j}^{I}+Q\boldsymbol{L} & R_2\geqslant S\end{cases},\tag{1}$$

式中：I 为当前迭代次数；$\boldsymbol{X}_{i,j}^{I}$ 为第 I 次迭代时，第 i 只麻雀在第 j 维中的位置，$j=1,2,\cdots,d$；M 为最大迭代次数；α 为随机数，取值范围为(0,1)；R_2 为预警值，$R_2\in[0,1]$；S 为安全值，$S\in[0.5,1.0]$；Q 为服从正态分布的随机数；$\boldsymbol{L}$ 为所有元素值为 1 的 $1\times d$ 矩阵。

当 $R_2<S$ 时，周围是安全的，发现者开始进入广泛搜索模式；当 $R_2\geqslant S$ 时，警觉者发现捕食者并发出警报，需要前往其他安全区域进行觅食。

在麻雀种群中，除生产者之外的麻雀都为加入者，其位置更新为

$$\boldsymbol{X}_{i,j}^{I+1}=\begin{cases}Q\exp\left(\dfrac{\boldsymbol{X}_{\text{worst}}^{I}-\boldsymbol{X}_{i,j}^{I}}{i^2}\right) & i>\dfrac{n}{2}\\ \boldsymbol{X}_{\text{P}}^{I+1}+\left|\boldsymbol{X}_{i,j}^{I}-\boldsymbol{X}_{\text{P}}^{I+1}\right|\cdot\boldsymbol{A}^{+}\cdot\boldsymbol{L} & \text{其他}\end{cases},\tag{2}$$

式中：$\boldsymbol{X}_{\text{worst}}^{I}$ 为第 I 次迭代时全局最差位置；$\boldsymbol{X}_{\text{P}}^{I+1}$ 为第 $I+1$ 次迭代时适应度值最优的发现者所处位置；$\boldsymbol{A}$ 为 $1\times d$ 矩阵，矩阵中的元素随机赋值 1 或−1，并且 $\boldsymbol{A}^{+}=\boldsymbol{A}^{\text{T}}\left(\boldsymbol{A}\boldsymbol{A}^{\text{T}}\right)^{-1}$。

当 $i>n/2$ 时，加入者没有得到食物，需要飞到别的地方觅食，以获得更多的能量；当 $i\leqslant n/2$ 时，加入者在发现者当前最佳位置 $\boldsymbol{X}_{\text{P}}$ 附近进行觅食。

种群在进行觅食时，会随机选取 10%～20% 的麻雀负责警戒，当发现周围有捕食者时，警觉者会发出警报，其位置更新为

$$\boldsymbol{X}_{i,j}^{I+1}=\begin{cases}\boldsymbol{X}_{\text{best}}^{I}+\beta\cdot\left|\boldsymbol{X}_{i,j}^{I}-\boldsymbol{X}_{\text{best}}^{I}\right| & f_i>f_{\text{b}}\\ \boldsymbol{X}_{i,j}^{I}+K\cdot\left(\dfrac{\left|\boldsymbol{X}_{i,j}^{I}-\boldsymbol{X}_{\text{worst}}^{I}\right|}{(f_i-f_{\text{w}})+\varepsilon}\right) & f_i=f_{\text{b}}\end{cases},\tag{3}$$

式中：$\boldsymbol{X}_{\text{best}}^{I}$ 为第 I 次迭代时全局最优位置；β 为控制步长的参数，是服从标准正态分布的随机数；f_i 为第 i 只麻雀的适应度；f_{b} 和 f_{w} 分别为当前全局最优和最差适应度；随机数 $K\in[-1,1]$，表示麻雀移动的方向，也可以控制移动的步长；ε 是不为 0 的常数，以防止分母为 0。

2 改进的麻雀搜索算法

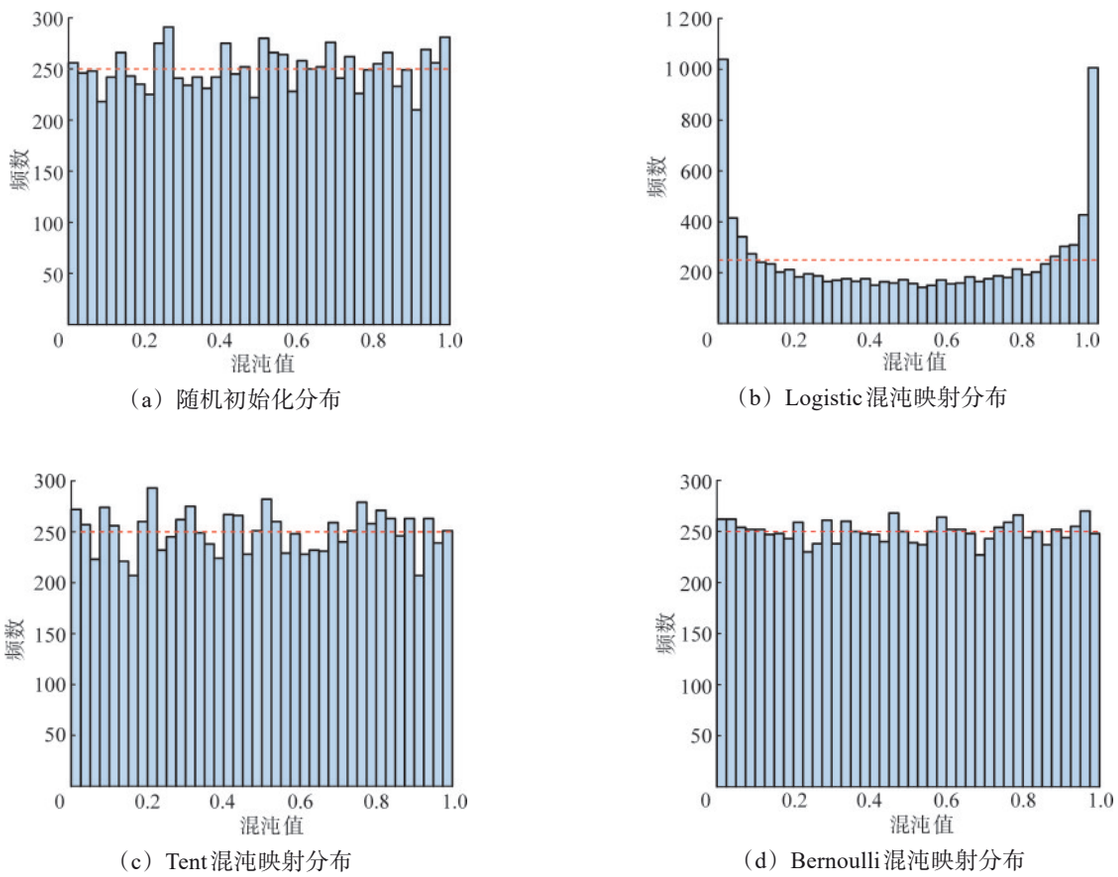
2.1 Bernoulli 混沌映射初始化种群

种群初始化对群智能算法的搜索精度有很大影响。原始的 SSA 中，采用的是随机初始化，这使得初始种群质量不稳定、多样性差，而混沌映射可以有效改善该问题。混沌是由简单的确定性系统产生的随机序列，具有遍历性和半随机性，有助于算法在搜索过程中避免陷入局部最优，在智能优化算法中得到广泛应用。

常用的混沌映射有 Logistic 混沌映射、Tent 混沌映射、Bernoulli 混沌映射等。使用这 3 种映射初始化种群及随机初始化种群的效果见图 1。由图 1 可以看出，Logistic 映射在(0,1)内服从切比雪夫分布[17]，呈现出中间低，两边高的特点，这种不均匀的分布会降低算法的效率。Tent 映射在(0,1)内的分布相对均匀，但存在小周期和不稳定的周期点。相较于 Logistic 映射和 Tent 映射，Bernoulli 映射取值分布更加均匀、稳定。因此，为了提高麻雀初始种群的质量，采用 Bernoulli 映射来替代随机数初始化麻雀个体位置，先利用 Bernoulli 映射关系将所得的值投影到混沌变量空间内，然后将产生的混沌值通过线性变换映射到算法初始空间中，Bernoulli 混沌映射表达式为

$$z_{k+1}=\begin{cases}\dfrac{z_k}{1-\rho} & z_k\in\left(0,1-\rho\right]\\ \dfrac{\left(z_k-1+\rho\right)}{\rho} & z_k\in\left(1-\rho,1\right)\end{cases},\tag{4}$$

式中：z_k 为产生的第 k 代混沌序列的值；ρ 为映射参数，$\rho\in(0,1)$，当 ρ 在 0.5 附近时能够获得较好的遍历性。



（a）随机初始化分布　（b）Logistic 混沌映射分布

（c）Tent 混沌映射分布　（d）Bernoulli 混沌映射分布

图 1　种群初始化效果

Fig.1　population initialization effect

2.2　融合鱼鹰优化算法的搜索机制

鱼鹰优化算法[18]（osprey optimization algorithm，OOA）在捕鱼阶段利用其强大的视力来探测水下鱼类的位置，并朝着比当前位置更优的区域移动，拥有较强的全局搜索能力，其位置更新为

$$\boldsymbol{O}_{i,j}^{\mathrm{P1}}=\boldsymbol{O}_{i,j}+r\left(\boldsymbol{E}_{i,j}-N\boldsymbol{O}_{i,j}\right),\tag{5}$$

式中：$\boldsymbol{O}_{i,j}^{\mathrm{P1}}$ 为第 i 只鱼鹰在第 j 维的新位置；$\boldsymbol{O}_{i,j}$ 为第 i 只鱼鹰在第 j 维的原始位置；r 为[0,1]的随机数，$\boldsymbol{E}_{i,j}$ 是第 i 只鱼鹰所选的鱼的位置；N 为集合{0,1}中的随机数。

在原始的 SSA 中，当 $R_2<S$ 时，随着迭代次数增加，发现者的搜索范围逐渐减小，搜索能力逐渐降低，导致全局搜索能力不足。为了提高麻雀搜索食物的效率，受鱼鹰优化算法捕鱼策略的启发，对发现者位置的搜索机制进行改进，避免其

过分依赖于上一代麻雀的位置，改进后的加入者位置更新为

$$\boldsymbol{X}_{i,j}^{I+1}=\begin{cases}\boldsymbol{X}_{i,j}^{I}+r\left(\boldsymbol{E}_{i,j}-N\boldsymbol{X}_{i,j}^{I}\right) & R_2<S\\ \boldsymbol{X}_{i,j}^{I}+\boldsymbol{QL}, & R_2\geqslant S\end{cases}。\tag{6}$$

改进后的麻雀个体在迭代时可以与其他个体进行信息交流，并实时判断是否存在比自身更优的解，若发现有更优解，便会迅速定位拥有该更优解的麻雀个体，并朝着这个方向靠近，以此调整自身位置。这种方式有效增强了麻雀个体间的信息流通，改变了以往信息交流匮乏的状况，推动整个算法朝着更优的方向迭代，同时也降低了算法陷入局部最优的可能性。

2.3 引入自适应 t 分布算子的加入者位置更新算法

t 分布[19]的概率密度函数为

$$Y(h)=\frac{\Gamma\left(\dfrac{v+1}{2}\right)}{\sqrt{v\pi}\,\Gamma\left(\dfrac{v}{2}\right)}\left(1+\frac{h^2}{2}\right)^{-\frac{v+1}{2}},\tag{7}$$

式中：v 为参数自由度；$\Gamma\left(\dfrac{v+1}{2}\right)$ 为第二型欧拉积分，当 $v=1$ 时，t 分布类似于柯西分布，当 v 趋于无穷大时，t 分布则趋近于高斯分布。

柯西分布、t 分布以及高斯分布的概率密度见图 2。t 分布的两端比较平坦，但峰值与柯西分布、高斯分布相差不大，可见 t 分布的扰动能力更强。

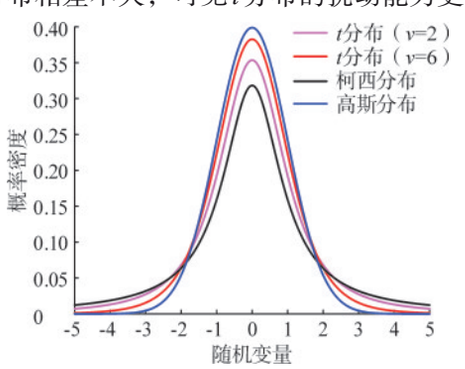


图 2 柯西分布、t 分布以及高斯分布的概率密度

Fig.2 probability density of Cauchy distribution, t-distribution, and Gaussian distribution

在原始的 SSA 中，当 $i>n/2$ 时，与发现者竞争食物失败的加入者会主动飞向其他区域觅食。从位置更新公式可以看出，其位置逐渐接近原点，这会导致搜索范围不足而陷入局部最优。因此，将迭代次数 I 作为 t 分布的变异算子，对个体位置进行扰动，使算法在迭代初期展现出较强的全局搜索能力，在迭代后期具备良好的局部开发能力，从而加快算法的收敛速度。引入变异算子后的加入者位置更新为

$$\boldsymbol{X}_{i,j}^{I+1'}=\boldsymbol{X}_{i,j}^{I}+t(I)\cdot\boldsymbol{X}_{i,j}^{I},\tag{8}$$

式中：$\boldsymbol{X}_{i,j}^{I+1'}$ 为经过扰动 t 分布之后麻雀的位置；$t(I)$ 为自由度参数为 I 的 t 分布变异算子。

引入自适应 t 分布算子，对于算法性能的提升有很大帮助，但如果对所有个体均使用该策略，算法的改进效果会适得其反。因此，根据随机概率 w 来决定麻雀个体是否进行 t 分布变异。若随机概率 $w\leqslant 0.5(w\in[0,1])$，则进行 t 分布变异；否则，按原算法的公式进行更新，位置更新计算式为

$$\boldsymbol{X}_{i,j}^{I+1}=\begin{cases}\boldsymbol{X}_{i,j}^{I+1'} & w\leqslant 0.5\\ \boldsymbol{X}_{i,j}^{I+1} & w>0.5\end{cases}。\tag{9}$$

2.4 融合黄金正弦策略的警觉者位置更新算法

黄金正弦算法（golden-SA）[20]是一种新型元启发式算法，该算法利用正弦函数的周期性特征进行迭代寻优，并通过黄金分割率缩小搜索空间，使其在较优的空间内进行搜索，有助于提升算法收敛速度，提升全局搜索效率，该算法的计算式为

$$\boldsymbol{X}_{i,j}^{I+1}=\boldsymbol{X}_{i,j}^{I}\left|\sin r_1\right|-r_2\sin r_1\left|a_1\boldsymbol{X}_{\text{best}}^{I}-a_2\boldsymbol{X}_{i,j}^{I}\right|,\tag{10}$$

式中：随机数 $r_1\in[0,2\pi]$，决定下一代个体移动距离；随机数 $r_2\in[0,\pi]$，决定下一代个体移动方向；a_1 和 a_2 是通过黄金分割数 τ 获得的系数，$\tau=\left(\sqrt{5}-1\right)/2$，$a_1=-\pi(1-\tau)+\pi\tau$，$a_2=-\pi\tau+\pi(1-\tau)$。

麻雀搜索算法中的警觉者具有反捕食行为，这种行为可以通过不断逃离当前位置来改善种群的质量。然而，随着迭代次数的增加，后期种群收敛，警觉者逃逸范围减小，从局部最优值逃逸的能力降低。为了克服上述缺点，利用黄金正弦策略来对警觉者位置进行更新，加快算法搜索速度，提高算法寻优能力。融合黄金正弦策略的警觉者位置更新为

$$\boldsymbol{X}_{i,j}^{I+1}=\begin{cases}\boldsymbol{X}_{i,j}^{I}\left|\sin r_1\right|-r_2\sin r_1\left|a_1\boldsymbol{X}_{\text{best}}^{I}-a_2\boldsymbol{X}_{i,j}^{I}\right| & f_i>f_{\text{b}}\\ \boldsymbol{X}_{i,j}^{I}+K\left(\dfrac{\left|\boldsymbol{X}_{i,j}^{I}-\boldsymbol{X}_{\text{worst}}^{I}\right|}{\left(f_i-f_{\text{w}}\right)+\varepsilon}\right) & f_i=f_{\text{b}}\end{cases}。\tag{11}$$

经过上述改进后的算法简称 ATSSA。

2.5 算法流程与伪代码

ATSSA 伪代码如下。

Input：

n: 种群总数

b: 发现者所占比例

c: 警觉者所占比例
M: 最大迭代次数
R_2: 预警值
S: 安全值
1: 采用Bernoulli映射初始化种群（式（4））
2: **for** I = 1 **to** M **do**
3:　计算适应度值f_i并排序，找出最优适应度值f_b、最差适应度值f_w及对应位置$\boldsymbol{X}_{best}$、$\boldsymbol{X}_{worst}$
4:　**for** i = 1 **to** (b×n) **do**
5:　　使用式（6）更新发现者的位置
6:　**end for**
7:　**for** i = (b×n+1) **to** n **do**
8:　　使用式（9）更新加入者位置
9:　**end for**
10:　**for** i = 1 **to** (c×n) **do**
11:　　使用式（11）更新警觉者位置
12:　**end for**
13:　计算麻雀个体适应度值f_i，并更新最优适应度值f_b和对应的麻雀个体位置$\boldsymbol{X}_{best}$
14:　I = I+1
15: **end for**
Output: $\boldsymbol{X}_{best}$和f_b

ATSSA流程见图3。

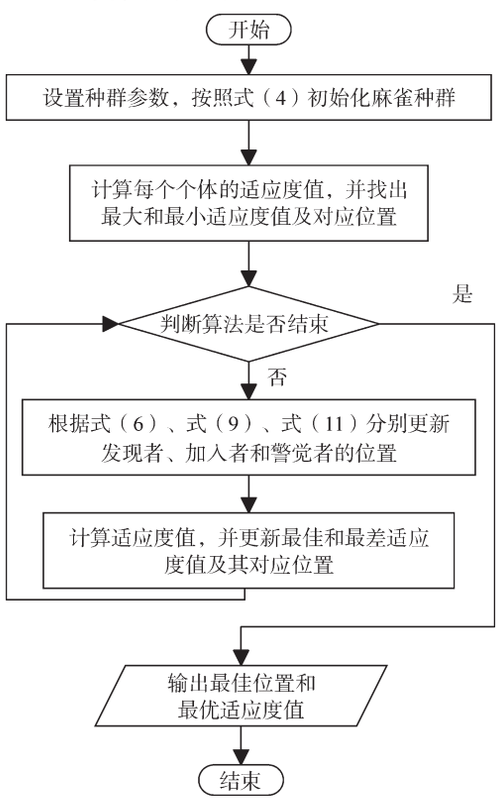


图3　ATSSA流程

Fig.3　ATSSA algorithm flow

2.6　时间复杂度分析

在原始SSA中，假设初始化种群参数的时间为t_1，每一维生成随机数的时间为t_2，求解适应度值所需要的时间为$f(d)$，初始化阶段的时间复杂度为

$$T_1=O\left(t_1+n\left(f\left(d\right)\right)+dt_2\right)=O\left(d+f\left(d\right)\right)。$$

发现者每一维按式（1）更新的时间为t_3，Q、α和R_2都是取值范围为(0,1)的随机数，生成的时间均为t_4，发现者位置更新的时间复杂度为

$$T_2=O\left(bn\left(t_3+t_4+t_4+t_4\right)d\right)=O\left(d\right)。$$

加入者每一维按式（2）更新的时间为t_5，参数Q的生成时间为t_4，加入者位置更新的时间复杂度为

$$T_3=O\left(\left(1-b\right)n\left(t_4+t_5\right)d\right)=O\left(d\right)。$$

警觉者每一维按式（3）更新的时间为t_6，参数β、K的生成时间分别为t_7和t_8，警觉者位置更新的时间复杂度为

$$T_4=O\left(cn\left(t_6+t_7+t_8\right)d\right)=O\left(d\right)。$$

综上，SSA的时间复杂度为

$$T=T_1+M\left(T_2+T_3+T_4\right)=O\left(d+f\left(d\right)\right)。$$

在ATSSA中，假设初始化种群参数的时间为u_1，每一维生成Bernoulli映射混沌值的时间为u_2，初始化阶段的时间复杂度为

$$T_1'=O\left(u_1+n\left(f\left(d\right)\right)+du_2\right)=O\left(d+f\left(d\right)\right)。$$

发现者每一维按式（6）更新的时间为u_3，Q、r和R_2都是取值范围为(0,1)的随机数，生成的时间均为u_4，参数N的生成时间为u_5，发现者位置更新的时间复杂度为

$$T_2'=O\left(bn\left(u_3+u_4+u_4+u_4+u_5\right)d\right)=O\left(d\right)。$$

加入者每一维按式（9）更新的时间为u_6，参数Q、w的生成时间为u_4，加入者位置更新的时间复杂度为

$$T_3'=O\left(\left(1-b\right)n\left(u_4+u_4+u_6\right)d\right)=O\left(d\right)。$$

警觉者每一维按式（11）更新的时间为u_7，参数r_1、r_2的生成时间分别为u_8和u_9，K的生成时间为u_{10}，警觉者位置更新的时间复杂度为

$$T_4'=O\left(cn\left(u_7+u_8+u_9+u_{10}\right)d\right)=O\left(d\right)。$$

综上，ATSSA的时间复杂度为

$$T'=T_1'+M\left(T_2'+T_3'+T_4'\right)=O\left(d+f\left(d\right)\right),$$

可以看出ATSSA的时间复杂度与原始的SSA相同。

3　仿真实验分析

3.1　实验设计

为评估ATSSA的性能表现，选取14个基准测试函数开展2组仿真实验，基准测试函数见表1。其中，F_1～F_6为单峰函数，F_7～F_{12}为多峰函数，F_{13}、F_{14}为固定低维多峰函数。第一组实验是利用

消融实验对所使用的策略进行测试，验证改进策略的有效性。第二组实验是将ATSSA与其他7种算法进行比较，同时运用Wilcoxon秩和检验，验证算法改进的有效性。所有实验均基于Windows 11（64 bit）操作系统，硬件配置为Intel(R) Core (TM) i5−8265U CPU @1.60GHz、1.80 GHz，8 GB内存，使用Matlab2017a软件实现。

表1 基准测试函数

Tab.1 benchmark test functions

编号	函数名称	维度	变量范围	最优值
F_1	Sphere	30	[-100,100]	0
F_2	Schwefel 2.22	30	[-10,10]	0
F_3	Quadric	30	[-100,100]	0
F_4	Schwefel 2.21	30	[-100,100]	0
F_5	Power	30	[-100,100]	0
F_6	Zakharov	30	[-5,10]	0
F_7	Rosenbrock	30	[-10,10]	0
F_8	Quartic	30	[-1.28,1.28]	0
F_9	Schwefel 2.26	30	[-500,500]	-418.982 9d
F_{10}	Rastrigrin	30	[-5.12,5.12]	0
F_{11}	Ackley	30	[-32,32]	0
F_{12}	Griewing	30	[-600,600]	0
F_{13}	Schaffer F7	6	[-100,100]	0
F_{14}	Foxholes	2	[-65,65]	1

3.2 策略有效性分析

为探究所提改进策略的有效性，对SSA分别进行单策略改进，形成以下对比算法：仅使用Bernoulli初始化的麻雀搜索算法（SSA1）；仅融入鱼鹰优化算法的麻雀搜索算法（SSA2）；仅引入自适应t分布的麻雀搜索算法（SSA3）；仅结合黄金正弦策略的麻雀搜索算法（SSA4）。将上述改进算法与本文提出的ATSSA以及原始SSA进行比较分析。

各算法独立运行30次，分别取最优值、标准差和均值作为评价指标，部分函数测试结果见表2。由表2可以看出，在函数F_1～F_6、F_7和F_{13}上，ATSSA都能找到理论最优值0，说明ATSSA的寻优精度很高。单策略改进的4种算法中，SSA3和SSA4在函数中找到最优解的次数最多，其次是SSA1，最后是SSA2。这表明ATSSA的寻优能力由策略SSA3和SSA4主导，SSA1和SSA2辅助。

部分函数的收敛效果见图4，其中，f为适应度。由图4可以看出，在结合各个策略之后，ATSSA的寻优精度和收敛速度都显著提高，并且能够及时跳出局部最优。

表2 部分函数策略有效性分析测试结果

Tab.2 partial function test results of policy effectiveness analysis

算法	F_1			F_2			F_3			F_4		
	最优值	标准差	均值	最优值	标准差	均值	最优值	标准差	均值	最优值	标准差	均值
ATSSA	0	0	0	0	0	0	0	0	0	0	0	0
SSA1	1.73×10^{-208}	3.19×10^{-59}	5.83×10^{-60}	1.65×10^{-288}	1.22×10^{-28}	2.53×10^{-29}	0	1.06×10^{-25}	2.25×10^{-26}	1.97×10^{-132}	9.63×10^{-24}	1.76×10^{-24}
SSA2	3.30×10^{-148}	1.25×10^{-122}	2.30×10^{-123}	1.66×10^{-75}	2.32×10^{-65}	5.55×10^{-66}	3.91×10^{-46}	2.47×10^{-27}	4.55×10^{-28}	1.21×10^{-57}	4.72×10^{-38}	8.65×10^{-39}
SSA3	0	0	0	1.63×10^{-320}	0	4.15×10^{-226}	0	0	0	0	0	1.59×10^{-215}
SSA4	0	0	0	0	0	2.90×10^{-275}	0	0	0	0	0	0
SSA	0	6.27×10^{-63}	1.64×10^{-63}	5.58×10^{-32}	2.53×10^{-32}	5.17×10^{-33}	1.89×10^{-151}	1.04×10^{-26}	2.76×10^{-27}	1.57×10^{-252}	3.41×10^{-28}	6.34×10^{-29}

算法	F_5			F_6			F_7			F_{13}		
	最优值	标准差	均值	最优值	标准差	均值	最优值	标准差	均值	最优值	标准差	均值
ATSSA	0	0	0	0	0	0	0	5.65×10^{-18}	1.17×10^{-18}	0	0	0
SSA1	6.53×10^{-320}	0	2.53×10^{-217}	1.41×10^{-77}	9.32×10^{-17}	1.72×10^{-17}	7.93×10^{-10}	1.53×10^{-5}	8.51×10^{-6}	0	3.92×10^{-18}	7.17×10^{-19}
SSA2	8.87×10^{-252}	0	2.25×10^{-227}	8.60×10^{-29}	3.63×10^{-13}	1.22×10^{-13}	0	7.25×10^{-5}	3.89×10^{-5}	3.15×10^{-38}	5.86×10^{-32}	1.18×10^{-32}
SSA3	0	0	0	0	0	0	6.66×10^{-9}	1.49×10^{-4}	4.50×10^{-5}	0	0	0
SSA4	3.80×10^{-255}	0	1.04×10^{-204}	0	0	0	9.40×10^{-9}	5.12×10^{-4}	1.35×10^{-4}	0	0	0
SSA	1.91×10^{-229}	0	1.89×10^{-218}	0	2.94×10^{-18}	5.37×10^{-19}	2.22×10^{-8}	5.35×10^{-4}	2.21×10^{-4}	0	4.68×10^{-18}	1.23×10^{-18}

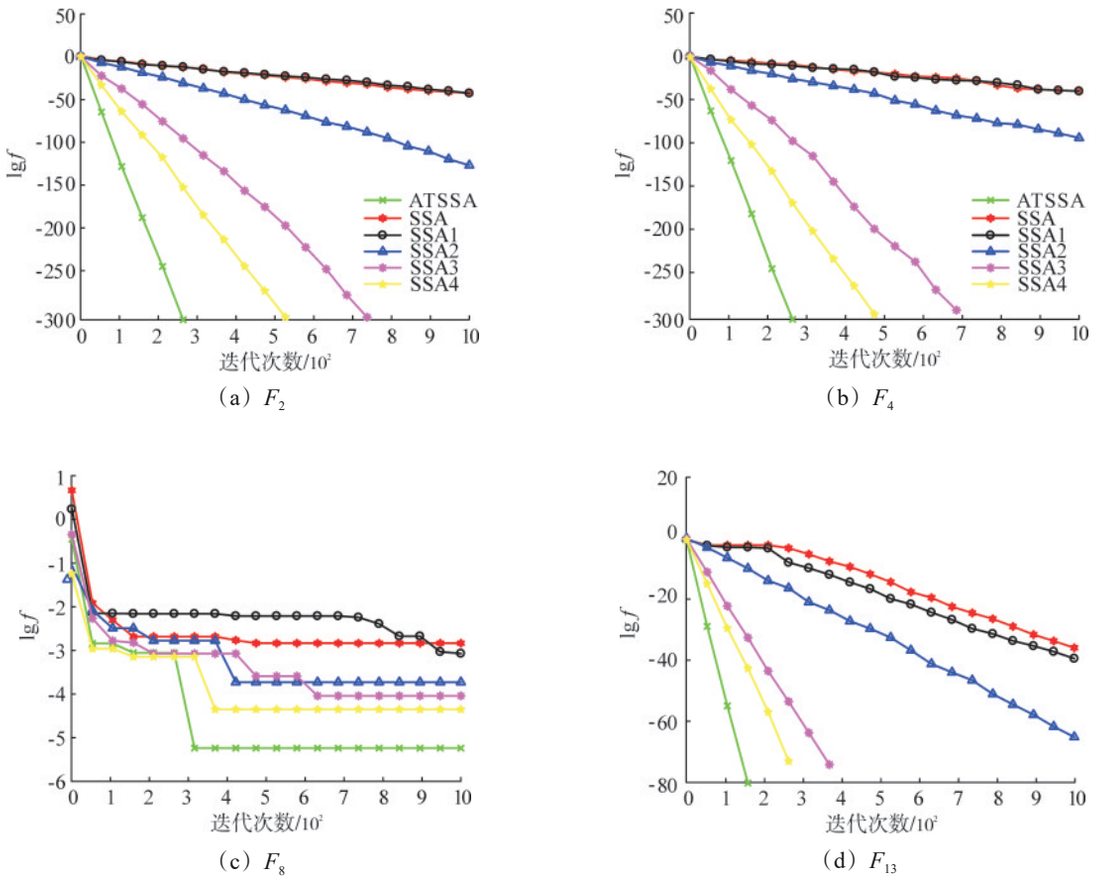


图 4　部分函数策略有效性分析收敛效果

Fig.4　part function convergence effect of strategy effectiveness analysis

3.3　对比分析

为了验证 ATSSA 的性能，将 ATSSA 与传统麻雀算法（SSA）[8]、多策略改进的麻雀算法（MISSA）[14]、混合正弦余弦算法和 Levy 飞行的麻雀算法（ISSA）[21]、自适应螺旋飞行麻雀算法（ASFSSA）[22]、基于莱维飞行扰动策略的麻雀算法（LSSA）[15]、蜣螂优化算法（DBO）[7]和北方苍鹰优化算法（NGO）[23]进行对比实验。上述麻雀搜索算法的发现者及警觉者比例均为 0.2，安全值为 0.8。各算法的种群规模为 30，迭代次数为 1 000。

为了增强实验结论的可信度与说服力，以均值、标准差、最优值和非参数 Wilcoxon 秩和检验[24]作为评价标准。Wilcoxon 秩和检验的显著水平设为 5%。原假设为两种算法之间没有显著差异，如果检验结果 p<5%，表示拒绝原假设，认为两种算法之间存在显著差异；如果 p>5%，则表示两种算法的差异不明显，性能相当。将均值最小的算法视为最优算法，记作 NA。当均值相同时，则认为标准差越小的算法越好。测试结果见表 3。

由表 3 可知，从收敛精度来看，ATSSA 在函数优化方面性能最好，尤其是在函数 F_1～F_4、F_6 和 F_{13} 上，每次都可以准确找到全局最优解 0。虽然 ATSSA 在函数 F_7 和 F_8 上未寻到最优值 0，但与其他 7 种算法相比，ATSSA 的求解效果更优，收敛精度更高。从算法稳定性的角度来看，ATSSA 的标准差始终最小，说明 ATSSA 比其他算法更稳定，尤其在函数 F_1～F_4、F_6 和 F_{13} 上，ATSSA 的均值和标准差均为 0，说明 ATSSA 具有很强的鲁棒性。

从秩和检验结果 p 来看，ATSSA 在 8 个基准函数上的优化性能明显优于 MISSA、LSSA、ISSA、DBO 和 NGO；相较于 ASFSSA，ATSSA 在 5 个函数上展现出更好的寻优性能，在 3 个函数上获得同等的性能指标；与原始 SSA 相比，ATSSA 在 7 个测试函数中呈现出统计显著性优势，仅在 1 个函数上与 SSA 效果相当。实验数据表明，ATSSA 具有明显优越性，其综合性能优于各对比算法。

表3 不同算法测试结果对比

Tab.3 comparison of test results of different algorithms

算法	F_1				F_2				F_3			
	最优值	标准差	均值	p	最优值	标准差	均值	p	最优值	标准差	均值	p
ATSSA	0	0	0	NA	0	0	0	NA	0	0	0	NA
MISSA	1.71×10^{-167}	1.77×10^{-146}	3.24×10^{-147}	1.21×10^{-12}	4.46×10^{-82}	3.22×10^{-74}	1.12×10^{-74}	1.21×10^{-12}	7.43×10^{-83}	1.50×10^{-61}	2.74×10^{-62}	1.21×10^{-12}
ASFSSA	0	0	0	NA	1.76×10^{-250}	0	5.57×10^{-184}	1.21×10^{-12}	0	0	0	NA
LSSA	3.03×10^{-142}	8.19×10^{-11}	1.50×10^{-11}	1.21×10^{-12}	7.07×10^{-194}	2.46×10^{-12}	4.50×10^{-13}	1.21×10^{-12}	2.30×10^{-96}	1.51×10^{-12}	2.89×10^{-13}	1.21×10^{-12}
ISSA	0	0	6.16×10^{-243}	1.66×10^{-11}	6.80×10^{-169}	5.99×10^{-126}	1.09×10^{-126}	1.21×10^{-12}	6.86×10^{-127}	5.91×10^{-55}	1.08×10^{-55}	1.21×10^{-12}
SSA	5.92×10^{-260}	7.94×10^{-55}	1.45×10^{-55}	1.21×10^{-12}	1.25×10^{-171}	1.43×10^{-28}	2.68×10^{-29}	1.21×10^{-12}	1.46×10^{-297}	1.82×10^{-29}	5.11×10^{-30}	1.21×10^{-12}
DBO	1.18×10^{-152}	3.25×10^{-104}	8.33×10^{-105}	1.21×10^{-12}	2.27×10^{-90}	2.87×10^{-58}	5.65×10^{-59}	1.21×10^{-12}	1.89×10^{-141}	4.63×10^{-25}	8.45×10^{-26}	1.21×10^{-12}
NGO	1.40×10^{-90}	1.42×10^{-86}	4.22×10^{-87}	1.21×10^{-12}	9.61×10^{-47}	2.61×10^{-45}	1.82×10^{-45}	1.21×10^{-12}	1.67×10^{-27}	4.42×10^{-22}	1.44×10^{-22}	1.21×10^{-12}

算法	F_4				F_6				F_7			
	最优值	标准差	均值	p	最优值	标准差	均值	p	最优值	标准差	均值	p
ATSSA	0	0	0	NA	0	0	0	NA	1.34×10^{-9}	5.17×10^{-5}	2.36×10^{-5}	NA
MISSA	2.45×10^{-80}	1.94×10^{-69}	3.69×10^{-70}	1.21×10^{-12}	2.80×10^{-49}	6.38×10^{-36}	1.17×10^{-36}	1.21×10^{-12}	2.46×10^{-6}	1.00×10^{1}	5.44×10^{0}	7.60×10^{-7}
ASFSSA	1.24×10^{-318}	0	1.32×10^{-174}	1.21×10^{-12}	0	0	4.94×10^{-324}	3.34×10^{-1}	7.38×10^{-8}	1.21×10^{-3}	6.48×10^{-4}	9.21×10^{-5}
LSSA	2.54×10^{-47}	9.76×10^{-6}	1.78×10^{-6}	1.21×10^{-12}	0	1.97×10^{-5}	4.16×10^{-6}	1.66×10^{-11}	1.78×10^{-5}	1.58×10^{-2}	9.38×10^{-3}	1.78×10^{-10}
ISSA	6.20×10^{-174}	1.57×10^{-139}	3.19×10^{-140}	1.21×10^{-12}	2.10×10^{-89}	2.04×10^{-22}	3.73×10^{-23}	1.21×10^{-12}	1.94×10^{-8}	2.18×10^{-3}	5.02×10^{-4}	2.27×10^{-3}
SSA	7.59×10^{-131}	2.01×10^{-30}	7.21×10^{-31}	1.21×10^{-12}	2.57×10^{-157}	4.38×10^{-18}	1.19×10^{-18}	1.21×10^{-12}	4.59×10^{-9}	5.77×10^{-5}	2.80×10^{-5}	9.35×10^{-1}
DBO	1.73×10^{-79}	9.52×10^{-55}	2.77×10^{-55}	1.21×10^{-12}	3.86×10^{-66}	3.91×10^{2}	9.09×10^{1}	1.21×10^{-12}	2.52×10^{1}	2.22×10^{-1}	2.58×10^{1}	3.02×10^{-11}
NGO	2.89×10^{-38}	1.93×10^{-37}	2.15×10^{-37}	1.21×10^{-12}	3.95×10^{2}	9.93×10^{2}	1.62×10^{3}	1.21×10^{-12}	2.48×10^{1}	4.08×10^{-1}	2.59×10^{1}	3.02×10^{-11}

算法	F_8				F_{13}			
	最优值	标准差	均值	p	最优值	标准差	均值	p
ATSSA	1.40×10^{-5}	1.40×10^{-4}	1.58×10^{-4}	NA	0	0	0	NA
MISSA	4.19×10^{-5}	1.25×10^{-3}	1.43×10^{-3}	1.20×10^{-8}	4.63×10^{-40}	1.30×10^{-35}	6.55×10^{-36}	1.21×10^{-12}
ASFSSA	1.76×10^{-5}	2.53×10^{-4}	3.20×10^{-4}	1.03×10^{-2}	0	0	0	NA
LSSA	4.57×10^{-5}	1.15×10^{-3}	1.41×10^{-3}	4.69×10^{-8}	3.88×10^{-28}	2.92×10^{-10}	6.08×10^{-11}	1.21×10^{-12}
ISSA	4.49×10^{-5}	6.53×10^{-4}	8.58×10^{-4}	3.59×10^{-5}	0	1.64×10^{-70}	3.71×10^{-71}	1.93×10^{-10}
SSA	1.41×10^{-4}	1.61×10^{-3}	1.75×10^{-3}	9.76×10^{-10}	1.55×10^{-71}	2.71×10^{-18}	5.05×10^{-19}	1.21×10^{-12}
DBO	7.03×10^{-5}	7.87×10^{-4}	1.22×10^{-3}	2.03×10^{-9}	7.59×10^{-43}	3.75×10^{-23}	6.86×10^{-24}	1.21×10^{-12}
NGO	8.81×10^{-5}	3.91×10^{-4}	6.14×10^{-4}	4.69×10^{-8}	4.52×10^{-28}	2.65×10^{-27}	3.42×10^{-27}	1.21×10^{-12}

注：NA表示最优算法。

算法收敛效果见图5。与其他7种算法相比，对于单峰函数F_2、F_4和F_6，ATSSA的收敛速度最快，能够以更少的迭代次数收敛至最优值。在多峰函数F_7、F_8和F_{13}中，虽然ATSSA的收敛速度相较于单峰函数有所降低，但其收敛速度和收敛精度明显优于其他7种算法，跳出局部最优的能力也非常显著。综上，ATSSA能够有效规避局部最优陷阱，具有较好的全局搜索能力。在相同的实验条件下，ATSSA在寻优性能与鲁棒性方面明显优于其他对比算法。

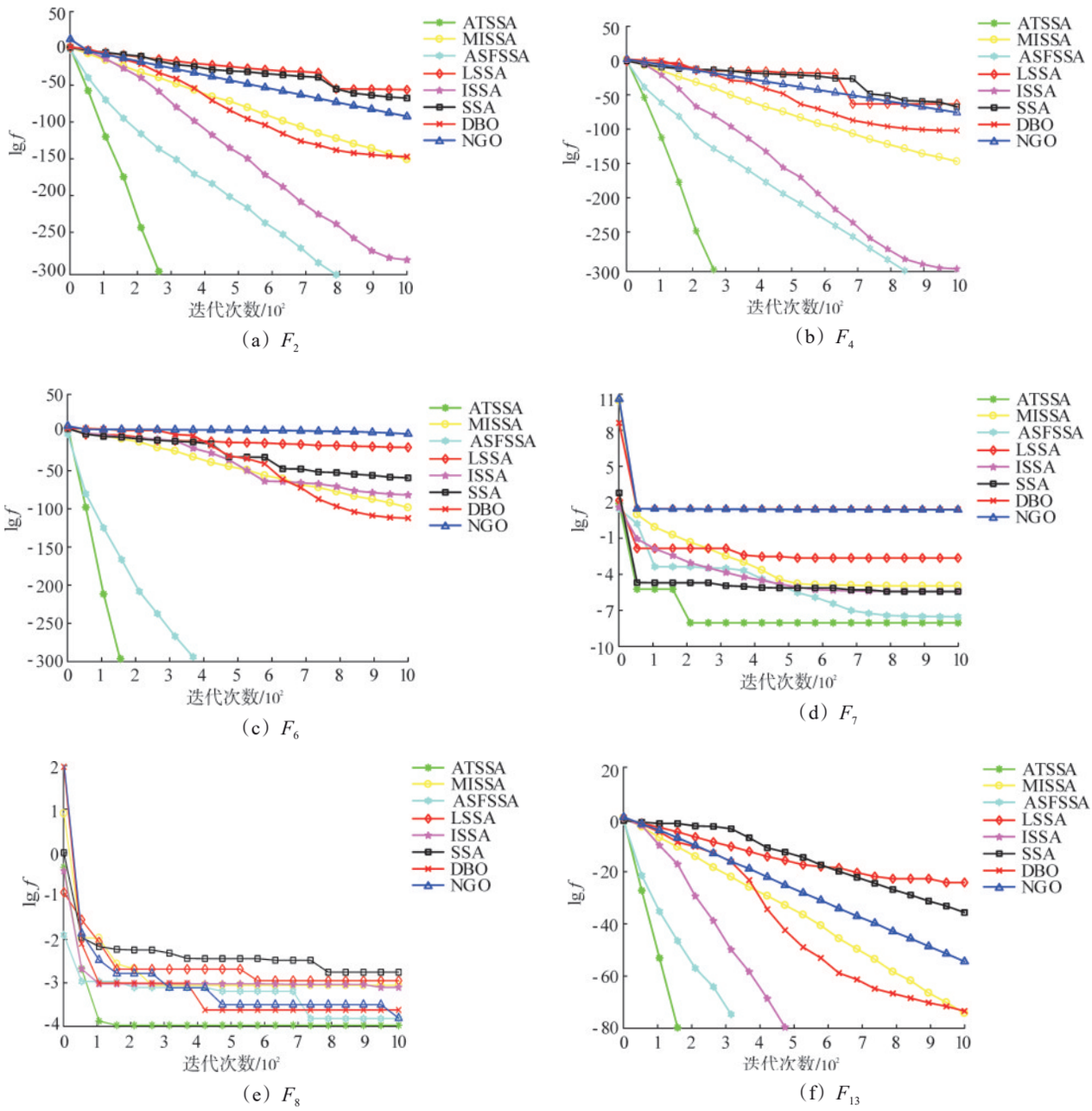


图 5　不同算法收敛效果对比

Fig.5　comparison of convergence effects of different algorithms

4　结论

（1）针对 SSA 存在的不足，提出了一种改进搜索机制的自适应 t 分布麻雀搜索算法。采用 Bernoulli 混沌映射增加初始种群的多样性，提高了初始解的质量。发现者搜索机制的改变使算法搜索范围增大，改善了全局搜索能力较差的问题。

（2）自适应 t 分布算子的引入使算法在全局搜索与局部开发之间达到了很好的平衡，提升了算法的收敛速度。黄金正弦策略的加入提高了算法跳出局部极值的能力。

（3）与其他 7 种算法在 14 个基准函数上进行比较，并进行 Wilcoxon 秩和检验，结果表明 ATSSA 具有比其他对比算法更优的搜索能力、收敛速度和鲁棒性。

（4）基于 ATSSA 的良好性能，在未来的研究中计划将其应用于现实的优化问题，如路径规划、选址研究、生产车间调度问题等。

参考文献（References）：

[1] POVALEJ Ž. Quasi-Newton's method for multiobjective optimization[J].Journal of Computational and Applied Mathematics, 2014,255:765-777.

[2] ZHANG J G, XIAO Y H, WEI Z X. Nonlinear conjugate gradient methods with sufficient descent condition for large-scale unconstrained optimization[J]. Mathematical Problems in Engineering,2009,2009:243290.

[3] LI M W,WANG Y T,GENG J,et al.Chaos cloud quantum bat hybrid optimization algorithm[J]. Nonlinear Dynamics, 2021, 103: 1167-1193.

[4] YIN S H, LUO Q F, DU Y L, et al. DTSMA: dominant swarm with adaptive t-distribution mutation-based slime mould algorithm[J]. Mathematical Biosciences and Engineering,2022,19(3):2240-2285.

[5] MIRJALILI S. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems[J]. Neural Computing and Applications, 2016,27(4):1053-1073.

[6] DHIMAN G,KUMAR V.Seagull optimization algorithm:theory and its applications for large-scale industrial engineering problems[J]. Knowledge-Based Systems,2019,165:169-196.

[7] XUE J K, SHEN B. Dung beetle optimizer: a new meta-heuristic algorithm for global optimization[J]. The Journal of Supercomputing,2023,79(7):7305-7336.

[8] XUE J K, SHEN B. A novel swarm intelligence optimization approach: sparrow search algorithm[J]. Systems Science & Control Engineering,2020,8(1):22-34.

[9] WU C Y,FU X S,PEI J K,et al.A novel sparrow search algorithm for the traveling salesman problem[J]. IEEE Access, 2021, 9: 153456-153471.

[10] LIU G Y, SHU C, LIANG Z W, et al. A modified sparrow search algorithm with application in 3d route planning for UAV[J]. Sensors,2021,21(4):1224.

[11] WU D M, YUAN C Z. Threshold image segmentation based on improved sparrow search algorithm[J]. Multimedia Tools and Applications,2022,81(23):33513-33546.

[12] YAN S Q, LIU W D, LI X Q, et al. Comparative study and improvement analysis of sparrow search algorithm[J]. Wireless Communications and Mobile Computing,2022,2022:4882521.

[13] TANG Y Q,LI C H,LI S,et al.A fusion crossover mutation sparrow search algorithm[J]. Mathematical Problems in Engineering, 2021, 2021:9952606.

[14] CHEN G, ZHU D L, CHEN X Y. Similarity detection method of science fiction painting based on multi-strategy improved sparrow search algorithm and Gaussian pyramid[J]. Multimedia Tools and Applications,2024,83(14):41597-41636.

[15] 马卫,朱婀.基于莱维飞行扰动策略的麻雀搜索算法[J].应用科学学报,2022,40(1):116-130.
MA Wei,ZHU Xian.Sparrow search algorithm based on levy flight disturbance strategy [J]. Journal of Applied Sciences, 2022, 40(1): 116-130.

[16] 宋立钦,陈文杰,陈伟海,等.基于混合策略的麻雀搜索算法改进及应用[J].北京航空航天大学学报,2023,49(8):2187-2199.
SONG Liqin,CHEN Wenjie,CHEN Weihai,et al.Improvement and application of hybrid strategy-based sparrow search algorithm[J]. Journal of Beijing University of Aeronautics and Astronautics, 2023,49(8):2187-2199.

[17] LIU T, MENG X Q. Hybrid strategy improved sparrow search algorithm in the field of intrusion detection[J].IEEE Access,2023, 11:32134-32151.

[18] DEHGHANI M,TROJOVSKÝ P.Osprey optimization algorithm:a new bio-inspired metaheuristic algorithm for solving engineering optimization problems[J]. Frontiers in Mechanical Engineering, 2023,8:1126450.

[19] ZHANG C,YANG Y.Porcellio scaber algorithm with t-distributed elite mutation for global optimization[J]. Scientific Programming, 2022,2022:1502988.

[20] TANYILDIZI E,DEMIR G.Golden sine algorithm:a novel math-inspired algorithm[J]. Advances in Electrical and Computer Engineering,2017,17(2):71-78.

[21] 毛清华,张强,毛承成,等.混合正弦余弦算法和Lévy飞行的麻雀算法[J].山西大学学报(自然科学版),2021,44(6):1086-1091.
MAO Qinghua, ZHANG Qiang, MAO Chengcheng, et al. Mixing sine and cosine algorithm with lévy flying chaotic sparrow algorithm[J]. Journal of Shanxi University (Natural Science Edition), 2021,44(6):1086-1091.

[22] OUYANG C,QIU Y X,ZHU D L.Adaptive spiral flying sparrow search algorithm[J].Scientific Programming,2021,2021:6505253.

[23] DEHGHANI M, HUBÁLOVSKÝ Š, TROJOVSKÝ P. Northern goshawk optimization: a new swarm-based algorithm for solving optimization problems[J].IEEE Access,2021,9:162059-162080.

[24] DERRAC J,GARCÍA S,MOLINA D,et al.A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms[J]. Swarm and Evolutionary Computation,2011,1(1):3-18.