DOI:10.19651/j.cnki.emt.2209151

基于正交对立学习的改进麻雀搜索算法*

王天雷[1,2] 张绮媚[3] 李俊辉[1] 周 京[1] 刘人菊[1] 谭南林[2]
(1.五邑大学智能制造学部 江门 529020;2.北京交通大学机械与电子控制工程学院 北京 100044;
3.五邑大学数学与计算科学学院 江门 529020)

摘 要:针对麻雀搜索算法种群多样性少,局部搜索能力弱的问题,本文提出了基于正交对立学习的改进型麻雀搜索算法(OOLSSA)。首先,在算法中引入正态变异算子,丰富算法种群多样性;其次,利用对立学习策略,增强算法跳出局部最优的能力;然后,在加入者更新之后引入正交对立学习机制,加快算法的收敛速度;最后,基于 15 个基准测试函数与 6 个传统优化算法和 2 个改进型算法进行仿真实验、非参数 Friedman 检验以及算法平衡能力进行分析,评估 OOLSSA 算法寻优性能。仿真结果证明,OOLSSA 与其余 8 种算法相比,算法的探索开发能力以及收敛速度都表现良好。

关键词:麻雀搜索算法;正交学习;对立学习;正态变异算子
中图分类号:TP301.6 **文献标识码**:A **国家标准学科分类代码**:510.40

Improved sparrow search algorithm based on orthogonal-opposition-based learning

Wang Tianlei[1,2] Zhang Qimei[3] Li Junhui[1] Zhou Jing[1] Liu Renju[1] Tan Nanlin[2]
(1. Intelligent Manufacturing Division, Wuyi University, Jiangmen 529020, China;
2. School of Mechanical, Electronic and Control Engineering, Beijing Jiaotong University, Beijing 100044, China;
3. School of Mathematics and Computational Science, Wuyi University, Jiangmen 529020, China)

Abstract: To solve the problem of low population diversity and weak exploitation of sparrow search algorithm, an improved sparrow search algorithm based on orthogonal-opposition-based learning (OOLSSA) is proposed in this paper. First, a normal mutation operator is used to enrich the diversity of algorithm population. Second, the opposition-based learning is used to enhance the ability of the algorithm to jump out of local optimum. Then, orthogonal-opposition-based learning is introduced after the update of the scrounger position to accelerate the convergence of the algorithm. Finally, performance test based on fifteen benchmark test functions, non-parametric Friedman test and balance analysis of algorithms shows that compared with six traditional optimization algorithms and two improved algorithms, OOLSSA has better searching performance on exploration and exploitation ability and convergence speed.

Keywords: sparrow search algorithm;orthogonal learning;opposition-based learning;normal mutation operator

0 引 言

群智能优化算法是一种受生物启发,模拟生物种群在进行觅食、筑巢等行为规律,在搜索空间内寻找最优解的一种算法。学者们通过模拟各种生物的行为规律,提出了一系列的群智能优化算法,其中包括:粒子群算法(particle swarm algorithm, PSO)[1]、果蝇优化算法(fruit fly optimization algorithm, FOA)[2]、鲸鱼优化算法(whale optimization algorithm, WOA)[3]、灰狼优化算法(grey wolf optimization, GWO)[4]和麻雀搜索算法(sparrow search algorithm, SSA)[5]等。其中,麻雀搜索算法是在 2020 年由 Xue 等提出的,属于一种新型的群智能优化算法。该算法对比于其他算法,具有求解效率高、稳定性好等优点。但该算法的种群多样性会随着迭代次数的增加而减

收稿日期:2022-03-03
*基金项目:国家自然科学基金(51505154,51437005)、2018 广东省教学质量工程与教改项目(GDJX2019012)、2020 年江门市科技计划项目(2020JC01035)、2019 年江门市科技计划项目(2019JC01005)资助

小，容易陷入局部最优。

针对麻雀搜索算法种群多样性少，存在容易陷入局部最优等问题，众多学者提出了不同的改进方法。从初始化角度出发，吕鑫等[6]通过改进 Tent 混沌序列并应用于种群的初始化，从而提高算法初始解的质量；尹德鑫等[7]用反向学习策略初始化种群，增强种群的遍历性。段玉先等[8]将 Sobol 序列引入初始化阶段，确保种群的多样性。从改进算法搜索方式角度出发，Li 等[9]设计了动态自适应 t 分布变异算子，增强了算法抗停滞能力；Jiang 等[10]利用随机游走等方法，帮助算法有效跳出局部最优；毛清华等[11]引入了莱维飞行策略，增强算法的局部逃逸能力；Ouyang 等[12]设计了自适应螺旋飞行策略，降低了算法陷入局部最优的概率；Tang 等[13]引入了遗传算法中的交叉和变异思想，弥补算法易陷入局部最优的缺陷。

以上学者使用多种策略应用在麻雀搜索算法上，一定程度上改善了算法的寻优能力，丰富了种群的多样性。而对于新提出的麻雀搜索算法而言，其寻优性能仍有较大提升空间。考虑到正态变异扰动与对立学习能够帮助算法逃离局部最优，以及正交对立学习策略能够加快算法收敛速度等特点，提出了基于正交对立学习的改进麻雀搜索算法(OOLSSA)。首先，在加入者公式中引入正态变异算子，扩大搜索范围，提高群体多样性；在侦查公式中使用对立学习策略避免算法陷入局部最优；利用正交对立学习策略，让加入者的当前解与其对立解按正交表进行正交并取最好的正交候选解进行位置更新，加快算法的收敛速度；最后，对 15 个基准测试函数进行仿真实验、非参数 Friedman 检验以及探索开发平衡能力分析，评估来验证 OOLSSA 有效性与可行性。

1　基本麻雀搜索算法

麻雀搜索算法的搜索原理来源于麻雀的捕食与反捕食行为。该算法中，麻雀被分为发现者和加入者。发现者是适应度值排名靠前的个体，其他个体则为加入者。当麻雀发现危险时，当前个体需要放弃食物并逃离到其他地方。

发现者数量一般为种群的 10%～20%，他们占据较好的位置，负责为指引其他麻雀的搜索方向，发现者位置更新公式如式(1)所示。

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

式(1)中：$\boldsymbol{X}_{i,j}^{t}$ 代表上一代的麻雀个体位置，i 表示当前第 i 个麻雀，$iter_{\max}$ 代表最大迭代次数，Q 代表服从 $N(0,1)$ 的随机数，$\boldsymbol{L}$ 代表元素全为 1 的 $1\times D$ 维的向量。$R_2\in[0,1]$，$ST\in[0.5,1]$，分别代表预警值和安全值。

加入者会根据发现者的位置进行位置更新，其公式如式(2)所示。

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

式(2)中：$\boldsymbol{X}_{worstj}^{t}$ 代表上一代全局最差解位置，$\boldsymbol{X}_{bestj}^{t+1}$ 代表当代发现者中的最优解位置，$\boldsymbol{A}$ 是一个由 1，−1 组成的 $1\times D$ 维的向量，$\boldsymbol{A}^{-}=\boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\cdot\boldsymbol{A}^{\mathrm{T}})^{-1}$。当 $i>\dfrac{n}{2}$ 时，代表适应度较差的饥饿个体要往其他地方进行觅食，而其他个体则往发现者最优位置附近进行觅食。

侦查预警的麻雀一般占种群总数的 10%～20%，若当前麻雀处于安全位置时，需要飞往边缘位置，若当前麻雀位置不是最好的，则需要飞往安全位置，其更新公式如式(3)所示。

$$\boldsymbol{X}_{i,j}^{t+1}=\begin{cases}\boldsymbol{X}_{bestj}^{t}+\beta\cdot\left|\boldsymbol{X}_{i,j}^{t}-\boldsymbol{X}_{bestj}^{t}\right|, & f_i>f_g\\ \boldsymbol{X}_{i,j}^{t}+K\cdot\left(\dfrac{\left|\boldsymbol{X}_{i,j}^{t}-\boldsymbol{X}_{worstj}^{t}\right|}{(f_i-f_w)+\varepsilon}\right), & f_i=f_g\end{cases}\tag{3}$$

式(3)中：β 表示服从 $N(0,1)$ 的随机数，$K\in(-1,1)$，f_i 表示当前个体适应度值，f_g 表示当前最优个体适应度值，f_w 表示当前全局最差适应度值，ε 表示一个接近于 0 的数，能够避免分母出现为 0 的情况。

基本麻雀搜索算法 SSA 的流程如图 1 所示。

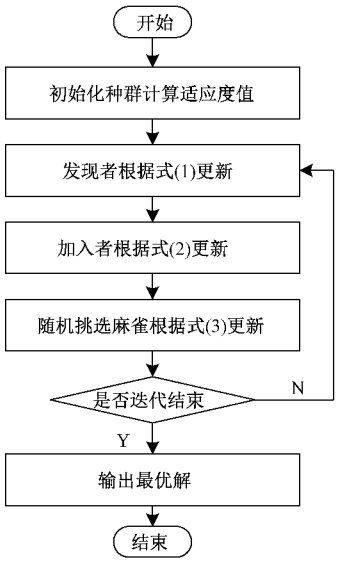


图 1　SSA 算法流程

2　改进麻雀搜索算法

2.1　正态变异扰动

在其原始加入者的位置更新公式中，直接让非饥饿的个体一开始从发现者最优位置附近进行更新，这样会导致种群的多样性单一，增加种群陷入局部最优的可能性。因此，这里引入正态变异算子对式(2)中的最优位置进行扰动，让非饥饿的加入者向发现者最优位置的变异解学习，这样能够增大种群多样性与跳出局部最优的能力。其中，正态变异公式为：$\boldsymbol{X}^{t}=\boldsymbol{X}^{t}+\alpha\cdot\boldsymbol{X}^{t}$。为了避免仅加值的方法

影响算法性能,在扰动公式中引入方向因子 F,改进公式如式(4)~(5)所示。

$$F=\begin{cases}-1, & r<0.5\\ 1,\text{其他}\end{cases} \tag{4}$$

$$\boldsymbol{X}_{Rbestj}^{t}=\boldsymbol{X}_{bestj}^{t}+F\cdot\alpha\cdot\boldsymbol{X}_{bestj}^{t} \tag{5}$$

式(4)~(5)中:r 是取值范围在[0,1]的随机数,α 是服从 $N(0,1)$的随机数。加入者更新公式改进如式(6)所示。

$$\boldsymbol{X}_{i,j}^{t-1}=\begin{cases}Q\cdot\exp\left(\dfrac{\boldsymbol{X}_{worstj}^{t}-\boldsymbol{X}_{i,j}^{t}}{i^{2}}\right), & i>\dfrac{n}{2}\\ \boldsymbol{X}_{Rbestj}^{t+1}+\left|\boldsymbol{X}_{i,j}^{t}-\boldsymbol{X}_{Rbestj}^{t+1}\right|\cdot\boldsymbol{A}^{+}\cdot\boldsymbol{L}, & \text{其他}\end{cases} \tag{6}$$

式(6)中:$\boldsymbol{X}_{Rbestj}^{t}$ 是当前最优发现者的正态变异扰动解。

2.2 对立学习

对立学习是一种常用的跳出局部最优解位置的策略。在原始麻雀搜索算法的侦查时,最优位置的麻雀往最差解靠拢,其他位置麻雀往最优值靠拢。虽然往最差解位置搜索,一定程度上能够避免陷入局部最优,但这样并不利于种群的收敛。而对立学习不仅能帮助个体快速逃离当前位置,而且对立位置相比于当前最差位置,其适应度值更有可能比于当前位置更优,因此,在侦查部分的位置更新公式引入对立学习策略,其改进公式如式(7):

$$\boldsymbol{X}_{i,j}^{t-1}=\begin{cases}\boldsymbol{X}_{bestj}^{t}+\beta\cdot\left|\boldsymbol{X}_{i,j}^{t}-\boldsymbol{X}_{bestj}^{t}\right|, & f_i>f_g\\ lb+ub-\beta\cdot\boldsymbol{X}_{i,j}^{t},\text{其他}\end{cases} \tag{7}$$

式(7)中:β 是服从 $N(0,1)$的随机数,lb 与 ub 分别为搜索空间的下界和上界。

2.3 正交对立学习

正交对立学习策略是利用当前解与对立解,通过正交实验设计,以较少的实验次数找到不同因素的水平最佳组合的一种方法。目前相关研究方面,阎大海等[14]提出了基于正交设计的反向学习策略,并应用在差分进化算法上,提高了算法的鲁棒性;周凌云等[15]基于萤火虫算法,引入了正交重心反向学习策略,增强了算法求解复杂问题的能力;基于此,本文融合对立学习与正交学习,提出正交对立学习策略,对麻雀搜索算法进行改进。该策略的基本思想是:利用当前位置与其对立位置,根据正交表构建正交候选解,接着对各候选解进行评估,最终取出其中最佳的正交组合。通过这种方式,充分利用个体和对立个体中各维度的信息并找到最佳组合,其流程如图 2。

正交候选解的构建过程用到了正交表,正交表被定义为 $\boldsymbol{L}_M(Q^K)$,$\boldsymbol{L}$ 为正交矩阵符号,M 为正交矩阵行数,Q 为水平,K 为因素。其中,K 对应正交矩阵的列数,是问题的维度,而 M 对应正交矩阵的行数,也是试验解的个数。下面以三因素二水平为例,其正交表如式(8):

$$\boldsymbol{L}_4(2^3)=\begin{bmatrix}1&1&1\\1&2&2\\2&1&2\\2&2&1\end{bmatrix} \tag{8}$$

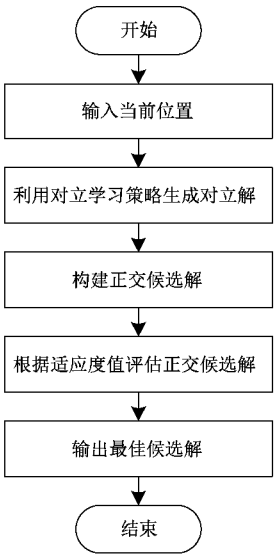


图 2 正交对立学习流程

在正交表中,同一列的 1 和 2 分别代表第一个水平和第二个水平的在该列的维度信息。矩阵中的第一行是当前个体本身,其他行是两个水平不同维度之间的正交组合,式(8)中只需要进行 4 次评估,即可找到当前解与对立解的最佳正交组合。而针对本文的正交对立学习策略,则需要构建 D 因素二水平的正交表,其中 D 为种群维度。随着维数的增多,试验评估的次数也会增加,但算法的收敛性能也会更好。

由于正交表包括个体本身位置,每一个候选解均有相同的概率成为最佳正交解,存在经过 M 次的评估后,仍然存在输出最优解为个体本身的可能性。假设种群中每一个个体都进行正交学习,这样固然能够更好的加快算法收敛速度,但这样同时也会大大增加实验评估的次数,并不适用于解决实际问题。因此,本文正交对立学习策略仅应用在加入者位置更新部分。在加入者按照式(6)更新后,再利用式(7)中的对立公式得到其对立解,将以上两个位置按照 D 因素二水平的正交表构建正交候选解,经过计算适应度值评估后,将最佳正交解候选解作为当前加入者的最终更新位置。通过这种方法,能够提高算法收敛速度与精度的同时避免算法陷入局部最优的问题。

基于正交对立学习的改进麻雀搜索算法 OOLSSA 的伪代码如算法 1 所示,流程如图 3 所示。

3 实验与分析

3.1 基准测试函数

为了检验改进算法的性能,分析算法的改进策略对收敛速度和搜索精度的影响,本节用 15 个基准测试函数 F1~F15 来验证 OOLSSA 的有效性。15 个基准测试函数中,包括用于分析算法的开发能力的 7 个高维单峰测试函数 F1~F7,用于测试算法跳出局部最优能力的 5 个高维多峰测试函数 F8~F12,和用于检验算法稳定性的 3 个低维多峰测试函数 F13~F15。所使用的测试函数均列于表 1。

算法 1:OOLSSA 算法伪代码

```
输入:目标函数
输出:全局最优位置和适应度值
随机初始化种群 N
评估个体适应度值并排序
while 未达到最大迭代次数
  for i=1:N
    发现者根据式(1)更新
    加入者根据式(6)更新
    加入者进行正交对立学习
  end for
  随机选取部分个体根据侦查公式(7)更新
  评估适应度值并排序
end while
输出最优位置及其适应度值
```

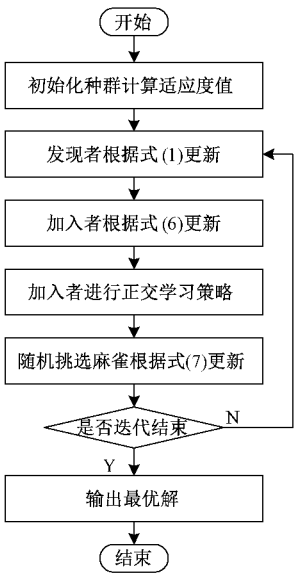


图 3　OOLSSA 算法流程图

仿真环境均在 Windows 10 环境下的 Matlab R2018a 中进行编程,仿真实验在 Inter(R) CoreTM i7-8565U CPU@ 1.80 GHz 四核处理器、64 GB 内存上进行。

3.2　与传统算法对比

下面将选取麻雀搜索算法(SSA),粒子群算法(PSO),布谷鸟算法(CS)[16]、多元宇宙算法(MVO)[17]、差分进化算法(DE)[18]、和声算法(HS)[19]作为对比算法,与改进麻雀搜索算法(OOLSSA)进行数值实验。上述对比的算法均是智能优化算法中的典型算法,常应用于工程优化问题中[20-22]。为确保各种优化算法在初始化时具有相同的搜索能力,种群规模统一为 30,最大迭代次数取 1 000 次。算法的具体参数设置如表 2 所示。

为了避免偶然误差,每个算法在各基准测试函数中均独立运行 30 次,表 3 依次列出了 OOLSSA、SSA、PSO、CS、MVO、DE、HS 7 个算法在 15 个基准测试函数上进行 30 次独立运行实验的平均结果。

表 1　基准测试函数

类型		名称	取值范围	理论最优值
高维单峰	F1	Sphere	$[-100,100]^{30}$	0
	F2	Schwefel 2.22	$[-10,10]^{30}$	0
	F3	Schwefel 1.2	$[-100,100]^{30}$	0
	F4	Schwefel 2.21	$[-100,100]^{30}$	0
	F5	Rosenbrock	$[-30,30]^{30}$	0
	F6	Step	$[-100,100]^{30}$	0
	F7	Noise	$[-1.28,1.28]^{30}$	0
高维多峰	F8	Rastrigin	$[-5.12,5.12]^{30}$	0
	F9	Ackley	$[-32,32]^{30}$	0
	F10	Griewank	$[-600,600]^{30}$	0
	F11	Penalized 1	$[-50,50]^{30}$	0
	F12	Penalized 2	$[-50,50]^{30}$	0
低维多峰	F13	Shekel 1	$[0,10]^{4}$	−10.153 2
	F14	Shekel 2	$[0,10]^{4}$	−10.402 8
	F15	Shekel 3	$[0,10]^{4}$	−10.536 3

表 2　各算法参数设置

优化算法	参数设置
OOLSSA	$PD=20\%$, $R_2=0.8$, $SD=10\%$
SSA	$PD=20\%$, $R_2=0.8$, $SD=10\%$
PSO	$\omega_{max}=0.9$, $\omega_{min}=0.2$, $C_1=C_2=2$
CS	$\beta=1.5$, $P=0.25$
MVO	$WEP_{max}=1$, $WEP_{min}=0.2$
DE	$\beta_{max}=0.8$, $\beta_{min}=0.2$, $pCR=0.2$
HS	$PAR=0.1$, $FWDR=0.995$, $HMCR=0.9$

由表 3 结果可知,对于高维单峰测试函数 F1～F7,OOLSSA 的寻优效果均优于其他算法,且在 F5 和 F6 中,其所有指标均达到 0。高维多峰测试函数中,OOLSSA 与 SSA 在 F8～F11 中的寻优性能相同且优于其他算法,而在 F11 与 F12 中,本文所提算法的 3 个指标的寻优精度比 SSA 高出 20～30 多个数量级。在低维多峰测试函数中,仅有 OOLSSA 找到理论最优值,虽然其标准差不及部分传统算法,但是其平均值也是最接近于理论最优的。

数据表明,在高维单峰与高维多峰测试函数中,OOLSSA 的开发能力,跳出局部最优能力,稳定性明显优于其他算法;在低维多峰测试函数中,虽然本文所提算法稳定性表现一般,但其平均搜索结果最接近理论最优解。

为了反映 OOLSSA 的动态收敛性能,图 4 依次列出了各算法在 15 个基准测试函数的平均最优适应度收敛曲线图。为了保证对比的公平性,下图的实验与上表数据为同一次实验。其中横轴表示迭代次数,纵轴表示适应度值。当曲线随着迭代次数的增加不再变化,即表示该算法陷入

表 3 不同算法的寻优结果

测试函数	指标	OOLSSA	SSA	PSO	CS	MVO	DE	HS
F1	最优值	0	1.90×10^{71}	1.46×10^{8}	4.68×10^{3}	2.98×10^{1}	3.58×10^{12}	5.73×10^{1}
	平均值	1.54×10^{-44}	2.02×10^{-41}	1.64×10^{-8}	1.10×10^{-2}	3.20×10^{-1}	6.03×10^{-12}	6.39×10^{1}
	标准差	8.43×10^{-44}	1.11×10^{-40}	7.43×10^{-9}	6.45×10^{-3}	1.08×10^{-2}	1.31×10^{-12}	2.59×10^{0}
F2	最优值	0	4.27×10^{-43}	3.39×10^{-4}	7.22×10^{-2}	3.76×10^{-1}	3.53×10^{-8}	1.03×10^{0}
	平均值	9.71×10^{-24}	1.38×10^{-21}	3.41×10^{-4}	1.22×10^{-1}	3.86×10^{-1}	4.69×10^{-8}	1.11×10^{0}
	标准差	5.30×10^{-23}	7.58×10^{-21}	9.00×10^{-6}	4.07×10^{-2}	5.05×10^{-3}	5.79×10^{-9}	2.71×10^{-2}
F3	最优值	0	1.55×10^{55}	1.63×10^{1}	3.57×10^{3}	4.65×10^{1}	2.48×10^{4}	9.24×10^{3}
	平均值	1.00×10^{37}	1.04×10^{32}	1.77×10^{2}	5.53×10^{3}	4.67×10^{1}	4.05×10^{4}	1.27×10^{4}
	标准差	5.48×10^{-37}	5.71×10^{-32}	2.84×10^{2}	1.52×10^{3}	1.21×10^{-1}	8.85×10^{3}	1.33×10^{3}
F4	最优值	0	6.56×10^{-38}	6.46×10^{-1}	1.31×10^{1}	9.67×10^{-1}	1.98×10^{0}	1.31×10^{1}
	平均值	1.70×10^{-22}	8.16×10^{-21}	1.83×10^{0}	1.87×10^{1}	9.81×10^{-1}	2.46×10^{0}	1.35×10^{1}
	标准差	9.32×10^{-22}	4.47×10^{-20}	9.95×10^{-1}	3.82×10^{0}	6.74×10^{-3}	2.38×10^{-1}	2.09×10^{-1}
F5	最优值	0	3.47×10^{-7}	6.33×10^{1}	6.73×10^{1}	2.94×10^{2}	5.52×10^{1}	2.39×10^{3}
	平均值	0	4.03×10^{-7}	4.16×10^{2}	2.02×10^{2}	2.95×10^{2}	8.52×10^{1}	2.78×10^{3}
	标准差	0	1.37×10^{7}	9.80×10^{2}	1.50×10^{2}	2.20×10^{1}	2.20×10^{1}	1.53×10^{2}
F6	最优值	0	4.32×10^{9}	1.90×10^{8}	4.80×10^{3}	3.00×10^{1}	3.64×10^{12}	4.95×10^{1}
	平均值	0	4.47×10^{-9}	2.23×10^{-8}	1.04×10^{-2}	3.24×10^{-1}	6.24×10^{-12}	5.56×10^{1}
	标准差	0	2.44×10^{-10}	1.50×10^{-8}	4.70×10^{-3}	1.07×10^{-2}	1.40×10^{-12}	2.42×10^{0}
F7	最优值	8.46×10^{-5}	1.21×10^{-4}	7.17×10^{-2}	1.03×10^{-1}	1.95×10^{-2}	2.65×10^{-2}	1.03×10^{-1}
	平均值	4.89×10^{-3}	5.81×10^{-3}	2.51×10^{1}	2.54×10^{-1}	5.13×10^{-1}	5.83×10^{-2}	1.51×10^{-1}
	标准差	5.96×10^{-3}	7.07×10^{-3}	1.05×10^{2}	9.28×10^{-2}	2.81×10^{-1}	1.89×10^{-2}	2.12×10^{-2}
F8	最优值	0	0	5.13×10^{1}	6.77×10^{1}	1.13×10^{2}	5.89×10^{1}	8.15×10^{0}
	平均值	0	0	5.63×10^{1}	9.95×10^{1}	1.13×10^{2}	8.14×10^{1}	8.62×10^{0}
	标准差	0	0	1.18×10^{1}	2.05×10^{1}	5.73×10^{3}	1.06×10^{1}	2.09×10^{1}
F9	最优值	8.88×10^{-16}	8.88×10^{-16}	3.94×10^{-4}	3.63×10^{0}	1.21×10^{0}	4.48×10^{-7}	2.83×10^{0}
	平均值	8.88×10^{16}	8.88×10^{16}	3.95×10^{4}	9.19×10^{0}	1.22×10^{0}	5.76×10^{7}	2.93×10^{0}
	标准差	0	0	3.65×10^{-6}	6.71×10^{0}	2.53×10^{-3}	6.56×10^{-8}	3.86×10^{-2}
F10	最优值	0	0	9.53×10^{-3}	2.84×10^{-2}	5.97×10^{-1}	1.14×10^{-10}	1.46×10^{0}
	平均值	0	0	9.53×10^{-3}	6.49×10^{-2}	6.16×10^{-1}	1.86×10^{-10}	1.52×10^{0}
	标准差	0	0	2.99×10^{10}	2.72×10^{2}	9.28×10^{3}	3.63×10^{11}	2.37×10^{2}
F11	最优值	1.57×10^{32}	8.14×10^{11}	3.46×10^{3}	7.89×10^{1}	1.57×10^{0}	3.79×10^{13}	8.77×10^{1}
	平均值	1.57×10^{-32}	8.26×10^{-11}	3.46×10^{-3}	2.65×10^{0}	1.57×10^{0}	6.13×10^{-13}	1.01×10^{0}
	标准差	5.57×10^{48}	1.53×10^{12}	2.47×10^{11}	1.43×10^{0}	9.58×10^{4}	1.34×10^{13}	5.75×10^{2}
F12	最优值	1.35×10^{-32}	7.30×10^{-9}	8.42×10^{-3}	2.82×10^{-2}	7.56×10^{-2}	1.57×10^{-12}	7.54×10^{0}
	平均值	1.35×10^{-32}	7.36×10^{-9}	8.42×10^{-3}	8.76×10^{-2}	7.76×10^{-2}	2.67×10^{-12}	8.51×10^{0}
	标准差	5.57×10^{-48}	8.08×10^{-11}	3.05×10^{-9}	5.99×10^{-2}	1.03×10^{-3}	6.07×10^{-13}	4.08×10^{-1}
F13	最优值	-1.02×10^{1}	-9.81×10^{0}	-7.97×10^{0}	-1.02×10^{1}	-7.96×10^{0}	-9.81×10^{0}	-4.66×10^{0}
	平均值	-1.01×10^{1}	-9.60×10^{0}	-7.27×10^{0}	-6.15×10^{0}	-7.96×10^{0}	-9.47×10^{0}	-4.66×10^{0}
	标准差	3.47×10^{1}	4.93×10^{1}	9.80×10^{1}	3.39×10^{0}	2.73×10^{4}	6.71×10^{1}	9.04×10^{11}
F14	最优值	-1.04×10^{1}	-9.87×10^{0}	-9.62×10^{0}	-1.04×10^{1}	-7.54×10^{0}	-1.03×10^{1}	-5.76×10^{0}
	平均值	-1.04×10^{1}	-9.72×10^{0}	-9.22×10^{0}	-5.71×10^{0}	-7.54×10^{0}	-9.82×10^{0}	-5.72×10^{0}
	标准差	2.14×10^{-1}	4.44×10^{-1}	6.96×10^{-1}	3.38×10^{0}	2.31×10^{-4}	1.00×10^{0}	6.86×10^{-3}
F15	最优值	-1.05×10^{1}	-1.04×10^{1}	-9.50×10^{0}	-1.05×10^{1}	-9.39×10^{0}	-1.05×10^{1}	-7.12×10^{0}
	平均值	-1.05×10^{1}	-1.02×10^{1}	-9.09×10^{0}	-5.52×10^{0}	-9.39×10^{0}	-1.03×10^{1}	-7.11×10^{0}
	标准差	6.99×10^{2}	4.51×10^{1}	8.17×10^{1}	3.48×10^{0}	3.23×10^{4}	5.60×10^{1}	1.85×10^{3}

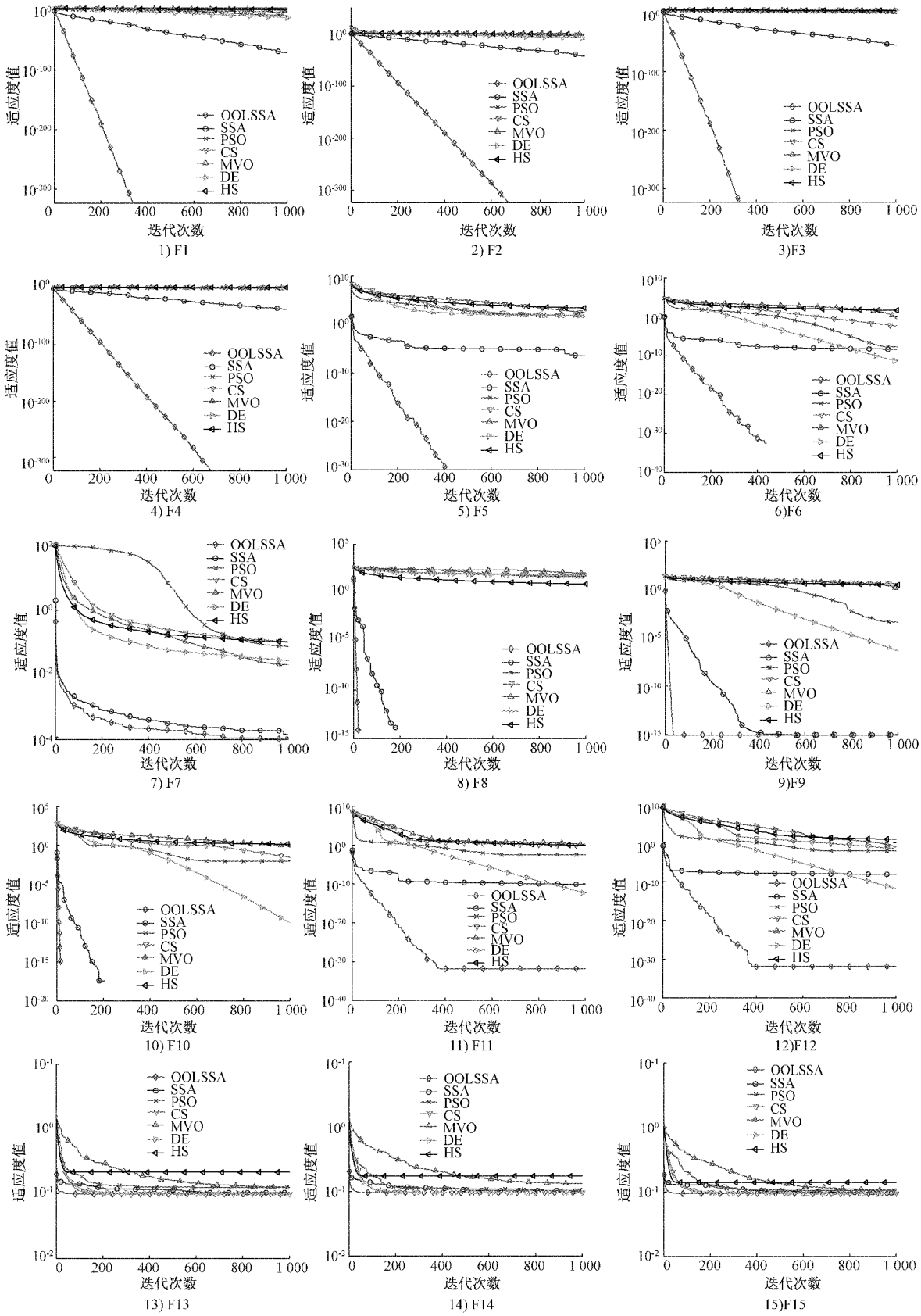

适应度值
迭代次数
OOLSSA
SSA
PSO
CS
MVO
DE
HS
1) F1
2) F2
3)F3
4) F4
5) F5
6)F6
7) F7
8) F8
9)F9
10) F10
11) F11
12)F12
13) F13
14) F14
15)F15

图 4　各算法收敛曲线

了局部最优或者已达到了理论最优。

由图4可知,在高维单峰测试函数F1～F6中,可以看到OOLSSA明显最快到达理论最优,并且该收敛速度稳定,保持下降趋势。在F7中,虽然本文所提算法陷入了局部最优,但相比于其他算法,其收敛精度是更好的。在高维多峰测试函数中F8～F10,在收敛精度相同的情况下,SSA在第200代以后收敛到该值,而OOLSSA均在20代以内完成收敛,且在F11与F12中,该算法在400代时的收敛精度高出其他算法20多个数量级。在低维多峰测试函数F13～F15中,虽然曲线较为接近,但可以明显看出OOLSSA在第50代左右就搜索到理论最优。实验表明本文所提算法的收敛速度性能有明显的提升。

3.3 与改进算法对比

为了进一步体现本文改进效果的有效性,挑选2个现存改进型麻雀搜索算法进行对比。改进麻雀搜索算法ISSA[11]与ISSA2[17]参数设置与原文相同,种群数量与迭代次数与3.2节设置相同。具体参数设置如表4所示。

表4 改进的算法的参数设置

优化算法	参数设置
OOLSSA	$PD=20\%$, $R_2=0.8$, $SD=10\%$
ISSA1	$PD=20\%$, $R_2=0.8$, $SD=10\%$, $\omega_{max}=1$, $\omega_{min}=0.4$
ISSA2	$PD=20\%$, $R_2=0.8$, $SD=10\%$

以上3个算法在15个基准测试函数的性能对比结果如表5所示,平均最优适应度收敛曲线图如图5所示。

表5 不同改进算法的寻优结果

测试函数	指标	OOLSSA	ISSA1	ISSA2	测试函数	指标	OOLSSA	ISSA1	ISSA2
F1	最优值	0	0	0	F9	最优值	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}
	平均值	1.50×10^{-44}	1.82×10^{-42}	0		平均值	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}
	标准差	8.20×10^{-44}	9.89×10^{-42}	0		标准差	0	0	0
F2	最优值	0	0	0	F10	最优值	0	0	0
	平均值	5.19×10^{-23}	1.08×10^{-22}	0		平均值	0	0	0
	标准差	2.84×10^{-22}	5.90×10^{-22}	0		标准差	0	0	0
F3	最优值	0	0	0	F11	最优值	1.57×10^{32}	8.35×10^{9}	1.46×10^{9}
	平均值	2.11×10^{-41}	9.95×10^{-41}	0		平均值	1.57×10^{-32}	8.43×10^{-9}	3.97×10^{-6}
	标准差	9.56×10^{-41}	5.45×10^{-40}	0		标准差	5.57×10^{-48}	6.50×10^{-11}	8.82×10^{-6}
F4	最优值	0	0	0	F12	最优值	1.35×10^{32}	5.25×10^{8}	5.23×10^{8}
	平均值	4.70×10^{-24}	1.79×10^{-23}	0		平均值	1.35×10^{-32}	5.42×10^{-8}	6.31×10^{-5}
	标准差	2.57×10^{-23}	9.78×10^{-23}	0		标准差	5.57×10^{-48}	3.44×10^{-9}	1.69×10^{-4}
F5	最优值	0	2.00×10^{5}	4.55×10^{6}	F13	最优值	-1.02×10^{1}	-7.09×10^{0}	-9.81×10^{0}
	平均值	0	2.08×10^{5}	1.34×10^{2}		平均值	-1.01×10^{1}	-6.95×10^{0}	-7.46×10^{0}
	标准差	0	7.08×10^{-7}	3.40×10^{-2}		标准差	1.67×10^{-1}	3.94×10^{-1}	2.41×10^{0}
F6	最优值	0	9.40×10^{8}	6.11×10^{8}	F14	最优值	-1.04×10^{1}	-7.57×10^{0}	-1.02×10^{1}
	平均值	0	9.56×10^{-8}	7.57×10^{-5}		平均值	-1.04×10^{1}	-7.20×10^{0}	-7.68×10^{0}
	标准差	0	1.89×10^{-9}	2.02×10^{-4}		标准差	1.95×10^{-1}	5.26×10^{-1}	2.60×10^{0}
F7	最优值	6.17×10^{5}	8.27×10^{5}	8.69×10^{5}	F15	最优值	-1.05×10^{1}	-8.01×10^{0}	-1.04×10^{1}
	平均值	4.97×10^{3}	5.30×10^{3}	5.15×10^{3}		平均值	-1.05×10^{1}	-7.74×10^{0}	-7.76×10^{0}
	标准差	6.11×10^{-3}	6.52×10^{-3}	6.21×10^{-3}		标准差	2.30×10^{-1}	6.03×10^{-1}	2.64×10^{0}
F8	最优值	0	0	0					
	平均值	0	0	0					
	标准差	0	0	0					

由表5与图5可知,在高维单峰测试函数中,3个改进算法在F1～F4中均较快的找到最优值,但OOLSSA的收敛速度不及ISSA1或ISSA2。在F5与F6中,在其他算法明显陷入局部最优的情况下,本文算法在400代附近找到理论最优。在F7中,所有算法并无明显差异性。在高维多峰测试函数F8～F10中,在所有算法收敛精度相同的情况下,OOLSSA收敛速度最快。而在高维多峰和低维多峰测试函数F11～F15中,OOLSSA的收敛速度与收敛精度均表现出优异性。

实验说明,相比于其他改进算法,OOLSSA收敛速度

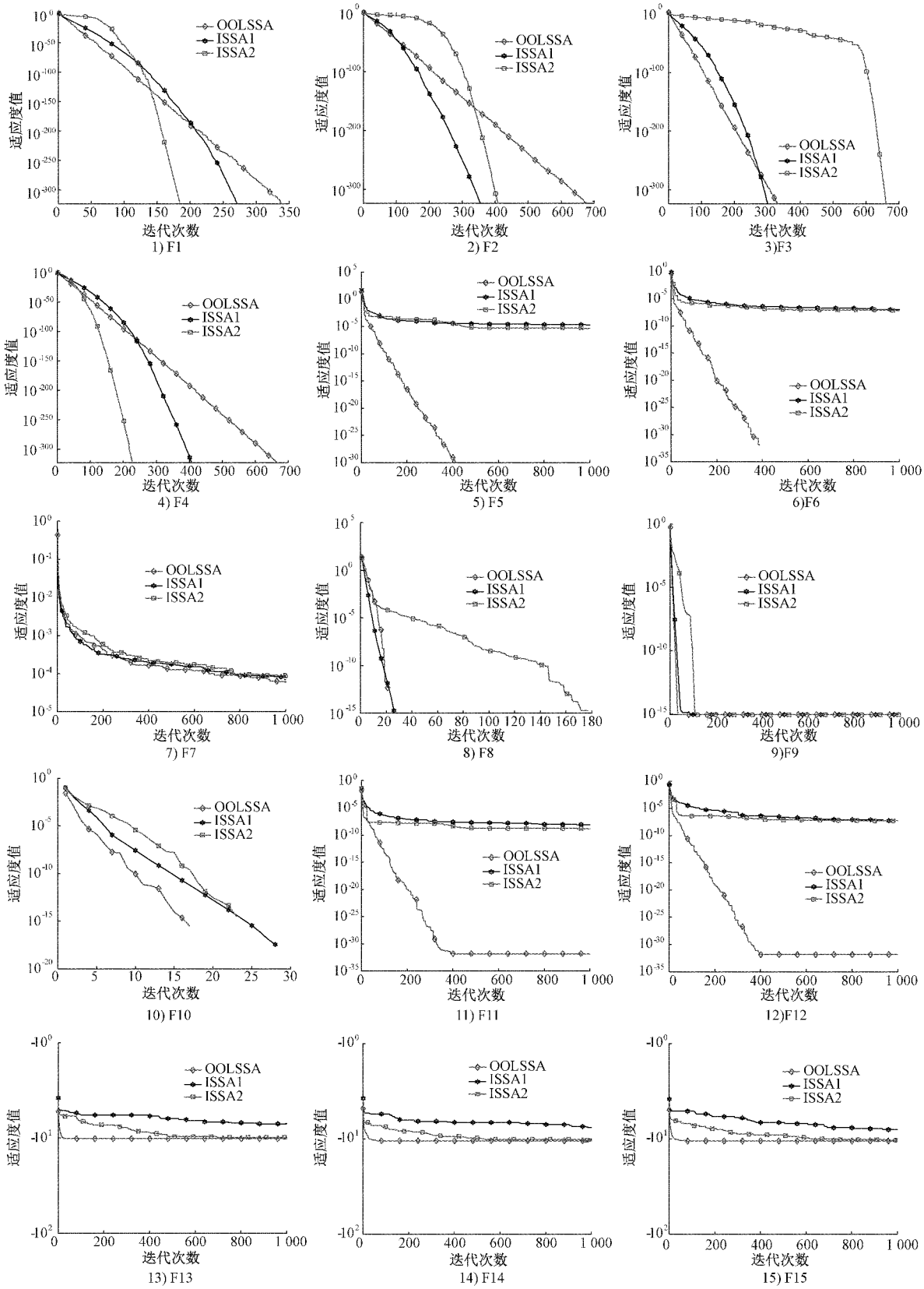


图 5　改进算法收敛曲线

较快，收敛精度较高，稳定性较好，有良好的全局搜索能力　　与局部开发能力。

3.4 非参数 Friedman 检验

为了进一步评估改进麻雀搜索算法的性能,下面取 3 种改进算法的平均值指标进行非参数 Friedman 检验。该检验是一种多样本齐一性的统计检验,其公式如式(9):

$$F_f=\frac{12n}{k(k+1)}\left[\sum_j R_j^2-\frac{k(k+1)^2}{4}\right] \tag{9}$$

式(9)中:k 是算法数量,R_j 是第 j 个算法的平均排名,n 是实验次数,该检验假设实验数据服从 $k-1$ 自由度的卡方分布,通过对 30 次的实验数据进行排序并计算平均秩,即可得到每个算法在不同测试函数的最终秩。非参数 Friedman 检验结果列于表 6。

表 6 非参数 Friedman 检验

测试函数	F1	F2	F3	F4	F5	F6	F7	F8
OOLSSA	2.37	2.33	2.27	2.40	1.00	1.00	1.70	2.00
ISSA1	2.63	2.67	2.73	2.60	2.00	2.00	2.30	2.00
ISSA2	1.00	1.00	1.00	1.00	3.00	3.00	2.00	2.00
测试函数	F9	F10	F11	F12	F13	F14	F15	平均排名
OOLSSA	2.00	2.00	1.00	1.00	1.03	1.20	1.13	1.63
ISSA1	2.00	2.00	2.00	2.00	2.57	2.33	2.30	2.28
ISSA2	2.00	2.00	3.00	3.00	2.40	2.47	2.57	2.10

由表 6 可知,在 F1~F4 中,ISSA2 表现出绝对的优势,在 F8~F10 中,所有算法无明显差异,但在 F5~F7 及 F11~F15 中,OOLSSA 具有比较显著的优势,证明 OOLSSA 的改进策略的有效性。

3.5 探索与开发能力分析

为了验证改进的算法能够平衡算法的探索与开发能力,本文利用 Hussain 等[23]提出的维度多样性度量方法来对算法的探索和开发能力进行度量,公式定义如式(10)~(11)所示。

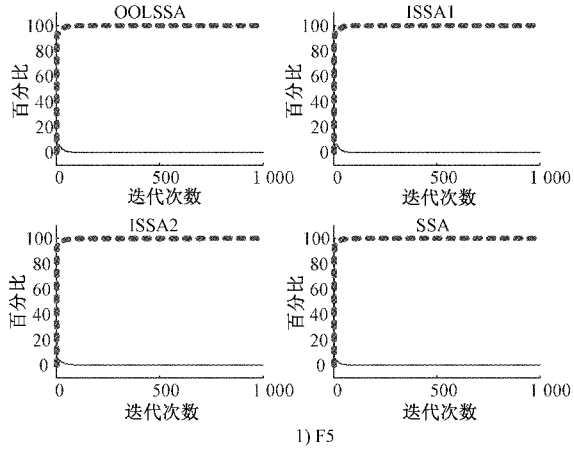
$$\begin{aligned}\boldsymbol{Div}_j&=\frac{1}{N}\sum_{i=1}^{N}median(\boldsymbol{x}^j)-\boldsymbol{x}_i^j\\ \boldsymbol{Div}^t&=\frac{1}{D}\sum_{j=1}^{D}\boldsymbol{Div}_j,t=1,2,\cdots,iter_{\max}\end{aligned} \tag{10}$$

$$\begin{aligned}Xpl\%&=\frac{\boldsymbol{Div}^t}{\boldsymbol{Div}_{\max}}\times 100\\ Xpt\%&=\frac{|\boldsymbol{Div}^t-\boldsymbol{Div}_{\max}|}{\boldsymbol{Div}_{\max}}\times 100\end{aligned} \tag{11}$$

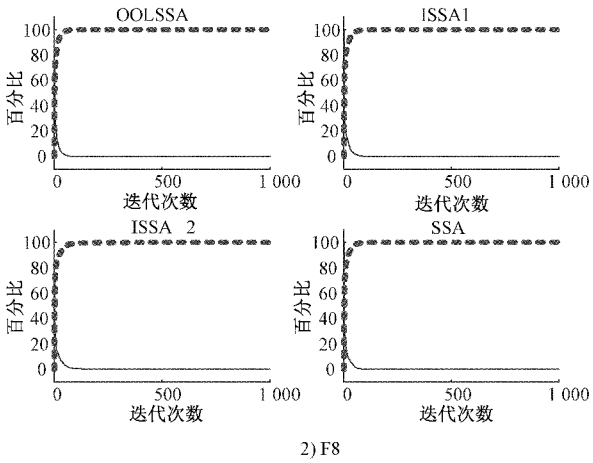
式(10)~(11)中:Div^t 记录下第 t 代各位置与其对应维度的中位数的差的平均值。$Xpl\%$ 代表探索值的百分比,$Xpt\%$ 代表开发值的百分比,二者之和为 1。

根据该方法的定义,算法在搜索过程中的探索与开发能力可通过维度内种群之间的平均距离进行度量。一般情况下,初期时由于初始化让种群分布均匀散布在搜索空间内,因此初期探索值大,而后期由于种群的收敛性,种群内距离较小,此时开发值较大。但由于测试函数的复杂性,探索与开发的占比规律可能会不同。图 6 是 3 个改进算法与 SSA 在不同类型测试函数的探索开发能力对比图。

图 6 中,实线代表探索能力,虚线代表开发能力。由图 6 可以看出,4 个算法在各个高维的测试函数上的有强烈的开发能力,均较早地进入开发阶段,总体并无明显差异。而在低维多峰的测试函数中,4 个算法才有明显的区别,结合表格数据可以看到,OOLSSA 在 100 代以内收敛到最优值后,开始进入开发阶段,进行精细的搜索,此时探索百分比下降,开发百分比上升,表现种群正在收敛。而其他算法则陷入了局部最优且收敛性与稳定性一般。该实验结果表明,OOLSSA 有较好的平衡探索与开发的能力。



1) F5



2) F8

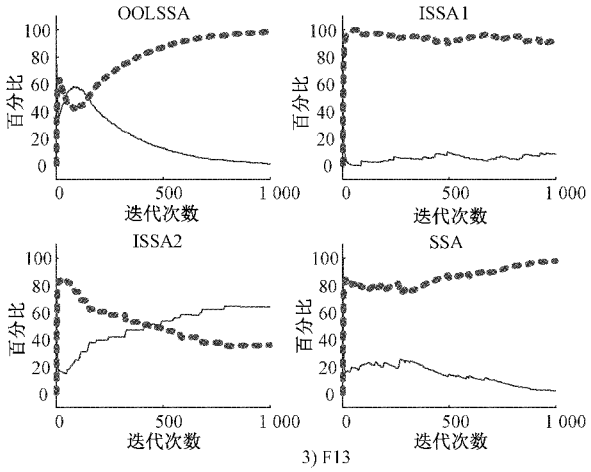


图 6　探索开发能力对比图

4　结　　论

针对麻雀搜索算法存在容易陷入局部最优等问题，本文提出一种基于正交对立学习的改进麻雀算法(OOLSSA)，并对 15 个基准测试函数进行仿真实验、非参数 Friedman 检验以及探索开发能力分析，本文的结论与展望如下：

OOLSSA 寻优精度显著提升。实验表明，引入的正态变异算子与对立学习策略有效提高了算法跳出局部最优的能力，算法的全局与局部搜索能力表现优异。

OOLSSA 收敛速度快。实验结果证明了正交对立学习机制能够有效提高算法的收敛速度，其平均迭代次数明显优于 6 个传统算法与部分改进型麻雀搜索算法。

未来可以考虑继续改进麻雀搜索算法，并将 OOLSSA 算法应用到实际工程问题中，检验 OOLSSA 在实际应用的价值。

参考文献

[1] KENNEDY J, EBERHART R. Particle swarm optimization[C]. Proceedings of ICNN'95-international Conference on Neural Networks, IEEE, 1995, 4: 1942-1948.

[2] PAN W T. A new fruit fly optimization algorithm: Taking the financial distress model as an example[J]. Knowledge-Based Systems, 2012, 26:69-74.

[3] MIRJALILI S, LEWIS A. The whale optimization algorithm[J]. Advances in Engineering Software, 2016, 95: 51-67.

[4] MIRJALILI S, MIRJALILI S M, LEWIS A. Grey wolf optimizer[J]. Advances in Engineering Software, 2014, 69: 46-61.

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

[6] 吕鑫，慕晓冬，张钧. 混沌麻雀搜索优化算法[J]. 北京航空航天大学学报，2021，47(8)：1712-1720.

[7] 尹德鑫，张达敏，蔡朋宸，等. 改进的麻雀搜索优化算法及其应用[J/OL]. 计算机工程与科学：1-8[2022-01-03]. http://kns.cnki.net/kcms/detail/43.1258.TP.20210930.1242.002.html.

[8] 段玉先，刘昌云. 基于 Sobol 序列和纵横交叉策略的麻雀搜索算法[J]. 计算机应用，2022，42(1)：36-43.

[9] LI J H, LEI Y S, YANG S H. Mid-long term load forecasting model based on support vector machine optimized by improved sparrow search algorithm[J]. Energy Reports, 2022, 8(S5).

[10] JIANG Z Y, GE J Q, XU Q Q, et al. Fast trajectory optimization for gliding reentry vehicle based on improved sparrow search algorithm[J]. Journal of Physics: Conference Series, 2021, 1986(1).

[11] 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.

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

[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.

[14] 阎大海，李元香，祝婕. 基于正交设计的反向学习差分进化算法[J]. 华中科技大学学报：自然科学版，2017，45(5)：23-27.

[15] 周凌云，丁立新，马懋德，等. 一种正交反向学习萤火虫算法[J]. 电子与信息学报，2019，41(1)：202-209.

[16] YANG X S, DEB S. Cuckoo search via Lévy flights[C]. 2009 World congress on nature & biologically inspired computing(NaBIC). Ieee, 2009: 210-214.

[17] MIRJALILI S, MIRJALILI S M, HATAMLOU A. Multi-verse optimizer: A nature-inspired algorithm for global optimization[J]. Neural Computing and Applications, 2016, 27(2): 495-513.

[18] STORN R, PRICE K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces[J]. Journal of Global Optimization, 1997, 11(4): 341-359.

[19] 雍龙泉. 和声搜索算法研究进展[J]. 计算机系统应用，2011，20(7)：244-248.

[20] 李黄曼，张勇，张瑶. 基于 ISSA 优化 SVM 的变压器故障诊断研究[J]. 电子测量与仪器学报，2021，243(3)：123-129.

[21] 吴阳，刘凯，陈柏，等. 自适应粒子群优化算法优化径向基函数神经网络用于电阻抗成像图像重建[J]. 仪器仪表学报，2020，41(6)：240-249.

[22] 戈一航，杨光永，徐天奇，等. 基于 SSA 优化 PID 在移动机器人路径跟踪中的研究[J]. 国外电子测量技术，2021，322(9)：64-69.

[23] KASHIF H, MOHD S, SHI C, et al. On the exploration and exploitation in popular swarm-based metaheuristic algorithms[J]. Neural Computing and Applications, 2019, 31(11): 7665-7683.

作者简介

王天雷(通信作者)，副教授，博士研究生，主要研究方向为欠驱动控制、智能控制及工业物联网。

E-mail: tianlei.wang@aliyun.com