第 36 卷 第 5 期
2021 年 10 月

天津科技大学学报
Journal of Tianjin University of Science & Technology

Vol. 36 No. 5
Oct. 2021

DOI:10.13364/j.issn.1672-6510.20200212
数字出版日期：2021-06-07；数字出版网址：http://kns.cnki.net/kcms/detail/12.1355.N.20210604.1552.003.html

基于全局最优值的群搜索优化算法

熊聪聪，李俊伟，杨晓艺，王 丹
（天津科技大学人工智能学院，天津 300457）

摘 要：标准群搜索优化（group search optimizer，GSO）算法在搜索的前期易于陷入局部最优，造成收敛速度变缓甚至在搜索时停滞等问题. 针对以上问题对 GSO 算法进行改进，提出一种基于全局最优值的群搜索优化（global optimal value-based group search optimizer，GGSO）算法，弥补标准群搜索优化算法在搜索的前期易于陷入局部最优的缺陷. 通过在 GSO 算法搜索过程中加入全局最优值方式，改进“发现者-加入者”模型，从而加快算法收敛速度. 11 个国际标准测试函数的对比实验表明，GGSO 算法无论在算法精度还是收敛速度上都优于标准 GSO 算法.

关键词：群搜索优化算法；全局最优值；收敛速度；收敛精度
中图分类号：TP18 **文献标志码**：A **文章编号**：1672-6510（2021）05-0062-06

Group Search Optimization Algorithm Based on Global Optimal Value

XIONG Congcong，LI Junwei，YANG Xiaoyi，WANG Dan
（College of Artificial Intelligence，Tianjin University of Science & Technology，Tianjin 300457，China）

Abstract：The standard group search optimizer（GSO）algorithm is easy to fall into the local optimum in the early stage of the search，resulting in slow convergence and even stagnation in the search. In view of the above problem，the global optimal value-based group search optimizer（GGSO）algorithm was improved by adding global optima to the search process and improving the GSO improved the “Producer-Scrounger” model by adding global optima value to the algorithm search process，thus speeding up the convergence of the algorithm. Comparative experiments on 11 international standard test functions showed that the proposed GGSO outperformed the standard GSO algorithm in terms of both algorithm accuracy and convergence speed.

Key words：group search optimizer（GSO）；global optimal value；convergence speed；convergence accuracy

在 20 世纪 60 年代早期，人们创造了许多工具来模拟生物的行为，仿生学就此诞生. 在这样的前提下，自然界有很多动物群体（蚁群、鸟群等）成为研究者关注的焦点，对动物的群体行为进行数学建模，并在计算机软件上进行模拟仿真实验，群智能（swarm intelligence，SI）算法[1-4]应运而生. 其中，比较经典的群智能算法有：Colorni 等[3]根据蚂蚁在寻找食物时的行为而提出的蚁群（ant colony optimizer，ACO）算法、Kenndy 等[4]根据鸟群在寻找食物时的行为从而提出的粒子群（particle swarm optimizer，PSO）算法、He 等[1, 5]提出的群搜索优化（group search optimizer，GSO）算法.

He 等[1]系统地阐述了群搜索优化算法的基本原理，解释了相关的数学模型，并将其与遗传算法（GA）、快速演化规划（FEP）[6]、快速演化策略（FES）[7]、粒子群（PSO）算法等群智能算法进行了比较和仿真实验，对比讨论该算法中的初始化参数对最终实验结果的影响.

虽然 GSO 在一些优化问题中显示出了比较显著的优越性，但其仍然存在部分不足之处，例如在处理一些优化问题时出现易陷入局部最优，造成收敛速度较慢、精度较低的问题. 因此，许多研究者都提出了

收稿日期：2020-12-10；**修回日期**：2021-03-22
基金项目：国家自然科学基金青年基金资助项目（11803022）
作者简介：熊聪聪（1961—），女，四川人，教授，xiongcc@tust.edu.cn

相应的改进方法. 安晓伟等[8]提出了保留发现者的搜索策略和由加入者执行鱼群算法的搜索策略，通过引入鱼群算法，可以避免种群搜索陷入局部最优. 在标准 GSO 算法的基础上，李丽娟等[9]提出了取消按角度搜索的策略，增加了控制变量随迭代次数减少而变化的概率，在一定程度上解决了标准 GSO 算法收敛速度慢的缺点. 杨文璐等[10]提出了一种基于交叉因子的改进群搜索优化算法，通过将交叉因子和模拟退火算法结合，使得群体中粒子多样性增加，从而避免使 GSO 算法陷入局部最优值的可能性. 景书杰等[11]提出一种群搜索优化算法，该算法通过对发现者搜索方式中加入最大下降方向，把游荡者生成策略改变为基因突变策略，从而提高了标准 GSO 算法的收敛速度. 郝璐萌[12]提出在发现者搜索过程中引入不同的差分变异策略，提高收敛精度. 王娟等[13]提出了基于改进 Tent 混沌映射的种群初始化和基于 Levy 飞行特征的跟随者更新策略，提高了解决高维复杂问题的收敛速度和求解精度.

针对 GSO 算法存在的问题，本文提出了一种基于全局最优值的群搜索优化(global optimal value-based group search optimizer，GGSO)算法，该算法针对在标准群搜索优化算法发现者搜索过程中收敛速度慢，引入全局最优值，从而加速算法收敛速度. 选取 11 组标准测试函数进行测试，并与 GSO 算法的测试结果比较，GGSO 算法整体上相比 GSO 算法具有更好的收敛速度和精度.

1　群搜索优化算法

群搜索优化(group search optimizer，GSO)算法，根据群居动物搜索食物行为而设计的一种仿真模拟算法. 该算法的原型是“发现者–加入者”(producer-scrounger，PS)模型. 在 GSO 算法中将研究对象按照在种群中的功能分为 3 类：搜索资源并共享其信息的发现者、对发现者搜索到的资源进行掠夺的加入者、执行随机搜索的游荡者. 群搜索优化算法是基于动物种群之间信息共享和相互合作的特性而建立的一种智能优化算法[1]. 该算法模仿了动物的视觉搜索机制，在其搜索过程中，适应度值最好的个体作为发现者带头搜索，发现者按照既定的搜索角度和方向寻找更好的资源，加入者跟随发现者进行局部搜索，剩余的个体作为游荡者，根据搜索角度在当前的范围内调整自己的位置. 群体中的 3 类个体相互协同协作完成搜索的任务，从而使所找到的资源为最优.

GSO 算法的实现过程：设 n 维搜索空间内，在每次迭代搜索过程中，第 i 个个体的位置用向量表示为 $\boldsymbol{X}_i^k \in R^n$，其搜索角度为 $\varphi_i^k = \left(\varphi_{i_1}^k, \cdots, \varphi_{i_{n-1}}^k\right) \in R^{n-1}$，搜索方向为向量 $\boldsymbol{D}_i^k\left(\varphi_i^k\right) = \left(d_{i_1}^k, \cdots, d_{i_n}^k\right) \in R^n$，该方向向量可以通过式(1)—式(3)按照搜索角度 φ_i^k 通过转换得到笛卡尔坐标.

$$d_{i_1}^k = \prod_{q=1}^{n-1} \cos\left(\varphi_{i_q}^k\right) \tag{1}$$

$$d_{i_j}^k = \sin\left(\varphi_{i_{j-1}}^k\right) \cdot \prod_{q=j}^{n-1} \cos\left(\varphi_{i_q}^k\right) \tag{2}$$

$$d_{i_n}^k = \sin\left(\varphi_{i_{n-1}}^k\right) \tag{3}$$

1.1　发现者

在每次迭代过程中，均选择适应度值最好的个体作为本次迭代过程中的发现者(producer)，发现者在共享资源的同时继续搜索资源. 搜索策略是在当前位置沿着 3 个不同的角度进行扫描搜索，找到 3 个方向上不同的位置，通过式(4)—式(6)分别在其前方、左方以及右方进行扫描查找.

$$X_z = X_p^k + r_1 l_{\max} D_p^k\left(\varphi^k\right) \tag{4}$$

$$X_r = X_p^k + r_1 l_{\max} D_p^k\left(\varphi^k + \frac{r_2 \theta_{\max}}{2}\right) \tag{5}$$

$$X_l = X_p^k + r_1 l_{\max} D_p^k\left(\varphi^k - \frac{r_2 \theta_{\max}}{2}\right) \tag{6}$$

其中：$r_1 \in R^1$ 为服从均值为 0、方差为 1 的正态分布随机数；$r_2 \in R^{n-1}$ 为[0，1)间均匀分布的随机数；$l_{\max}$ 为搜索的最大距离，并能够通过式(7)计算得到；$\theta_{\max}$ 为搜索的最大角度.

$$l_{\max} = \left\| \sqrt{\sum_{i=1}^{n} \left(U_i - L_i\right)^2} \right\| \tag{7}$$

其中 U_i 和 L_i 分别是取值范围的上界和下界.

如果 3 个随机位置中某一个要好于当前发现者的位置，就将发现者的位置和角度更新为随机位置中较好的位置和角度；反之，发现者则不改变其当前位置信息，并通过式(8)更新搜索角度，进行下一次迭代.

$$\varphi^{k+1} = \varphi^k + r_2 \alpha_{\max} \tag{8}$$

其中 $\alpha_{\max} \in R^1$ 为最大的搜索转换角度.

发现者如果在 α 次迭代后无法找到更好的资源时，通过式(9)回到 0°位置.

$$\varphi^{k+\alpha} = \varphi^k \tag{9}$$

其中 $\alpha \in R^1$ 为实验人员自行确定的常数.

1.2　加入者

随机选择剩余个体的 80% 作为加入者

(scrounger)，加入者一直跟随共享由发现者搜索到的位置资源信息，如果加入者找到比当前发现者更好的资源时，在下一次迭代中按照式(10)加入者会转换为发现者角色进行搜索.

$$X_i^{k+1} = X_i^k + r_3\left(X_p^k - X_i^k\right) \tag{10}$$

其中 $r_3 \in R^n$ 为服从[0，1)均匀分布的随机数.

1.3　游荡者

剩余个体作为游荡者(ranger)在搜索空间中随机分布并且进行资源搜索. 在第 k 次迭代中，游荡者通过式(8)产生随机角度，并通过式(11)得出随机距离 l_i，通过式(12)更新位置.

$$l_i = \alpha \cdot r_1 l_{\max} \tag{11}$$

$$X_i^{k+1} = X_i^k + l_i D_i^k\left(\varphi_i^{k+1}\right) \tag{12}$$

通过3种角色个体间的相互合作，更新种群信息，最终收敛得到种群中的最优个体. 但是为了在搜索资源时更加方便，所以在一定程度上限制了个体的搜索界限，如果某个个体在搜索时跳出该搜索界限，会判断其是否越界，并让该个体返回到之前的搜索位置.

1.4　GSO算法整体框架

标准 GSO 算法流程图如图 1 所示.

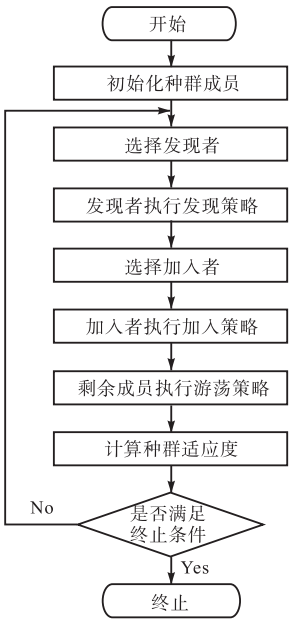

图 1　标准 GSO算法流程图

Fig. 1　Standard GSO algorithm flow chart

标准 GSO 算法(算法 1)如下：

1. 输入：随机向量 $\boldsymbol{X}_i$.
2. 输出：当前最优个体位置信息.
3. 初始化种群中所有个体的角度和位置信息.
4. 计算种群个体的适应度值 fvalue.
5. WHILE(iter＜MaxIter).
6. 选取发现者，按照式(4)—式(9)执行发现者操作.
7. 选择剩余成员 80% 作为加入者，按照式(10)执行加入者操作.
8. 选择剩余成员 20% 作为游荡者，按照公式(11)和公式(12)执行游荡者操作.
9. 计算种群的适应度值 fvalue.
10. END WHILE.

1.5　GSO参数选择

文献[5]给出了以下相关参数的参数值：$\varphi^0 = (\pi/4, \cdots, \pi/4)$，$\theta_{\max} = \pi/a^2$，$\alpha_{\max} = \theta_{\max}/2$，$a = \text{round}\left(\sqrt{n+1}\right)$.

2　改进的群搜索优化算法

为了提升算法在搜索空间内的资源寻优能力，提出一种基于全局最优值的群搜索优化(global optimal value-based group search optimizer，GGSO)算法. 该算法以标准 GSO 算法的 PS 模型为基础，并在搜索时加入全局最优值作为发现者，提高了算法的搜索性能和收敛精度. 在该算法中，先求得所有群体的适应度值，然后得到其中的最优值，在发现者的搜索过程中，加入全局最优值. 在不断的迭代过程中，每一次迭代加入全局最优值以后，使得全局最优值有机会参与到发现者的搜索过程中，进而使算法可以跳出局部最优值，提升算法收敛精度，从而提高算法的全局搜索能力.

2.1　GGSO算法的实现过程

(1)在前期时，算法采用标准 GSO 算法流程进行进化和迭代.

(2)保存所有成员适应度值的最小值，即在迭代开始之前保存某个成员 i 当前位置 X_i^k 及其适应度 $f\left(X_i^k\right)$ 作为全局最优值 fbestval.

$$\text{fbestval=min(fvalue)} \tag{13}$$

(3)在发现者搜索过程中，每一轮迭代在发现者适应度值中都加入之前保存的全局最优值 fbestval.

(4)与此同时，发现者、加入者和游荡者均按照 GSO 算法的搜索方式进行搜索.

2.2　GGSO算法伪代码

GGSO 算法伪代码(算法 2)如下：

1. 输入：随机向量 $\boldsymbol{X}_i$.
2. 输出：当前最优个体位置信息.

3. BEGIN.

4. 随机初始化种群种个体的位置和角度信息.

5. 计算种群个体的适应度值 fvalue.

6. 选取适应度值最好的点作为发现者.

7. WHILE(迭代是否满足).

8. 将全局最优值加入发现者群体.

9. 选择发现者，由式(4)—式(9)执行发现者操作.

10. 选择剩余成员 80% 作为加入者，由式(10)执行加入者策略.

11. 选择剩余成员 20% 作为游荡者，由式(11)和式(12)执行游荡者策略.

12. 再次计算种群的适应度值 fvalue，得到当前最优适应度值.

13. END WHILE.

全局最优值即在优化问题的全值域范围内得到的最优值. 局部最优值即在优化问题的解在一定范围或者区域内的最优值，或者优化问题在一定的限制条件内最优值. 从图 2 即可明显看出两者的区别.

如果存在 $X_B^* \in B$，使得对 $\forall X \in B$，存在

$$\left(X_B^*\right) \leqslant f(X), X \in B \tag{14}$$

其中：$B \subset S \subseteq R^n$，$S$ 为函数的搜索空间，则称 X_B^* 为 $f(X)$ 的局部最小值点，$f\left(X_B^*\right)$ 为局部最小值.

如果存在 $X^* \in S$，使得对 $\forall X \in S$，存在

$$f\left(X^*\right) \leqslant f(X), X \in S \tag{15}$$

其中 $S \subseteq R^n$ 为函数的搜索空间，则称 X^* 为 $f(X)$ 的全局最优值点，$f\left(X^*\right)$ 为其全局最优值.

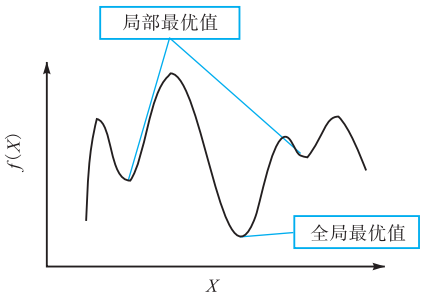

图 2　局部最优值与全局最优值示意图

Fig. 2　Diagram of local optimum and global optimum

2.3　全局最优策略

虽然 GSO 算法在部分问题的解决上有优越的表现，但其仍然面临在发现者搜索效果不为显著的问题. 受到“适者生存”的规律启发，优秀的个体较为容易生存. 因此，将全局最优策略应用到 GSO 算法的发现者个体选择中. 算法 3 详细描述了该操作.

算法 3：根据适应度值对种群中最优个体进行记录.

1. 输入：每轮迭代中的最小适应度值.

2. 输出：所有迭代中的最小适应度值位置信息.

3. BEGIN.

4. WHILE(Iter＜MaxIter).

5. 得到迭代中的最小适应度值 fvalue 及其位置信息.

6. Min(fvalue)→index.

7. END WHILE.

8. 根据 index 更新发现者群体信息.

9. END.

在该策略中虽然体现出了与粒子群算法较为相似之处，粒子群算法初始化得到随机粒子群，然后再通过迭代找到最优解. 在之后的每次迭代中，其中的粒子通过查找局部最优值和全局最优值更新其本身的位置. 当初始化粒子群后，得到其适应度值并找到全局最优值. 但是，上述算法 3 中提到的策略与粒子群算法不同之处在于，在得到每次迭代中的最优值之后将其记录下来，最终将所有最优值均作为发现者群体进行搜索，对发现者种群产生影响，进而可以进一步得到其全局最优值.

在对具体优化问题求解时，主要思路就是找到当前空间内的局部最优值，或者说通过迭代计算找到目标函数的解，直到找到两个解没有变化为止，此时即为该目标函数的局部最优值. 如果目标函数及其约束条件均为凸函数，通过算法找到的解即为全局最优值. 如果目标函数不是凸函数，通过算法找到的最优值，有可能是全局最优值，也有可能为局部最优值.

3　实验仿真与结果分析

3.1　国际标准测试函数

测试过程中选取了 11 个国际标准函数(见表 1)，主要分为两大类，单峰函数 f_1—f_6 和多峰函数 f_7—f_{11}.

3.2　实验环境与参数设置

实验所用到的环境为：操作系统 Win10，软件 Matlab2017a，运行内存 8 GB. 实验中算法所用到的参数与初始 GSO 算法参数均保持一致，种群规模设置为 48，种群个体维数设置为 30，最大迭代次数设置为 5 000 次，测试函数运行次数设置为 50 次.

3.3　实验结果及算法比较分析

通过对比标准 GSO 算法和 GGSO 算法在 11 个国际标准测试函数上的平均值和标准差(表 2),评价两种算法的性能. 其中,平均值和标准差是同一个标准测试函数上运行 50 次的实验平均结果的统计. 同时,黑色加粗字体表示两个算法在同一个测试函数上所取得的平均值的最优值.

表 1　11个测试函数

Tab. 1　Eleven benchmark functions

测试函数	n	S	$f_{\min}$
$f_1(x)=\sum_{i=1}^{n}x_i^2$	30	$[-100,100]^n$	0
$f_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	$[-10,10]^n$	0
$f_3(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{n}x_j\right)^2$	30	$[-100,100]^n$	0
$f_4(x)=\max_i\{\lvert x_i\rvert,1\leqslant i\leqslant n\}$	30	$[-30,30]^n$	0
$f_5(x)=\sum_{i=1}^{n-1}\left(100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)\right)^2$	30	$[-100,100]^n$	0
$f_6(x)=\sum_{i=1}^{n}\left(x_i+0.5\right)^2$	30	$[-1.28,1.28]^n$	0
$f_7(x)=-\sum_{i=1}^{n}\left(x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$	30	$[-500,500]^n$	− 12 569.5
$f_8(x)=\sum_{i=1}^{n}\left(x_i^2-10\cos\left(2\pi x_i\right)+10\right)^2$	30	$[-5.12,5.12]^n$	0
$f_9(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right)+20+\mathrm{e}$	30	$[-32,32]^n$	0
$f_{10}(x)=\frac{1}{4000}\sum_{i=1}^{30}\left(x_i-100\right)^2-\prod_{i=1}^{n}\cos\left(\frac{x_i-100}{\sqrt{i}}\right)+1$	30	$[-600,600]^n$	0
$f_{11}(x)=\frac{\pi}{n}\left\{10\sin^2\left(\pi y_1\right)+\sum_{i=1}^{29}\left(y_i-1\right)^2\left[1+10\sin^2\left(\pi y_{i+1}\right)\right]+\left(y_n-1\right)^2\right\}+\sum_{i=1}^{30}u\left(x_i,10,100,4\right)$ 式中：$y_i=1+\frac{1}{4}\left(x_i+1\right)$，$u\left(x_i,a,k,m\right)=\begin{cases}k\left(x_i-a\right)^m & x_i>a\\ 0 & -a\leqslant x_i\leqslant a\\ k\left(-x_i-a\right)^m & x_i<-a\end{cases}$	30	$[-50,50]^n$	0

表 2　两种算法在 11 个测试函数上的比较结果

Tab. 2　Compared results of the two algorithms on 11 benchmark functions

函数	最小值	GSO		GGSO	
		平均值	标准差	平均值	标准差
f_1	0	$1.034\,59\times10^{-9}$	$1.886\,23\times10^{-9}$	$\mathbf{4.498\,19\times10^{-10}}$	$7.284\,64\times10^{-10}$
f_2	0	$4.094\,27\times10^{-5}$	$1.700\,48\times10^{-5}$	$\mathbf{3.801\,00\times10^{-5}}$	$2.261\,63\times10^{-5}$
f_3	0	$9.745\,49\times10^{-8}$	$1.317\,76\times10^{-7}$	$\mathbf{2.396\,85\times10^{-8}}$	$3.881\,37\times10^{-8}$
f_4	0	$2.320\,22\times10^{-2}$	$6.571\,481\times10^{-3}$	$\mathbf{1.957\,24\times10^{-2}}$	$5.144\,637\times10^{-3}$
f_5	0	**52.255 2**	37.974 42888	52.346 2	32.360 3822
f_6	0	1.238 67	0.468 429 451	**1.173 05**	0.544 232 004
f_7	−12 569.5	$-1.909\,049\times10^{3}$	$4.547\,47\times10^{-13}$	$-1.909\,05\times10^{3}$	$4.547\,47\times10^{-13}$
f_8	0	3.397 84	1.339 541 464	**3.214 11**	2.512 191 297
f_9	0	$2.343\,90\times10^{-3}$	0.006 342 395	$\mathbf{6.630\,03\times10^{-4}}$	0.001 231 918
f_{10}	0	$8.037\,75\times10^{-2}$	0.050 755 855	$\mathbf{6.991\,22\times10^{-2}}$	0.052 334 834
f_{11}	0	$1.733\,68\times10^{-12}$	$2.396\,01\times10^{-12}$	$\mathbf{8.624\,51\times10^{-13}}$	$5.408\,24\times10^{-13}$

由表 2 可知:对于单峰函数(f_1—f_6),GGSO 算法收敛进度与 GSO 算法相差不大,f_1、f_2、f_3、f_4、f_6 在 GGSO 算法上得到的最优值优于标准 GSO 上的最优值,但其离散程度比标准 GSO 的离散程度要高. 对于多峰函数(f_7—f_{11}),除 f_7 外,其余 4 个测试函数在 GGSO 算法上的表现均优于在标准 GSO 算法上的表现,且其离散程度表现不一,f_9、f_{11} 的离散程度低于在标准 GSO 上的离散程度. 在 11 个标准测试函数上,

有 9 个标准测试函数在 GGSO 算法上的表现优于在标准 GSO 算法上的表现，仅有 1 个标准测试函数结果的精度稍逊于标准 GSO 算法，另有 1 个测试函数在两个算法上表现一致. 总体表明，GGSO 算法性能优于标准 GSO 算法.

为了更好地展示测试改进的算法在测试函数上的表现，实验中测试了迭代次数从 500 到 5 000 次的表现，以测试函数 f_6 和 f_{11} 的结果对比为例，结果如图 3 和图 4 所示. 从图 3 中可看出，改进的群搜索优化算法在测试函数 f_6 上在 3 000 次迭代之前得到的平均适应度均优于标准 GSO 算法，在 3 500 次迭代后收敛速度明显优于标准 GSO 算法的收敛速度，且得到更好的平均适应度. 由图 4 可知，虽然 GGSO 算法的收敛速度与标准 GSO 算法中收敛速度一致，但是可以得到比标准 GSO 算法更好的平均适应度.

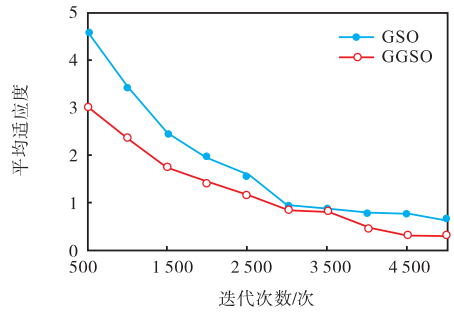

图 3　测试函数 f_6 结果对比

Fig. 3　Comparison of benchmark function f_6 results

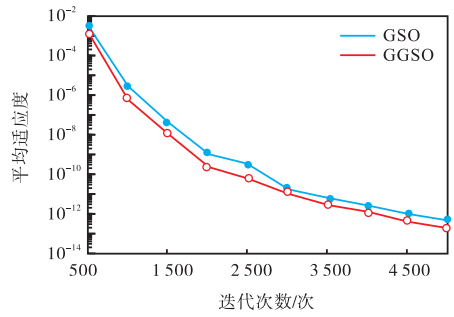

图 4　测试函数 f_{11} 结果对比

Fig. 4　Comparison of benchmark function f_{11} results

4　结　论

在标准 GSO 算法基础上，通过对算法中发现者的搜索策略进行改进，以达到提高算法的收敛速度和精度的效果. 针对标准 GSO 算法未考虑全局最优值的影响，通过在发现者搜索过程中加入全局最优值，一定程度增加了种群的多样性，算法的收敛速度和精度相比之前有所提高，避免了发现者在搜索时陷入局部最优. 实验结果表明，标准 GSO 算法的不足之处在 GGSO 算法上有一定的改进和弥补. 未来工作尚需在发现者和加入者搜索过程中加入更多优化策略，把改进过的群搜索优化算法具体应用实现.

参考文献：

[1] HE S，WU Q H，SAUNDERS J R. A novel group search optimizer inspired by animal behavioural ecology[C]// IEEE. 2006 IEEE International Conference on Evolutionary Computation. New York：IEEE，2006.

[2] HUANG W，OH S K，GUO Z，et al. A space search optimization algorithm with accelerated convergence strategies[J]. Applied soft computing，2013，13(12)：4659–4675.

[3] COLORNI A，DORIGO M，MANIEZZO V. Distributed optimization by ant colonies[EB/OL]. [2020–12–09]. http://staff.washington.edu/paymana/swarm/colorni92-ecal.pdf.

[4] KENNEDY J，EBERHART R. Particle swarm optimization[C]// IEEE. Icnn95-international Conference on Neural Networks. New York：IEEE，1995.

[5] HE S，WU Q H，SAUNDERS J R. Group search optimizer：An optimization algorithm inspired by animal searching behavior[J]. IEEE Transactions on evolutionary computation，2009，13(5)：973–990.

[6] YAO X，LIU Y. Evolutionary programming made faster[J]. IEEE Transactions on evolutionary computation，1999，3(2)：82–102.

[7] YAO X，LIU Y. Fast evolution strategies[EB/OL]. [2020–12–09]. http://link.springer.com/content/pdf/10.1007%2F BFb0014808.pdf.

[8] 安晓伟，苏宏升. 一种改进的群搜索优化算法[J]. 郑州大学学报：工学版，2015，36(2)：105–109.

[9] 李丽娟，张雯雱，徐小通，等. 改进的群搜索优化算法及其应用[J]. 空间结构，2010，16(2)：17–23.

[10] 杨文璐，宁玉富. 基于交叉因子和模拟退火的群搜索优化算法[J]. 计算机工程与设计，2013，34(6)：2020–2024.

[11] 景书杰，陈耀，牛海峰，等. 改进的群搜索优化算法[J]. 数学的实践与认识，2017，47(1)：258–264.

[12] 郝璐萌. 基于组群优化的聚类算法研究[D]. 天津：天津科技大学，2017.

[13] 王娟，王致杰，赵刘亮，等. 基于改进群搜索优化算法的综合能源系统运行优化[J]. 上海电机学院学报，2020，23(1)：1–8.

责任编辑：郎婧