Advances in Applied Mathematics 应用数学进展, 2023, 12(7), 3165-3178
Published Online July 2023 in Hans. https://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2023.127317

Hans汉斯

基于惯性权值和自适应变异改进的麻雀搜索算法

韩斌彬*，朱金秋

南京邮电大学理学院，江苏 南京

收稿日期：2023年6月14日；录用日期：2023年7月9日；发布日期：2023年7月17日

摘　要

传统的麻雀搜索算法(SSA)在求解复杂优化问题的时候，可能出现陷入局部最优的情况。为改进SSA算法以提高算法的收敛速度与收敛精度，本文在已有研究的基础上提出了基于惯性权值和自适应变异改进的麻雀搜索算法(IASSA)。首先利用改进的Chebyshev混沌初始化种群来增加种群的多样性，提高麻雀个体的遍历性，以期望种群可以尽可能均匀分布搜索空间；其次，增加一种惯性权值来改善发现者的更新位置，以减小过早陷入局部最优的概率；最后，提出一种自适应的选择变异策略使得麻雀在陷入局部最优时能够跳出。通过对10个基准测试函数进行的仿真实验，结果表明，所提算法较原算法有着更好的收敛速度与收敛精度。

关键词

麻雀搜索算法，混沌映射，惯性权值，自适应权重，高斯柯西变异

Improved Sparrow Search Algorithm Based on Inertia Weight Value and Adaptive Mutation

Binbin Han*, Jinqiu Zhu

College of Science, Nanjing University of Posts and Telecommunications, Nanjing Jiangsu

Received: Jun. 14th, 2023; accepted: Jul. 9th, 2023; published: Jul. 17th, 2023

*通讯作者。

文章引用: 韩斌彬, 朱金秋. 基于惯性权值和自适应变异改进的麻雀搜索算法[J]. 应用数学进展, 2023, 12(7): 3165-3178. DOI: 10.12677/aam.2023.127317

Abstract

The traditional sparrow search algorithm (SSA) may fall into local optimization when solving complex optimization problems. In order to improve the SSA algorithm to improve the convergence speed and accuracy of the algorithm, an improved sparrow search algorithm (IASSA) based on inertia weight value and adaptive mutation is proposed based on the existing research. Firstly, the improved Chebyshev chaos is used to initialize the population to increase the diversity of the population and improve the ergodicity of sparrow individuals, so as to expect the population to distribute the search space as evenly as possible; Secondly, an inertia weight value is added to improve the update position of the discoverer to reduce the probability of falling into the local optimum too early; Finally, an adaptive selection mutation strategy is proposed to make sparrows jump out when they fall into local optimization. The simulation results of 10 benchmark functions show that the proposed algorithm has better convergence speed and convergence accuracy than the original algorithm.

Keywords

Sparrow Search Algorithm, Chaotic Mapping, Inertia Weight Value, Adaptive Weight, Gauss Cauchy Variation

1. 前言

一些学者在受到生物群体间合作、竞争、捕食等行为的启发，提出了许多智能优化算法，如：萤火虫算法(Firefly Algorithm, FA) [1]、粒子群算法(Particle Swarm Optimization, PSO) [2]、灰狼优化算法(Grey Wolf Optimizer, GWO) [3]、鲸鱼优化算法(Whale Optimization Algorithm, WOA) [4]、正弦余弦算法(Sine Cosine Algorithm, SCA) [5]等，这些算法在解决复杂优化问题时提高了效率与准确率。

麻雀搜索算法(Sparrow Search Algorithm, SSA) [6]是东华大学的薛建凯等人于 2020 年提出的一种新型群体智能优化算法，该算法以结构简单、寻优效果好、设定参数少等优点在图像[7]、工业[8]、农业[9]等领域受到广泛应用。但是该算法也存在一些缺陷，尤其是在接近全局最优时，会出现种群多样性减少，陷入局部最优的情况。

考虑到麻雀搜索算法存在的缺陷后，许多学者提出改进的方法来对抗局部最优：唐延强[10]等人使用猫映射混沌序列初始化种群，同时引入柯西变异和 Tent 混沌扰动拓展局部搜索能力，最后提出探索者－跟随者数量自适应调整策略来提高算法的寻优精度；陈功[11]等人首先采用一种无限次折叠的 ICMIC 混沌初始化种群，再融入一种螺旋探索策略和基于精英差分和随机反向的混合变异策略，加快算法收敛速度，改善算法跳出局部最优的能力；汤安迪[12]等人引入等级制度和布朗运动，使得该算法加快收敛速度并能够跳出局部最优；毛清华[13]等人利用柯西变异和反向学习在最优麻雀位置进行扰动变异，增强了算法跳出局部空间的能力；柳长安[14]等人将反向学习和自适应 t 分布变异策略融入麻雀搜索算法中，增强了算法后期局部搜索能力；王辉[15]等人引入高维 Sine 混沌映射、衰减因子和柯西变异和变化选择策略，缓解了陷入局部最优的情况，增强了搜索能力；胡树斌[16]等人利用精英反向学习机制、黄金正弦策略和莱维飞行随机步长，克服了麻雀搜索算法存在的迭代过程中早熟收敛等不足。

虽然上述的文献从种群初始化、变异策略等角度对麻雀搜索算法进行了改进，但是从寻优的结果上来看，仍有很大的提升空间。为了改进 SSA 算法以提高算法的收敛性与收敛精度，本文在已有研究的基础上提出了基于惯性权值和自适应变异改进的麻雀搜索算法，用来提升算法的收敛性与收敛精度。本文首先利用改进后的 Chebyshev 混沌初始化种群来增加种群多样性，提高麻雀个体的遍历性；其次增加一种惯性权值来改善发现者的更新位置，用来降低过早陷入局部最优的概率；最后提出一种自适应的选择变异策略使得麻雀在陷入局部最优时能够跳出。为了证明了改进策略的有效性，本文对提出的算法进行了 10 个基准测试函数的仿真测试。

2. 麻雀搜索算法

麻雀搜索算法是受到自然界麻雀觅食的行为而启发的，将麻雀的整个种群分为发现者和跟随者。除此之外，为了防范天敌，再从群体中选取部分作为预警者。

发现者担负着整个种群觅食方向的重任，因此会更容易发现食物，它们的更新公式如下：

$$X_{i,j}^{t+1}=\begin{cases}X_{i,j}^{t}\cdot\exp\left(\dfrac{-i}{\alpha\cdot Iter_{max}}\right), & R_2<ST\\ X_{i,j}^{t}+Q\cdot L, & R_2\geq ST\end{cases}\tag{1}$$

式中：$X_{i,j}^{t}$ 表示的是在当前第 t 次迭代时，第 i 只麻雀的第 j 维位置；$X_{i,j}^{t+1}$ 表示的是在当前第 $t+1$ 次迭代时，第 i 只麻雀的第 j 维位置；随机值 $\alpha\in(0,1]$；$Iter_{max}$ 代表最大迭代次数；Q 是服从常态分布的随机值；L 表示的是一个 1 行 d 列且元素均为 1 的矩阵；预警值 $R_2\in[0,1]$ 和安全值 $ST\in[0.5,1]$，R_2 为一个随机值，而 ST 为一个定值；如果 $R_2<ST$，说明预警值小于安全值，此地安全，可以安心的觅食；若 $R_2\geq ST$，则说明预警值大于安全值，此地不安全，需要离开此地觅食。

发现者找到食物后，其余的跟随者将会根据下面的公式更新位置：

$$X_{i,j}^{t+1}=\begin{cases}Q\cdot\exp\left(\dfrac{X_{worst}^{t}-X_{i,j}^{t}}{i^2}\right), & i>\dfrac{n}{2}\\ X_{p}^{t+1}+\left|X_{i,j}^{t}-X_{p}^{t+1}\right|\cdot A^{+}\cdot L, & \text{other}\end{cases}\tag{2}$$

式中：X_{p}^{t+1} 是指式(1)中发现者找到的适应度最高的位置；X_{woret} 是指适应度最差的位置；麻雀总数为 n，其中 $i>n/2$ 表明经过适应度排序，后一半的麻雀处于饥饿状态，需要寻找其他地方寻找食物；A 表示的是一个 1 行 d 列且元素为 1 或−1 的矩阵，$A^{+}=A^{\mathrm{T}}\left(AA^{\mathrm{T}}\right)^{-1}$。

除了发现者和跟随着，为了自身的安全问题，还会从种群中抽取占种群数量 10%~20%的麻雀作为预警者。

对于预警者：

$$X_{i,j}^{t+1}=\begin{cases}X_{best}^{t}+\beta\cdot\left|X_{i,j}^{t}-X_{best}^{t}\right|, & f_i>f_g\\ X_{i,j}^{t}+k\cdot\left(\dfrac{\left|X_{i,j}^{t}-X_{worst}^{t}\right|}{(f_i-f_w)+\varepsilon}\right), & f_i=f_g\end{cases}\tag{3}$$

式中：X_{best} 是指适应度最差的位置；β 为服从标准正态分布的步长修正系数；k 是为表示位置移动的方向，取值范围在−1 与 1 之间的随机数；f_i 表示第 i 麻雀在当前迭代次数下的适应度值，f_w 和 f_g 各表示此刻群众整体最差适应度、最优适应度；为了使分母不为 0，ε 取一个极小的常数；当 $f_i>f_g$ 时，表示

预警者目前远离最优位置的麻雀，有被捕食的风险，应当向最优适应度的位置移动更新；当 $f_i = f_g$ 时，预警者已经感受到了危险，应当更新位置远离天敌。

3. 改进的麻雀搜索算法

3.1. Chebyshev 混沌映射

Chebyshev 混沌映射[17]是一维混沌序列，该序列只受到初始值和阶数的影响。其迭代表达式为：

$$y_{n+1} = \cos\left(k * \arccos\left(y_n\right)\right)$$

式中：y_n 是初始值，y_{n+1} 是迭代后的输出值；k 是控制阶数，且取值范围为 $k \geq 2$。

Chebyshev 混沌映射的分岔图如图 1 所示：

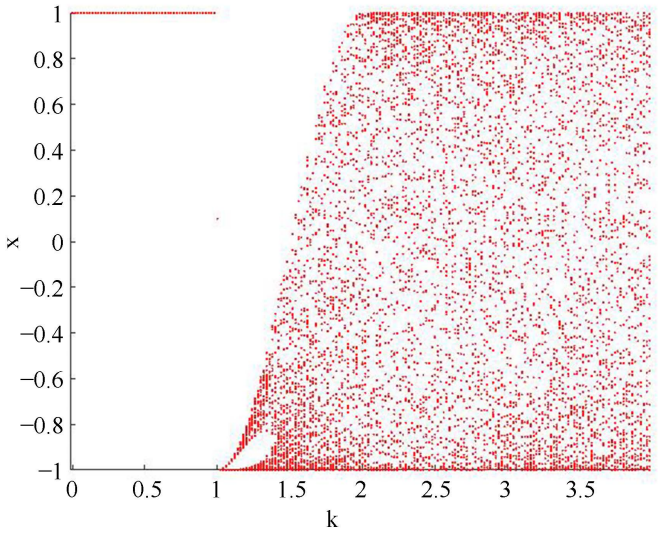


Figure 1. Bifurcation diagram of Chebyshev mapping
图 1. Chebyshev 映射的分岔图

从图 1 可以看出，Chebyshev 混沌映射有如下缺陷：1) 当阶数 $0 \leq k < 1$ 时混沌映射存在空白区域；2) 当阶数 $1 \leq k < 2$ 时混沌映射既存在空白区域又存在映射不均匀的问题；3) 当 $k \geq 2$ 时，混沌映射存在映射不均匀的问题。

所以，针对 Chebyshev 混沌映射存在的上述问题，本文引入一个非线性项[18]来对方程进行改进。改进后的方程为：

$$y_{n+1} = \cos\left(\left(\left(4+u\right)\pi - k\sin\left(\pi y_n\right)\right) * \arccos\left(y_n\right)\right) \tag{4}$$

式中：y_n 为初始值，y_{n+1} 为迭代后的输出的值；k 是控制阶数，且新的取值范围为 $k \geq 0$；经实验证明，该映射关系可以产生取值在−1 到 1 之间、不重复的混沌序列，且优于原 Chebyshev 混沌映射。其对应的分岔图如图 2 所示。

由图 2 可以看出，改进后的 Chebyshev 映射不论 k 为何值时，都不存在空白区域且映射相对均匀，原 Chebyshev 混沌映射的缺陷基本得到解决。

将此映射带入种群初始化的公式为：

$$X_i = X_{lb} + \frac{\left(X_{ub} - X_{lb}\right)\left(y_i + 1\right)}{2} \tag{5}$$

式中： X_i 为麻雀初始化后的位置； X_{ub} 和 X_{lb} 分别为迭代区域的上限和下限； y_i 是改进的 Chebyshev 混沌映射产生序列值。

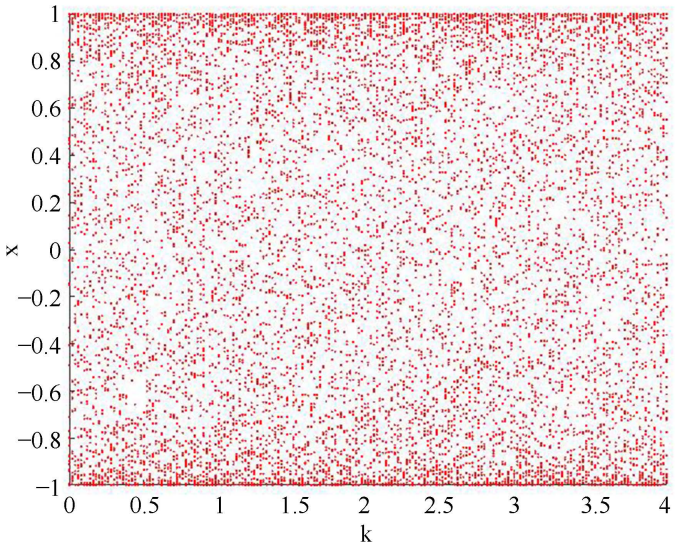


Figure 2. Bifurcation diagram of improved Chebyshev mapping
图 2. 改进的 Chebyshev 映射的分岔图

3.2. 惯性权值

根据 SSA 的算法原理，发现者肩负着整个种群寻找食物的重担，是名副其实的“侦察兵”。但是根据公式(1)，可以发现：发现者寻找食物的过程中往往是激进的，发现最优解后，其他个体迅速向最优解靠拢，这就会导致降低了种群的多样性，很容易会陷入局部最优。与此同时，发现者本身的位置未能充分的利用起来，这样会导致麻雀未能仔细探查就掠过该位置，从而失去在该区域搜索的机会。所以为了解决这两个问题，进一步优化算法在发现食物阶段的搜索方式，将动态变化的权值 ω_t 加入到发现者的更新公式。在迭代的早期， ω_t 的值较大，发现者可以在更宽广的区域搜索，随着迭代的增加，惯性权值将非线性递减，会对发现者的贡献力量进行动态调整，会加强对局部位置的搜索。新的发现者位置更新公式为：

$$X_{i,j}^{T+1}=\begin{cases}\omega_t\cdot X_{i,j}^{t}\cdot\exp\left(-\dfrac{i}{\alpha*Iter_{\max}}\right), & R_2<ST\\ \omega_t\cdot\left(X_{i,j}^{t}+Q\cdot L\right), & R_2\geq ST\end{cases} \tag{6}$$

式中： $\omega_t=0.5\cdot\left(1-\left(\dfrac{2t}{Iter_{max}}-1\right)^3\right)$。

3.3. 自适应的选择变异策略

在麻雀算法迭代的后期，搜索个体会向一个和几个地方迅速靠拢，这就会导致算法容易陷入局部最优，无法再进一步探索更优的位置。为针对这一个问题，设计了一个自适应的选择变异策略。然而为了避免由于增加了变异策略从而导致算法的运行时间变长，所以定义了两个变异的条件，条件如下：

首先定义一个自适应线性变化的值 C，如果随机数 $r>C$，就认定符合变异的第一个条件。C 值与当

前迭代次数和最大迭代次数有关。当迭代早期的时候，并不希望太早的变异，仍然希望可以按照原来的规则迭代；到迭代后期时候，希望算法可以尽可能的变异。C 值得计算公式如下所示：

$$C = 1 - \frac{t}{Iter_{max}} \tag{7}$$

第二个条件就是经过两次迭代后，最优适应度都未能发生变化，就可以认定该算法已经陷入到了局部最优，便需要进行变异。同时满足以上两个条件就可以按照下面的规则变异：

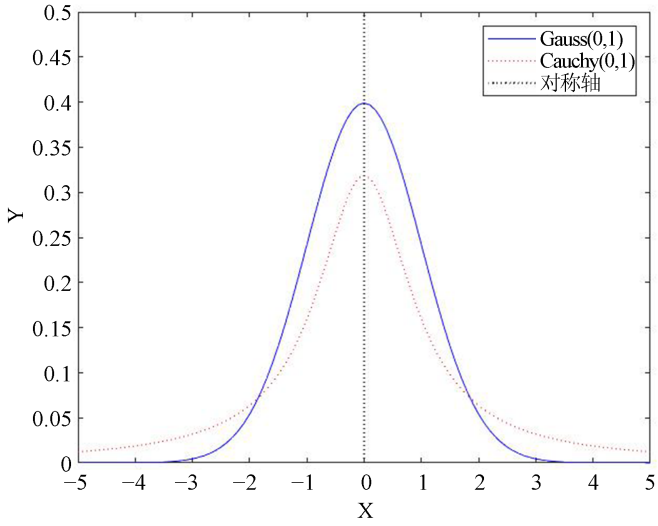


Figure 3. Probability density plots of $N(0,1)$ and $cauchy(0,1)$

图 3. $N(0,1)$ 和 $cauchy(0,1)$ 的概率密度图

t 分布有两个边界的特例分布，分别为高斯分布和柯西分布。图 3 就是标准正态分布和标准柯西分布的概率密度图，横坐标 X 为连续随机变量，纵坐标 Y 为概率密度。由图 3 中可知，标准正太分布水平方向小于标准柯西分布，但是在 0 的周围略大于标准柯西分布。若对麻雀进行高斯变异，主要是在个体附近的局部区域进行扰动，可以更仔细地搜索该区域。而标准柯西分布恰恰相反，经过柯西变异的麻雀更有概率产生一个远离 0 值得随机数，可以更为宽广的搜索区域，有利于跳出局部最优。所以将麻雀个体按照适应度从小到大排序后，设定适应度值在前一半的麻雀做高斯分布扰动，后一半的麻雀做柯西变异。最后变异的公式为：

$$X_t' = \begin{cases} X_t + N(0,1) \cdot X_t, & 0 < i < 0.5 \cdot pop \\ X_{best}^t + cauchy(0,1) \cdot X_t, & 0.5 \cdot pop < i < pop \end{cases} \tag{8}$$

式中：X_t 表示变异前麻雀的位置；X_t' 表示麻雀在变异后的位置；pop 是种群的个数；X_{best}^t 是指式(1)中发现者找到的适应度最优的位置；$N(0,1)$ 为标准正态分布；$cauchy(0,1)$ 为标准柯西分布。

3.4. 改进后算法的描述与流程

改进后的麻雀搜索算法 IASSA 流程如下：

1) 设定该算法的参数，包括种群的数量 pop，最大迭代次数 $Iter_{max}$，发现者，跟随着和预警者的比例，以及安全值 ST，和预警值 R_2。

2) 利用改进后的 Chebyshev 公式(4)结合公式(5)进行麻雀种群的初始化，计算种群的适应度值并排序。

3) 将公式(6)对原算法中的公式(1)进行替换，并依据事先设定好的参数，结合公式(6) (2) (3)进行种群位置的更新。

4) 根据公式(7)和是否两次结果一致的条件来判定是否进行公式(8)的麻雀变异。

5) 不断重复(2)~(4)的步骤，直至达到最优结果或最大的迭代次数为止。

改进后的麻雀搜索算法 IASSA 流程图如图 4 所示：

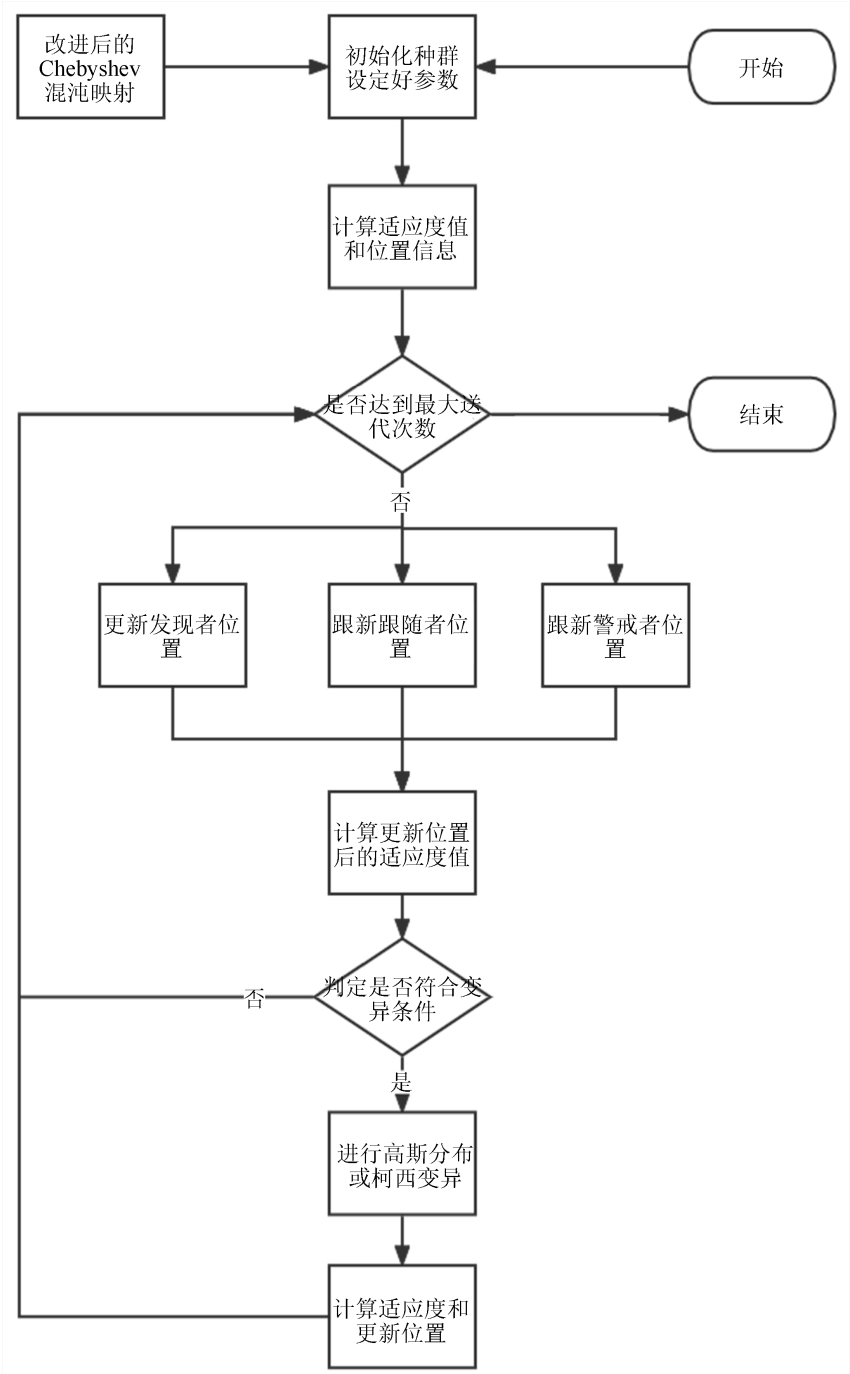


Figure 4. Algorithm flowchart
图 4. 算法流程图

4. 算法性能测试

为了体现该算法的有效性，于是用了目前较为新颖的几个算法进行测试对比，分别为灰狼算法(GWO)、鲸鱼优化算法(WOA)、以及原本的麻雀搜索算法(SSA)，本文也引进了两种改进后的麻雀搜索算法，分别为混合正弦余弦算法和 Lévy 飞行的麻雀算法中的 ISSA [13]算法和自适应 t 分布与黄金正弦改进的麻雀搜索算法及其应用[19]中的 t-GSSA 算法，为了体现添加变异策略的有效性，文中也添加了只进行 Chebyshev 混沌映射和惯性权值的 I-SSA 算法。

4.1. 参数的设置

仿真实验采用测试软件 MATLAB 2020a，在 Windows 10 64 bit 操作系统、AMD 锐龙 R5 3600、16GB 内存的计算机上实现。GWO、WOA、SSA、ISSA、t-GSSA 参数设置与原文献一致，最大迭代次数为 100，种群规模为 30，每种算法独立运行 30 次，对运行结果计算最优值、平均值和标准差。实验所使用算法中的参数设置如表 1 所示：

Table 1. Algorithm parameter settings
表 1. 算法参数设置

优化算法	参数设置
GWO	a 从 2 线性递减至 0， $r_1, r_2 \in [0,1]$
WOA	$b=1$
SSA	$PD=0.2, ST=0.8, SD=0.2$
ISSA	$PD=0.2, ST=0.8, SD=0.2, \xi=1.5$
t-GSSA	$PD=0.2, ST=0.8, SD=0.2, \omega_1=0.5, \omega_2=0.1$
I-SSA	$PD=0.2, ST=0.8, SD=0.2$
IASSA	$PD=0.2, ST=0.8, SD=0.2$

4.2. 基准测试函数

本文选取 10 个基准测试函数进行测试，表 2 详细展示了函数名称、函数公式、实验维度、定义域和最优值，其中 F_1~F_4 为多维单峰测试函数、F_5~F_8 为多维多峰测试函数、F_9~F_{10} 为固定维度测试函数。

Table 2. Test function information
表 2. 测试函数信息

函数名称	函数公式	维度	定义域	最优值
Sphere Function	$F_1(x)=\sum_{i=1}^{n} x_i^2$	30	[−100, 100]	0
Schwefel's Problem 2.22	$F_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	[−10, 10]	0
Schwefel's Problem 1.2	$F_3(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	30	[−100, 100]	0
Schwefel's Problem 2.21	$F_4(x)=\max_i\{\lvert x_i\rvert, 1\le i\le n\}$	30	[−100, 100]	0

Continued

Generalized Rastrigin's Function	$F_5(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	30	[−5.12, 5.12]	0
Ackley's Function	$F_6(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	30	[−32, 32]	0
Generalized Griewank's Function	$F_7(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\frac{x_i}{\sqrt{i}}+1$	30	[−600, 600]	0
Generalized Penalized Function	$F_8(x)=\frac{\pi}{n}\left\{10\sin^2(\pi y_i)+\sum_{i=1}^{n-1}(y_i-1)^2\cdot\left[1+10\sin^2(\pi y_{i+1})\right]+(y_n-1)^2\right\}+\sum_{i=1}^{n}u(x_i,10,100,4)$	30	[−50, 50]	0
Kowalik's Function	$F_9(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_i(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	4	[−5, 5]	3.07E−04
Branin Function	$F_{10}(x)=\left(x_2-\frac{5.1}{4\pi^2}x^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_i+10$	2	[−5, 5]	0.398

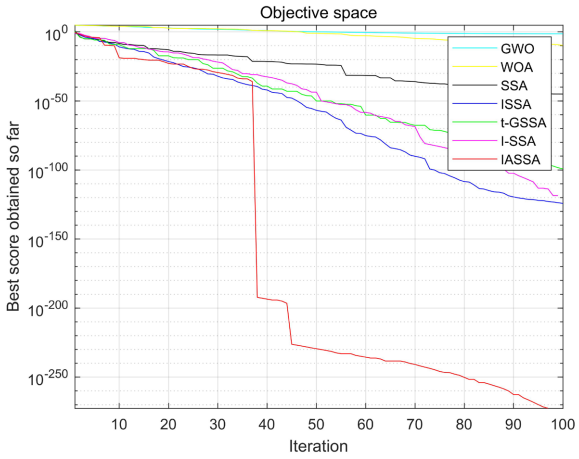
4.3. 测试结果和收敛曲线对比分析

Table 3. Test results
表 3. 测试结果

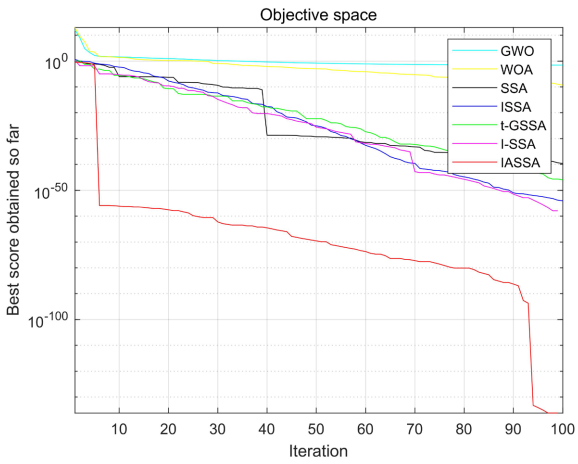
	算法	最优值	平均值	标准差		算法	最优值	平均值	标准差
	GWO	0.0043	0.016667	0.013437		GWO	0.0114	0.024993	0.009077
	WOA	1.27E−17	3.11E−11	8.55E−11		WOA	4.41E−10	6.49E−07	1.37E−06
	SSA	**0**	9.12E−35	4.91E−34		SSA	8.88E−16	8.88E−16	**0**
F_1	ISSA	2.8E−148	1.1E−109	5.9E−109	F_6	ISSA	8.88E−16	8.88E−16	**0**
	t-GSSA	8.7E−114	6.53E−83	3.52E−82		t-GSSA	8.88E−16	8.88E−16	**0**
	I-SSA	0	0	0		I-SSA	8.88E−16	8.88E−16	**0**
	IASSA	**0**	**0**	**0**		**IASSA**	**8.88E−16**	**8.88E−16**	**0**
	GWO	0.0128	0.026037	0.008376		GWO	0.0091	0.0707	0.063158
	WOA	4.89E−11	1.13E−08	2.39E−08		WOA	2E−15	0.042743	0.161268
	SSA	1.2E−107	1.55E−19	8.36E−19		SSA	**0**	**0**	**0**
F_2	ISSA	4.5E−71	4.36E−52	2.35E−51	F_7	ISSA	**0**	**0**	**0**
	t-GSSA	2.58E−59	1.56E−41	5.51E−41		t-GSSA	**0**	**0**	**0**
	I-SSA	0	0	0		I-SSA	**0**	**0**	**0**
	IASSA	**0**	**0**	**0**		**IASSA**	**0**	**0**	**0**
	GWO	23.5246	284.1185	282.3751		GWO	0.0841	0.506143	0.422182
	WOA	32235	90428.1	31447.77		WOA	0.0312	0.330823	0.904795
	SSA	**0**	6.09E−32	1.58E−31		SSA	3.78E−11	2.23E−06	5.57E−06
F_3	ISSA	2.7E−129	4.71E−94	2.53E−93	F_8	ISSA	1.1E−09	4.55E−06	9.14E−06
	t-GSSA	1.13E−85	2E−50	7.02E−50		t-GSSA	5.29E−08	2.38E−06	2.8E−06
	I-SSA	0	0	0		I-SSA	8.00E−11	1.26E−06	1.98E−06
	IASSA	**0**	**0**	**0**		**IASSA**	**4.44E−11**	**7.24E−07**	**1.27E−06**

Continued

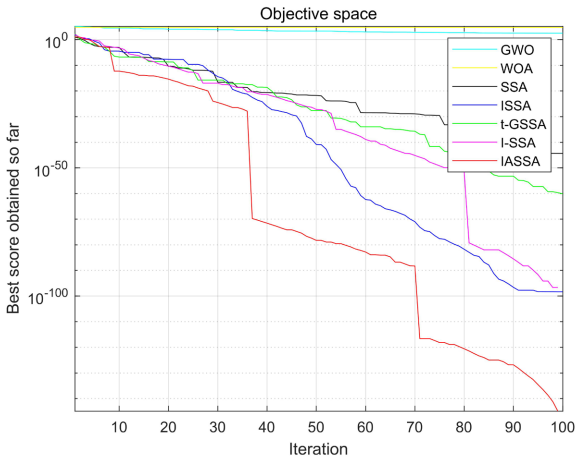
	GWO	0.6858	1.478837	0.553643		GWO	0.000444	0.005504	0.008322
	WOA	3.9407	60.05007	25.8615		WOA	0.000316	0.000752	0.000466
	SSA	**0**	5.18E−23	2.32E−22		SSA	0.000308	0.000325	6.98E−05
F_4	ISSA	1.53E−69	7.68E−55	4.13E−54	F_9	ISSA	0.000307	0.000313	1.12E−05
	t-GSSA	1.81E−60	1.51E−46	5.59E−46		t-GSSA	0.000308	0.000317	1.57E−05
	I-SSA	0	0	0		I-SSA	0.000308	0.000313	7.825E−06
	IASSA	**0**	**0**	**0**		**IASSA**	**0.000308**	**0.000317**	**1.26E−05**
	GWO	11.882	34.47297	14.81146		GWO	0.3979	0.39795	8.85E−05
	WOA	5.68E−14	4.515523	23.689		WOA	0.3979	0.398657	0.001902
	SSA	**0**	**0**	**0**		SSA	0.3979	0.3979	**0**
F_5	ISSA	**0**	**0**	**0**	F_{10}	ISSA	0.39789	0.39789	1.8E−05
	t-GSSA	**0**	**0**	**0**		t-GSSA	0.39789	0.39789	1.8E−05
	I-SSA	**0**	**0**	**0**		I-SSA	0.3979	0.3979	**0**
	IASSA	**0**	**0**	**0**		**IASSA**	**0.3979**	**0.3979**	**0**



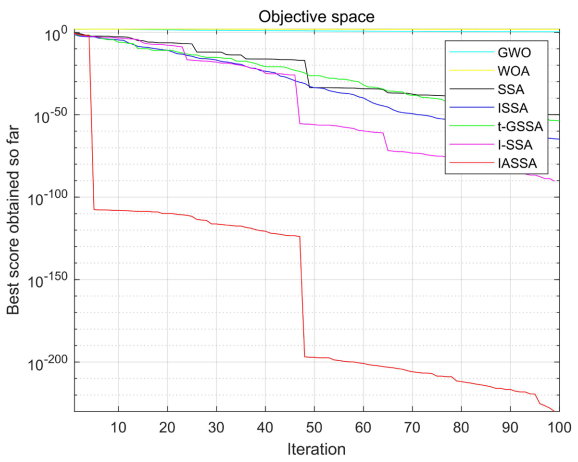
F1 测试函数



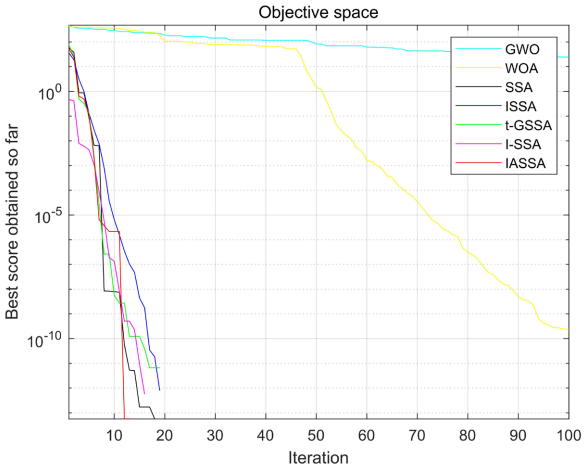
F2 测试函数



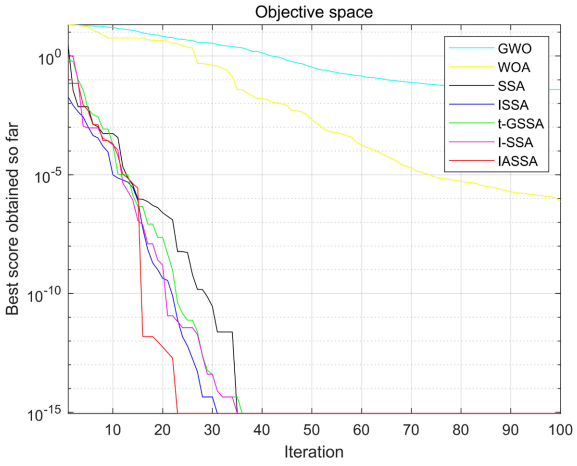
F3 测试函数



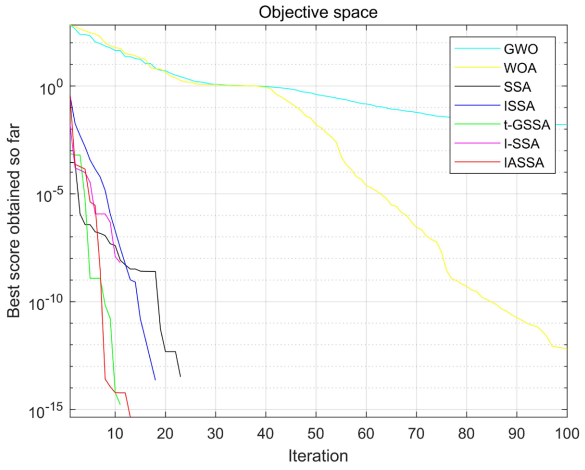
F4 测试函数



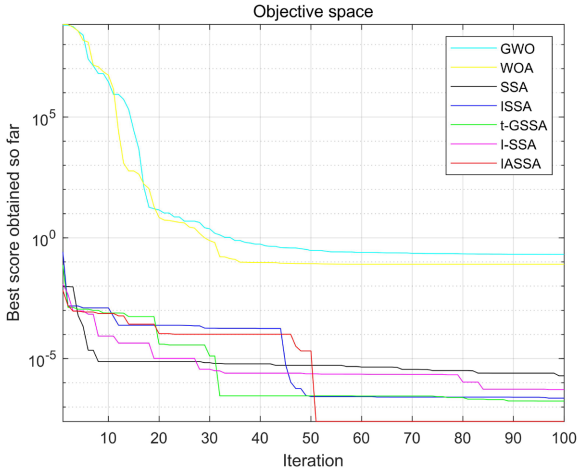
F5 测试函数



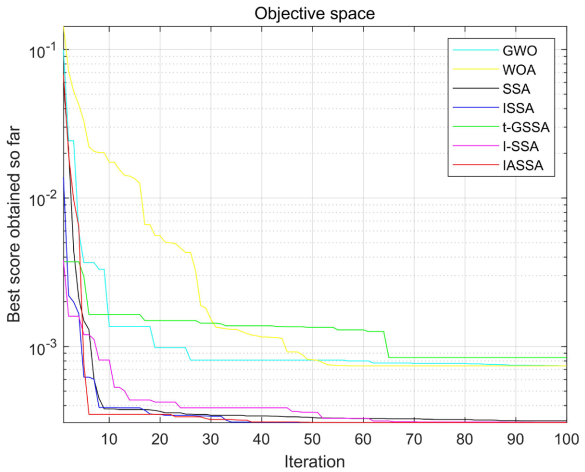
F6 测试函数



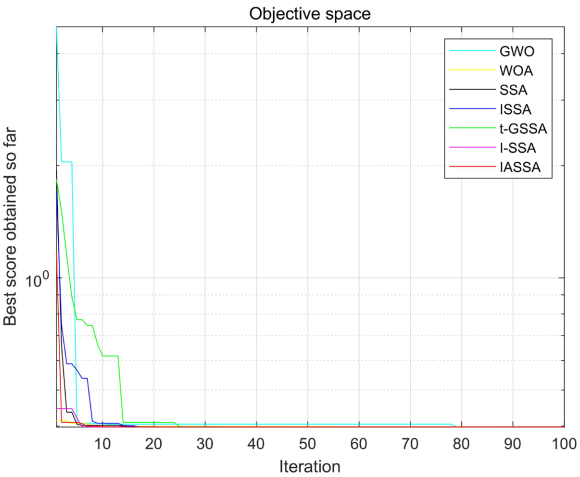
F7 测试函数



F8 测试函数



F9 测试函数



F10 测试函数

Figure 5. Convergence curve of test function
图 5. 测试函数收敛曲线图

从表 3 中可以得到以下信息：首先，F_1~F_4 是多维单峰测试函数，从测试结果上来看，IASSA 均可以在规定的迭代次数内找到最优值，而其它优化算法在迭代次数内仅能靠近最优值，并不能达到，表明了 IASSA 在多维单峰测试函数上的寻优能力明显优于其余的几个算法，同时标准差为 0，也体现了 IASSA 算法在多维单峰测试函数上的稳定性；其次，F_5~F_8 是多维多峰测试函数，在 F_5~F_7 测试函数中，除 GWO 和 WOA 算法外，SSA、ISSA、t-GSSA、IASSA 均可以达到测试函数的最优值，说明 IASSA 并没有下降 SSA 算法在多峰多维函数上的精确度，对于 F_8，IASSA 的最优值，平均值，标准差均小于其他算法，最接近最优值 0，与算法 I-SSA 对比，IASSA 跳出了局部最优，提升了寻优能力。F_9~F_{10} 为固定维多峰测试函数，从上表中可以看出，几个算法皆可以找到测试函数的最优值，但是 IASSA 与算法 I-SSA 对比也可以发现，添加变异策略使得求解的结果更加稳定。

从图 5 的 10 张收敛图上来看，在对比的几个算法中，IASSA 的收敛速度与收敛精度无疑是最好的。

综合测试函数的结果和收敛图来看，本文所提出的算法 IASSA，不论是在收敛精度、收敛速度上，还是从求解的稳定性出发，对于其他的优化算法都具有一定的优势，远远高于原 SSA 算法，这也从侧面说明改进策略的有效性。

4.4. Wilcoxon 秩和检验

虽然从测试结果上看，本文提出的 IASSA 算法相较于其它几个对比算法有着较为明显的提升，但未能从统计检验的角度来证明是否存在显著性区别。Wilcoxon 秩和检验可以检验算法间是否存在显著性差异。设原假设 H_0：两种算法之间没有显著差异，备择假设 H_1：两种算法之间有明显差异，置信度为 $\alpha = 0.05$。本文将 IASSA 算法与其他的 5 个算法进行对比分析，得到 p 值在下表 4 所示。如果 p 值小于 0.05，就拒绝原假设，说明两者之间有着显著性的差异，反之则认为两者之间没有显著性差异。NA 指两种算法皆达到最优，无法进行对比。

Table 4. p-values of Wilcoxon rank sum test
表 4. Wilcoxon 秩和检验的 p 值

函数	GWO	WOA	SSA	ISSA	t-GSSA
	P 值	P 值	P 值	P 值	P 值
F_1	1.2098e−12	1.2118e−12	4.5736e−12	1.2118e−12	1.2118e−12
F_2	1.2108e−12	1.2118e−12	1.2118e−12	1.2118e−12	1.2118e−12
F_3	1.2118e−12	1.2118e−12	4.5736e−12	1.2118e−12	1.2118e−12
F_4	1.2118e−12	1.2118e−12	4.5736e−12	1.2118e−12	1.2118e−12
F_5	1.2118e−12	1.2098e−12	NA	NA	NA
F_6	1.2098e−12	1.2118e−12	NA	NA	NA
F_7	1.2118e−12	1.2118e−12	NA	NA	NA
F_8	3.0199e−11	3.0199e−11	0.6100	0.0905	7.1988e−05
F_9	2.9137e−11	1.3273e−10	0.1580	0.1023	0.3403
F_{10}	6.4629e−04	3.1322e−04	NA	1.2160e−12	1.1651e−13

从上表中可以发现，IASSA 对 GWO 和 WOA 算法的 P 值皆小于 0.05，可以认为 IASSA 显著优于它们；从 SSA 算法的 P 值中可以发现除了 F_8 和 F_9 没有显著差异，其余都是显著的；最后从 ISSA 和 t-GSSA 上来看，函数 F_9 没有显著差异，其余也都是显著的。就函数 F_9 本身上分析，ISSA 和 t-GSSA 的寻优结果都很靠近理论最优值，没有太大的提升空间。

综上，可以认为，本文提出的 IASSA 算法在寻优能力上相较于几个对比算法有着显著的提升。

4.5. IASSA 时间复杂度分析

在算法的实际应用中，其复杂程度是一个无法忽视的因素，如果效率太低，就会大大降低该算法的价值。因此对本文提出的 IASSA 进行时间复杂度分析。

在原本的 SSA 算法中，先设定基本的要素，其中种群规模为 Num，发现者数量 $pNum$，跟随着的数量为 $sNum$，预警者的数量为 $yNum$，空间维数为 D，最大的迭代次数为 $Iter_{max}$，计算适应度函数所需的时间为 $f(D)$，根据文献[10]可知，SSA 的总时间复杂度为：$T=O(D+j(D))$。

而在 IASSA 算法中，首先设定进行参数初始化的时间为 t_1，生成 Chebyshev 序列的时间为 t_2，按照公式(5)将此映射序列带入种群初始化的时间为 t_3，所以在初始阶段所需的时间为 $T_1=O(t_1+Numf(D)+D(t_2+t_3))$。在发现者阶段，先计算权值 ω_t 所需时间 t_4，后进行发现者位置更新公式(6)所需时间 t_5，所以发现者阶段的时间为：$T_2=O(pNum(t_4+t_5)D)$。跟随着,预警者阶段与原 SSA 一致，进行公式(2) (3)更新，所需时间分别为 t_6 和 t_7，所以跟随着阶段的时间为：$T_3=O(sNumt_6D)$，预警者阶段所需时间为：$T_4=O(yNumt_7D)$。在变异阶段中，计算适应度值 C 所需时间为 t_8，将麻雀进行适应度排序的时间为 t_9，按照公式(8)进行高斯柯西变异所需时间为 t_{10}。所以变异阶段的总时间为 $T_5=O(Num(t_8+t_9+t_{10})D)$。综上所述，IASSA 的时间复杂度为 $T'=T_1+Iter_{max}(T_2+T_3+T_4+T_5)=O(D+j(D))$，得到 $T=T'$，由此可见，IASSA 并没有改变原 SSA 的复杂程度。

此外，为了证明理论的有效性，本文还通过实验将两种算法的实际运行时间放在一起对比验证。表 5 中记录了 10 个测试函数每一次运行的最短、最长、平均运行时长。

Table 5. Running time table
表 5. 运行时长表

函数	最短运行时长/s		最大运行时长/s		平均运行时长/s	
	SSA	IASSA	SSA	IASSA	SSA	IASSA
F_1	0.016738	**0.017476**	0.019707	**0.05467**	0.017376	**0.020554**
F_2	0.017405	**0.01832**	0.021827	**0.025623**	0.017914	**0.019392**
F_3	0.048641	**0.050025**	0.071564	**0.053449**	0.050383	**0.050813**
F_4	0.016708	**0.017505**	0.021089	**0.022164**	0.017236	**0.01806**
F_5	0.018109	**0.018931**	0.023578	**0.027238**	0.018722	**0.022275**
F_6	0.018821	**0.01947**	0.02528	**0.027764**	0.019507	**0.021608**
F_7	0.021092	**0.021835**	0.026777	**0.031721**	0.021638	**0.024718**
F_8	0.056178	**0.058165**	0.077235	**0.064792**	0.059228	**0.059654**
F_9	0.016274	**0.016491**	0.020706	**0.025097**	0.016755	**0.017359**
F_{10}	0.015347	**0.015644**	0.024303	**0.024915**	0.016152	**0.017925**

从表中可以明显看出，两者的平均时间非常相近。IASSA 在多维单峰测试函数 F_1~F_4 上平均时长比标准 SSA 略微高出一点，而在多维多峰测试函数 F_5~F_8 上最短运行时间基本相同，在固定维度测试函数 F_9~F_{10} 的平均时长上，两者的也是非常接近的。由此可见，表中的数据也从侧面印证了两者的时间复杂度是相同的。

5. 结论

为了改善麻雀搜索算法容易陷入局部最优，收敛精度低等缺陷，本文提出了基于惯性权值和自适应变异改进的麻雀搜索算法(IASSA)，从种群初始化、发现者的更新策略和陷入最优后的变异策略三个方面对原算法进行了改进。从仿真实验结果上来看，效果较原算法有较大的提升，体现了 IASSA 算法的优良性。下一步的研究方向可以从两个方面入手，一是寻优结果较理论值还有一定的距离，可以继续改进算法以提高算法的精度与速率。二是由于 IASSA 有着较好的收敛精度和收敛速度，可以用来解决复杂工程的优化问题，具有良好的应用价值。

参考文献

[1] Tilahun, S.L. and Ngnotchouye, J.M.T. (2017) Firefly Algorithm for Discrete Optimization Problems: A Survey. *KSCE Journal of Civil Engineering*, **21**, 535-545. https://doi.org/10.1007/s12205-017-1501-1

[2] Kennedy, J. and Eberhart, R. (1995) Particle Swarm Optimization. *Proceedings of the IEEE International Conference on Neural Networks*, **4**, 1942-1948.

[3] Mirjalili, S. and Lewis, A. (2014) Grey Wolf Optimizer. *Advances in Engineering Software*, **69**, 46-61. https://doi.org/10.1016/j.advengsoft.2013.12.007

[4] Mirjalili, S. and Lewis A. (2016) The Whale Optimization Algorithm. *Advances in Engineering Software*, **95**, 51-67. https://doi.org/10.1016/j.advengsoft.2016.01.008

[5] Mirjalili, S. (2016) SCA: A Sine Cosine Algorithm for Solving Optimization Problems. *Knowledge-Based Systems*, **96**, 120-133. https://doi.org/10.1016/j.knosys.2015.12.022

[6] Xue, J.K. and Shen, B. (2020) A Novel Swarm Intelligence Optimization Approach: Sparrow search Algorithm. *Systems Science & Control Engineering*, **8**, 22-34. https://doi.org/10.1080/21642583.2019.1708830

[7] 马卫, 朱娴, 李微微. 基于莱维飞行麻雀搜索优化的 Otsu 多阈值图像分割[J]. 计算机时代, 2023(4): 77-85.

[8] 董维振, 陈燕. 基于改进麻雀搜索算法的带钢面标识别方法[J]. 现代制造工程, 2023(2): 84-91+69.

[9] 刘美辰. 基于高光谱技术的牛奶蛋白质含量预测与分析[D]: [硕士学位论文]. 呼和浩特: 内蒙古农业大学, 2022.

[10] 唐延强, 李成海, 宋亚飞, 陈晨, 曹波. 自适应变异麻雀搜索优化算法[J]. 北京航空航天大学学报, 2023, 49(3): 681-692.

[11] 陈功, 曾国辉, 黄勃, 刘瑾. 螺旋探索与自适应混合变异的麻雀搜索算法[J]. 小型微型计算机系统, 44(4): 779-786.

[12] 汤安迪, 韩统, 徐登武, 谢磊. 基于等级制度和布朗运动的混沌麻雀搜索算法[J]. 空军工程大学学报(自然科学版), 2021, 22(3): 96-103.

[13] 毛清华, 张强, 毛承成, 柏嘉旋. 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(6): 1086-1091.

[14] 柳长安, 冯雪菱, 孙长浩, 赵丽娟. 基于改进麻雀算法的最大二维熵分割方法[J]. 激光技术, 2022, 46(2): 274-282.

[15] 王辉, 童楠, 符强. 多策略融合的麻雀搜索算法[J]. 计算机系统应用, 2023, 32(6): 159-165.

[16] 胡树斌, 魏霖静. 基于混合策略改进的麻雀搜索算法[J]. 计算机技术与发展, 2023, 33(4): 146-153+160.

[17] 蓝婷婷, 游林, 翁昕耀. 基于 Chebyshev 混沌映射与模糊金库的指静脉安全认证方案[J]. 通信技术, 2019, 52(6): 1469-1476.

[18] 蒋东华, 朱礼亚, 沈子懿, 王兴元, 陈颖频. 结合二维压缩感知和混沌映射的双图像视觉安全加密算法[J]. 西安交通大学学报, 2022, 56(2): 139-148.

[19] 张伟康, 刘升. 自适应 t 分布与黄金正弦改进的麻雀搜索算法及其应用[J]. 微电子学与计算机, 2022, 39(3): 17-24.