应用反向学习和差分进化的群搜索优化算法

邹华福[1] 谢承旺[2] 周杨萍[1] 王立平[3]
(江西工业工程职业技术学院信息工程学院 江西 萍乡 337055)[1]
(广西师范学院科学计算与智能信息处理广西高校重点实验室 南宁 530023)[2]
(萍乡学院信息与计算机工程学院 江西 萍乡 337055)[3]

摘 要 针对标准群搜索优化算法在解决一些复杂优化问题时容易陷入局部最优且收敛速度较慢的问题,提出一种应用反向学习和差分进化的群搜索优化算法(Group Search Optimization with Opposition-based Learning and Differential Evolution,OBDGSO)。该算法利用一般动态反向学习机制产生反向种群,扩大算法的全局勘探范围;对种群中较优解个体实施差分进化的变异操作,实现在较优解附近的局部开采,以改善算法的求解精度和收敛速度。这两种策略在 GSO 算法中相互协同,以更好地平衡算法的全局搜索能力和局部开采能力。将 OBDGSO 算法和另外 4 种群智能算法在 12 个基准测试函数上进行实验,结果表明 OBDGSO 算法在求解精度和收敛速度上具有较显著的性能优势。

关键词 反向学习,差分进化,群搜索优化算法
中图法分类号 TP301 **文献标识码** A

Group Search Optimization with Opposition-based Learning and Differential Evolution

ZOU Hua-fu[1] XIE Cheng-wang[2] ZHOU Yang-ping[1] WANG Li-ping[3]
(Information Engineering College,Jiangxi Vocational College of Industry & Engineering,Pingxiang,Jiangxi 337055,China)[1]
(Science Computing and Intelligent Information Processing of Guangxi Higher Education Key Laboratory,Guangxi Teachers Education University,Nanning 530023,China)[2]
(School of Information and Computer Engineering,Pingxiang University,Pingxiang,Jiangxi 337055,China)[3]

Abstract In general,the standard group search optimization algorithm (GSO) is easy to fall into the local optimum and its convergence speed is slow when solving some complex optimization problems. A group search optimization algorithm based on opposition-based leaning and differential evolution (OBDGSO) was proposed in this paper. The OBDGSO uses the opposition-based learning operator to generate the opposite population to expand the global exploration range. In addition,the operator of differential evolution (DE) is utilized to perform local exploitation to improve the solution accuracy. These two strategies are integrated into the GSO to better balance the abilities of the global convergence and local search. The OBDGSO is tested on 12 benchmark functions along with four other peering algorithms,and the experimental results show that the OBDGSO has significant performance advantages in solution accuracy and convergence speed.

Keywords Opposition-based learning,Differential evolution,Group search optimization algorithm

1 引言

He 等[1]于 2006 年在模拟生物群体搜索食物的基础上提出了群搜索优化算法(Group Search Optimization,GSO)。该算法是一种基于发现者-加入者(Producer-Scrounger,PS)模型的群智能优化算法,它按照角度搜索机制来寻优,每次迭代时个体都会更新自己的位置和角度,而且不同角色的个体有不同的更新机制。GSO 算法自提出以来,主要被用于解决连续空间优化问题。与其他群智能优化算法不同的是,它将种群中的个体分成了 3 种角色:负责搜索资源并共享资源信息的发现者(Producer)、从发现者获取资源信息并掠夺资源的加入者(Scrounger)以及兼有发现者和加入者角色的游荡者(Ranger)。群体中任何个体在某一时刻只能扮演一种角色,即要么是发现者,要么是加入者,要么是游荡者,但任何个体都可以在以上 3 种角色中进行切换。

GSO 算法自提出以来在解决高维多峰函数问题中取得了良好的效果,并在多个领域得到了成功应用。例如,在神经元网络的训练方面,GSO 算法被用来训练神经元网络,基于

本文受国家自然科学基金(61763010),科学计算与智能信息处理广西高校重点实验室开放课题(GXSCIIP201604),江西省高校人文社会科学重点基地项目(JD17127),江西省重点研发计划项目(20071BBE50049)资助。
邹华福(1979-),男,硕士,副教授,主要研究方向为智能计算;**谢承旺**(1974-),男,博士,副教授,主要研究方向为智能计算;**周杨萍**(1978-),女,硕士,高级实验师,主要研究方向为智能计算;**王立平**(1979-),男,博士,副教授,主要研究方向为智能计算,E-mail:wlp8631@163.com。

GSO 算法的神经元网络被用来预测 TicTacToe game 的树叶节点的状态[2]；在电力系统方面，Tang 等[3]利用 GSO 算法解决了电力系统方面的问题；在工程结构的优化设计方面，GSO 算法被用于桁架结构的截面优化设计[4]等。

虽然 GSO 算法在多个优化问题中获得了成功，但大量的研究表明，标准 GSO 算法存在一些不足，例如，在处理多模态优化问题时容易陷入局部极值；在优化的后期收敛速度明显变慢，甚至停滞不前，难以获得全局最优解；算法需要进行大量的三角函数的乘积运算，导致算法求解速度较慢。为了弥补该算法的缺陷，不同的研究者使用不同的策略对 GSO 算法进行了改进，并由此产生了多种 GSO 算法的变体。李丽娟等[5]改进了标准 GSO 算法，放弃了按角度搜索的方式，增加了一个随迭代次数递减的控制变量-分量变异概率，从而在一定程度上解决了 GSO 算法耗时、收敛性差的缺陷。庞艳娟等[6]在 GSO 发现者的搜索模式中巧妙地加入了 Metropolis 准则，使算法能够接受一些劣解，使其跳出局部极值点，提高了算法的全局搜索能力，并在高维多峰函数的优化中效果明显。文献[7]提出了一种结合群搜索和粒子群优化的混合算法，进一步改善了种群的多样性和算法的效率。汪慎文等[8]考虑个体与发现者间的分布关系，提出了 GSO 中角色分配的策略，实验表明这种改进方法不仅提高了种群的多样性，而且加快了其收敛速度。

已有 GSO 算法及其变种的研究虽在一定程度上弥补了算法的缺陷，但算法的性能仍然存在亟待提高的需求和条件。首先，现实中的优化问题越来越多且日益复杂，需要设计性能更佳的智能优化算法来应对挑战；其次，智能计算领域不断涌现出新的学习机制和元启发式方法，这些新型优化机制具有其特点和优势，如果能将它们适当结合起来进行优势互补，则能克服单一算法(机制)所固有的局限性。

基于上述背景，针对 GSO 算法存在的问题，本文提出一种应用反向学习和差分进化的群搜索算法。OBDGSO 算法的主要特点包括：1)利用反向学习(Opposition-based Learning)机制产生反向群体，扩大了种群搜索的空间，提供了更多的机会来发现潜在的最优解，增加了群体的多样性，从而改善了算法全局勘探的能力；此外，基于反向种群和当前群体的精英选择策略能促使较好的个体参与下一代繁殖，有利于加快算法收敛。2)算法对种群中较好的解个体实施差分进化(Differential Evolution)操作，促使算法在较优解附近进行局部开采，有利于提高算法的求解精度。以上两种搜索模式相互协同并作用于 GSO 算法的每一代种群，以提高算法求解复杂优化问题的性能。

本文第 2 节介绍 GSO 算法的基本概念；第 3 节描述应用反向学习和差分进化的 GSO 算法；第 4 节给出实验仿真与分析；最后总结全文。

2　群搜索优化算法

GSO 算法模拟的是一些群居动物的觅食策略，其思想是在每一次迭代中，选择当前最优个体作为唯一的发现者，发现者根据动物的视觉扫描机制在最大转角范围内按照一定的方式调整视觉角度，在可视范围内寻找更好的资源信息[9]。算法随机选择部分个体充当加入者，加入者向发现者特定的方向前进。群体中剩余的个体则被选为游荡者，游荡者可在最大转角范围内调整自己的位置。如此反复，群体中 3 类个体相互作用，协同完成任务，从而驱使算法找到最优解。

群搜索算法中，一个 n 维的决策空间中，个体 i 在第 t 次迭代时的位置可表示为 $x_i^t\in\mathbb{R}^n$，它的搜索角度为 $\phi_i^t=(\phi_{i1}^t,\phi_{i2}^t,\cdots,\phi_{i(n-1)}^t)\in\mathbb{R}^{n-1}$，对应的搜索方向为 $D_i^t(\phi_i^t)=(d_{i1}^t,d_{i2}^t,\cdots,d_{in}^t)\in\mathbb{R}^n$，且搜索方向按照式(1)进行计算。

$$\begin{cases} d_{i1}^t=\prod_{q=1}^{n-1}\cos(\phi_{iq}^t) \\ d_{ij}^t=\sin(\phi_{i(j-1)}^t)\prod_{q=j}^{n-1}\cos(\phi_{iq}^t),1<j<n \\ d_{in}^t=\sin(\phi_{i(n-1)}^t) \end{cases} \tag{1}$$

2.1　发现者

GSO 算法在每次迭代时选择适应值最好的个体作为本次迭代过程的发现者。发现者在共享其信息资源的同时，继续搜索资源。其搜索的方法是假设发现者 p 在当前位置 x_p^t 处选择 3 个不同的角度进行视觉扫描，并观察到 3 个不同的位置 x_z，x_r 和 x_l，这 3 个位置的计算方法如式(2)所示。

$$\begin{cases} x_z=x_p^t+r_1 l_{\max} D_p^t(\phi_p^t) \\ x_r=x_p^t+r_1 l_{\max} D_p^t(\phi_p^t+r_2\theta_{\max}/2) \\ x_l=x_p^t+r_1 l_{\max} D_p^t(\phi_p^t-r_2\theta_{\max}/2) \end{cases} \tag{2}$$

其中，$r_1\sim\sigma(0,1)$，$r_2\in\mathbb{R}^{n-1}$ 是一个随机数序列，并且在 0 和 1 之间均匀分布。$l_{\max}$ 为视觉能扫描到的最大距离，$\theta_{\max}$ 为视觉能转动的最大角度。

如果要使 3 个随机位置(x_z，x_r 和 x_l)优于发现者当前的位置，则更新发现者 p 的位置和角度；反之，发现者的位置保持不变并通过式(3)更新角度。

$$\phi_p^{t+1}=\phi_p^t+r_2\alpha_{\max} \tag{3}$$

其中，$\alpha_{\max}$ 是最大搜索角度。

如果发现者经过连续 C 次迭代都没有发现更优的位置，则将它的角度重新转回 0°，如式(4)所示。

$$\phi^{t+C}=\phi^t \tag{4}$$

其中，C 为常数。

2.2　加入者

算法在每次迭代时，随机选择一定数量的剩余个体作为加入者，加入者 s 根据发现者共享的信息执行式(5)的动作来更新位置，但其角度不变。

$$x_s^{t+1}=x_s^t+r_3\circ(x_p^t-x_s^t) \tag{5}$$

其中，$r_3\in\mathbb{R}^n$ 是一个随机数序列，r_3 中各分量为区间[0,1]内均匀分布的随机数，“∘”表示 Hadamard 积。

算法迭代时如果出现某个加入者找到的位置比其他加入者和发现者所处的位置更优，则该加入者在下一次迭代时将会转换成发现者，而原来的发现者则可能转换成加入者或游荡者。

2.3　游荡者

在种群中的发现者和加入者确定后，余下的就是游荡者。

假设在第 t 次迭代中个体 i 成为游荡者，则游荡者 i 的随机角度为：

$$\phi_i^{t+1}=\phi_i^t+r_2\alpha_{\max} \tag{6}$$

为游荡者赋予一个随机距离 l_i，l_i 的计算式如式(7)所示。

$$l_i=ar_1 l_{\max} \tag{7}$$

此后，游荡者运动到一个新的位置，如式(8)所示。

$$x_i^{t+1}=x_i^t+l_i\cdot D_i^t(\phi_i^{t+1}) \tag{8}$$

2.4 GSO 算法流程

一个标准的 GSO 算法由 4 个基本步骤组成，即初始化、发现者个体的更新、加入者个体的更新和游荡者个体的更新。算法 1 给出了标准 GSO 算法的伪代码。

算法 1 标准 GSO 算法

1. 随机初始化所有个体的位置和角度，设置最大迭代次数为 T_{max}，并置迭代计数器 t=0
2. WHILE (t≤T_{max})
3. 计算个体的适应度，并进行 3 种角色的分配
4. 执行 2.1 节的发现者操作
5. 在种群中选择一定数量的加入者，并执行 2.2 节的加入者操作
6. 将种群中余下的个体选作游荡者，并执行 2.3 节的游荡者操作
7. t=t+1
8. END WHILE

群搜索算法将种群中的个体分为 3 类，每一类都有特定的分工。发现者用来探索较优位置周围的区域；加入者体现了种群的开采性；而游荡者体现了个体在搜索空间更大范围的随机搜索，有利于算法跳出局部极值。

3 OBDGSO 算法

3.1 反向学习机制

Tizhoosh[10]于 2005 年提出了反向学习的概念，并说明了反向解比当前解有高出近 50%的概率靠近全局最优解。其主要思想是在当前个体所在区域产生反向个体，并使反向个体与当前个体一起参与竞争，使优秀的个体进入下一代繁殖。反向学习已被证明是提高随机搜索算法的搜索能力的一种有效方法。下面给出反向学习的有关定义。

定义 1(反向解) 设在 D 维搜索空间中一个可行解为 $x_i=(x_{i1},x_{i2},\cdots,x_{iD})$，$x_{ij}\in[a_j,b_j]$，$j\in[1:D]$。设其反向解 $x_i'=(x_{i1}',x_{i2}',\cdots,x_{iD}')$ 且满足式(9)：

$$x_{ij}'=a_{ij}+b_{ij}-x_{ij} \tag{9}$$

定义 2(广义反向解) 令 $x_{ij}'=k(a_j+b_j)-x_{ij}$，$x_{ij}\in[a_j,b_j]$，$i=1,2,\cdots,|Popsize|$，$j=1,2,\cdots,D$。$|Popsize|$ 为种群规模，D 为搜索空间的维度。

定义 2 中的 k 可以取不同的实数，当 $k=0$ 时称其为基于解对称的广义反向学习；当 $k=0.5$ 时称其为基于对称区间的广义反向学习；当 $k=1$ 时称其为广义的反向学习；而当 k 为 $[0,1]$ 区间内的随机数时称其为随机广义反向学习。

定义 3(一般动态反向学习) 令 $x_{ij}'=k(da_j+db_j)-x_{ij}$，其中 da_j 和 db_j 分别表示当前代种群搜索空间中第 j 维上的最小值和最大值，即：

$$\begin{cases} da_j=\min(A_j) \\ db_j=\max(A_j) \end{cases} \tag{10}$$

其中，A_j 为种群中的个体在第 j 维上所有取值的集合，$k\in[0,1]$为一般化系数。

为验证反向学习的有效性，下面通过一个例子表明其具有较强的勘探能力[11]。假设待优化问题 $\min f(q)=\|q-A\|$，优化问题搜索空间的维度为 2，A 是优化问题的全局最优位置，并设其为(14,15)。假设决策变量 x_1，x_2 的取值范围是 $[0,20]$，在决策空间内取一点 $p=(9,8)$，其适应值 $f(p)=\sqrt{74}$，根据定义 1 可得其反向解点 $p'=(11,12)$，其适应度值 $f(p')=3\sqrt{2}$，显然，$f(p')<f(p)$，说明反向点更加接近最优解位置。

本文算法利用一般动态反向学习策略产生反向种群，并从当前种群和反向种群的并集中选择较好的个体组成临时种群。该学习机制的优点是既扩大了种群搜索的范围，同时又避免了无效搜索，使得进化的群体能较快地收敛于全局最优解。

3.2 差分进化

差分进化算法(Differential Evolution，DE)最初是由 Storn 等[12]于 1995 年提出的，它使用了与标准进化算法相似的计算步骤进行操作，但又不同于传统的进化算法(Evolutionary Algorithm，EA)。DE 利用随机选择的独立种群成员之差乘以一定系数得到的值作为变异算子来对当前种群成员进行扰动，并不只是用单独的概率分布来产生子代。DE 算法是一种简单的实参数优化算法，它通过对一些步骤进行简单循环来实现。图 1 给出了差分进化算法的主要步骤。

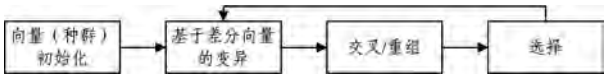


图 1 差分进化算法的主要步骤

本文利用差分进化变异策略对种群中较优的个体实施差分进化变异的操作，以实现在较优解附近的局部开采。其动机在于：通常种群中较优的解个体可能更加靠近全局最优解，因此在较优解附近进行搜索就有较大的机会发现全局最优解。差分进化算法主要有 3 个版本，这些版本的差别主要体现在变异方法上。根据变异个体的生成方式的不同，DE 的变异方式也不同，常见的变异方式有 DE/rand/1/bin，DE/best/1/bin 和 DE/ current-to-best/2/bin 3 种。本文采用 DE/current-to-best/2/bin 变异方式，设种群 $Pop=\{X_1,X_2,\cdots,X_{|Popsize|}\}$，则按式(11)进行变异操作：

$$v_{i,j}=x_{i,j}+\lambda*(x_{best,j}-x_{i,j})+F*(x_{r1,j}-x_{r2,j}) \tag{11}$$

其中，$r1\neq r2\neq r3\neq i$，$r1,r2,r3\in\{1,2,3,\cdots,N\}$；变异算子 $F\in[0,2]$是一个常数因子，控制偏差变量的缩放；λ 是一个附加的控制参数；$x_{best,j}$ 是当前种群最好个体的第 j 维分量。然后采用式(12)所示的交叉操作，生成新个体 $u_{i,j}$。

$$u_{i,j}=\begin{cases} v_{i,j}, & rand(0,1)\leqslant CR \text{ or } j=rand(1,D) \\ x_{i,j}, & \text{others} \end{cases} \tag{12}$$

其中，$CR\in[0,1]$为交叉概率，$rand(0,1)$为$[0,1]$之间均匀分布的随机数，$rand(1,D)$是$[1,D]$之间的一个随机整数，这种交叉可以保证 u_i 中至少有一个分量由 v_i 提供。本文利用差分进化方法对每一代的临时种群(临时种群的概念见 3.1 节)

中较优的前 50%的个体实施变异操作,其目的在于更好地平衡算法的全局勘探和局部开采能力。

3.3 OBDGSO 算法

在上述基础上,给出 OBDGSO 算法的流程,如算法 2 所示。

算法 2 OBDGSO 算法

1. 随机初始化规模为 N 的种群 pop,算法最大迭代次数为 T_{max},并置迭代计数器 t=0
2. WHILE ($t \leqslant T_{max}$)
3. 利用 3.1 节描述的一般动态反向学习策略产生当前种群 pop(t)的反向种群 obp(t),然后将当前种群 pop(t)和反向种群 obp(t)合并,并依据种群个体的适应度从合并后的群体中选择较优的 N 个个体组成临时种群 temp_pop(t)
4. 利用 3.2 节描述的差分进化方法对 temp_pop(t)中较优的前 50%的个体实施差分进化的变异操作,变异后的个体与 temp_pop(t)中较差的一半个体共同组成下一代种群
5. t=t+1
6. END WHILE
7. 输出最优解

4 实验与结果分析

4.1 对等比较算法

为了验证本文 OBDGSO 算法的有效性,选取 4 种代表性的群智能算法作为对等比较算法,它们分别是:1)基本的差分进化算法 DE[12];2)基本的群搜索算法 GSO[1];3)一种带精英反向学习的粒子群算法 EOPSO[13];4)基于正交实验设计的人工蜂群算法 ABC-OED[14]。

4.2 实验参数与环境

为使实验结果具有公正性,所有对等比较算法的种群规模均为 20,各算法最大迭代次数均为 1000,参与对比的算法的其他参数取其相应文献中建议的值。文献[13]的研究表明,一般化系数 $k=1$ 时算法能获得较好的效果,本文的 OBDGSO 算法中 k 亦取值为 1。另外,OBDGSO 算法的其他参数的取值情况为:$\varphi^0=(\pi/4,\cdots,\pi/4)$,$a=round(\sqrt{n+1})$,$\theta_{max}=\pi/a^2$,$\alpha_{max}=\theta_{max}/2$,$l_{max}=\|U-L\|=\sqrt{\sum_{i=1}^{n}(U_i-L_i)^2}$。

为减少性能分析中随机因素的影响,每种算法在所有的测试函数上均独立运行 30 次。本文的仿真实验在 Think Pad X200 笔记本电脑上运行,电脑配置 5GB 内存和 2.4GHz 双核 CPU,安装 Windows 7 X64 操作系统,算法运用 Java 编程,在 JDK1.7 环境下编译运行。

4.3 测试函数

针对单目标优化问题,选取了 12 个代表性的测试函数,它们均参考于 CEC2010 特别报告[15]。表 1 列出了测试函数的定义。

表 1　12 个基准测试函数

测试函数	函数表达式	变量范围	决策空间维度	函数最优值
Sphere	$f_1(x)=\sum_{i=1}^{D}x_i^2$	[−100,100]	30	0
Schwefel's	$f_2(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	[−10,10]	30	0
Quadric	$f_3(x)=\sum_{i=1}^{D}(\sum_{j=1}^{i}x_j)^2$	[−100,100]	30	0
Step	$f_4(x)=\sum_{i=1}^{D}(\lvert x_i+0.5\rvert)^2$	[−100,100]	30	0
Quadric Noise	$f_5(x)=\sum_{i=1}^{D}i*x_i^4+random[0,1)$	[−1.28,1.28]	30	0
Rastrgin	$f_6(x)=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$	[−5.12,5.12]	30	0
Non-Rastrgin	$f_7(x)=\sum_{i=1}^{D}[y_i^2-10\cos(2\pi y_i)+10]$ $y_i=\begin{cases}x_i, & \lvert x_i\rvert<0.5\\ round(2x_i)/2, & \text{else}\end{cases}$	[−5.12,5.12]	30	0
Ackley	$f_8(x)=-20\exp(-0.2\sqrt{1/D\sum_{i=1}^{D}x_i^2})-\exp(1/D\sum_{i=1}^{D}\cos(2\pi x_i))+20+e$	[−32,32]	30	0
Griewank	$f_9(x)=1/4000\sum_{i=1}^{D}x_i^2-\prod\cos(x_i/\sqrt{i})+1$	[−600,600]	30	0
伸展 V 型正弦函数	$f_{11}(x)=\sum_{i=1}^{D-1}(x_{i+1}^2+x_i^2)^{0.25}\{\sin^2[50(x_{i+1}^2+x_i^2)^{0.1}]+1\}$	[−10,10]	30	0
Easom's	$f_{18}(x)=-\cos(x_1)\cos(x_2)\exp(-[(x_1-\pi)^2+(x_2-\pi)^2])$	[−100,100]	2	0
Six-hump	$f_{19}(x)=(4-2.1x_1^2+x_1^{4/3})x_1^2+x_1x_2+(-4+4x_2^2)x_2^2$	[−1,1]	2	−1.0316

4.4 实验结果与分析

通过比较 5 种算法在 12 个测试函数上的均值和方差来评估算法的性能。其中,均值(mean)和标准差(std.)是算法在同一测试问题上独立运行 30 次的统计结果;t-test 值是 OBDGSO 算法与其他对等比较算法在同一测试函数上进行 t-检验时的 t 值;"+""−"和"="表示本文算法获得的结果在显著性水平为 5%的双尾 t 检验中分别优于、劣于和等于对应列的对等算法在对应行的测试问题上的显著性区分结果;"Score"表示本文算法显著优于对应列的对等算法在 12 个测试问题中的净胜得分,即得"+"与得"−"的测试问题个数之差。同时,采用粗体字表示所有对比算法在每一个测试问题中获得的最优值。表 2 列出了 5 种算法在 12 个测试函数上获得的统计结果。

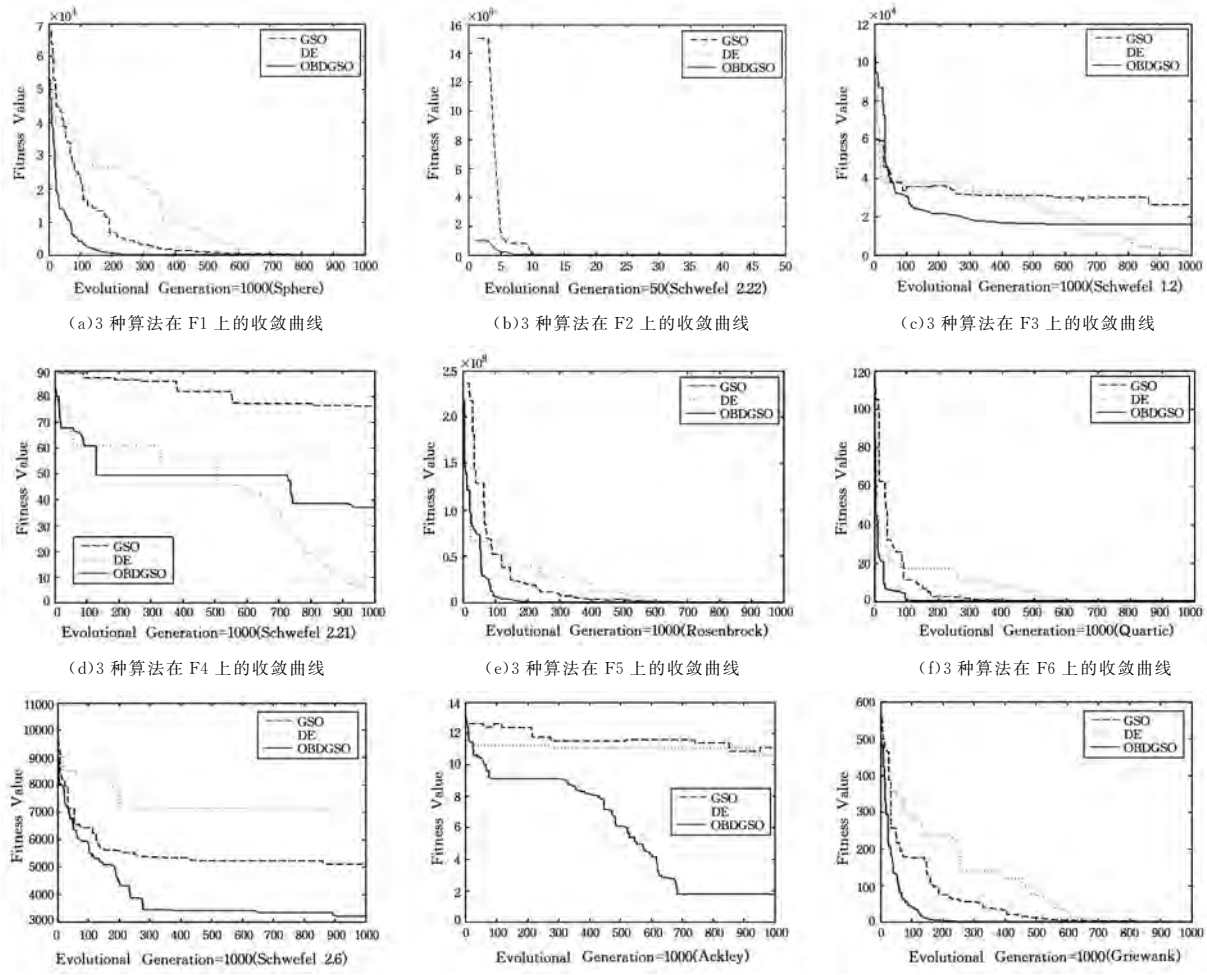
表 2 5 种算法在 12 个测试函数上获得的结果

测试函数		OBDGSO	基本 DE	EOPSO	基本 GSO	ABC-OED
F1	mean	**0.00E+0.00**	1.00E−320	5.00E−100	5.80E−115	1.46E−36
	std.	0.00E+0.00	0.00E+0.00	7.35E−210	3.30E−200	3.75E−80
	t-test		+	+	+	+
F2	mean	**0.00E+0.00**	2.28E−30	2.45E−170	3.23E−08	1.68E−09
	std.	0.00E+0.00	1.10−50	0.00E+1.00	7.11E−15	1.23E−15
	t-test		+	+	+	+
F3	mean	1.55E−05	3.05E+02	1.53E−12	1.34E+01	**1.57E−14**
	std.	7.25E−09	2.35E+04	3.21E−23	2.19E+00	1.08E−28
	t-test		+	−	+	−
F4	mean	**0.00E+0.00**	9.16E−17	8.28E−24	6.85E−35	2.76E−10
	std.	0.00E+0.00	6.14E−30	1.80E−45	1.33E−68	2.15E−18
	t-test		+	+	+	+
F5	mean	3.88E−16	4.25E−15	4.66E+02	**0.00E+0.00**	1.44E−30
	std.	6.25E−31	5.65E−16	6.58E+04	0.00E+0.00	3.33E−50
	t-test		+	+	−	−
F6	mean	**6.68E−06**	7.66E−05	4.28E−03	2.38E−03	1.18E−02
	std.	7.77E−11	5.82E−09	3.56E−06	1.48E−06	1.35E−05
	t-test		+	+	+	+
F7	mean	1.22E−14	1.24E−14	1.48E−14	**9.00E−15**	1.38E−14
	std.	2.35E−28	2.35E−28	4.09E−24	1.87E−30	2.18E−28
	t-test		=	+	−	=
F8	mean	**9.253E−06**	5.58E−03	9.81E−02	9.29E−01	9.66E−01
	std.	3.44E−10	1.66E−06	1.62E−03	8.63E−04	1.85E−03
	t-test		+	+	+	+
F9	mean	**2.28E−94**	2.64E−78	5.68E−21	7.40E−16	2.59E−14
	std.	6.14E−201	1.41E−154	8.96E−40	5.81E−30	1.15E−29
	t-test		+	+	+	+
F10	mean	3.61E−21	8.56E−17	**3.60E−90**	3.91E−80	2.64E−56
	std.	5.66E−40	5.13E−32	0.00E+0.00	6.92E−39	1.41E−40
	t-test		+	−	=	−
F11	mean	**2.53E−29**	2.74E−10	1.25E−23	9.07E−03	8.57E−03
	std.	9.19E−57	2.28E−18	4.61E−45	5.86E−07	2.37E−07
	t-test		+	+	+	+
F12	mean	**2.14E+00**	7.35E+00	4.24E+00	2.72E+00	3.20E+00
	std.	2.50E−01	3.15E+00	6.75E−01	2.90E−01	2.63E−01
	t-test		+	+	+	+
Better(+)			11	10	9	8
Same(=)			1	0	1	1
Worse(−)			0	2	2	3
Score			11	8	8	5

从表 2 可以看出，本文的 OBDGSO 算法在 F1，F2，F4，F6，F8，F9，F11 和 F12 等 8 个基准的测试函数上获得了最优的均值；EOPSO 算法在 F10 函数上获得了最优的均值；基本 GSO 算法在 F5 和 F7 两个测试函数上获得了最优的均值；ABC-OED 算法在 F3 函数上获得了最优的均值；而基本 DE 算法在所有 12 个测试函数上均未能获得最优的均值。

从表 2 的 *t*-检验结果来看，本文的 OBDGSO 算法对比基本 DE 算法获得的净胜分为 11 分，对比 EOPSO 算法和基本 GSO 算法获得的净胜分均为 8 分，OBDGSO 算法对比 ABC-OED 算法获得的净胜分为 5 分。因此，本文的 OBDGSO 算法较其他 4 种群智能优化算法具有显著性的优势，OBDGSO 算法在求解的精度和算法的稳定性方面均优于其他 4 种对比算法。究其原因，OBDGSO 算法使用反向学习机制扩展了种群探索的范围，增加了算法发现全局最优解的概率；另外，本文采用差分进化变异操作，在种群较优解附近进行局部开采，有利于提高算法的解题度。这两种策略有机结合并相互协调，有效地平衡了算法的全局勘探和局部开采能力，改善了算法在复杂优化问题中的解题性能。

进一步地，为检测 OBDGSO 算法的收敛速度，选取基本的 DE 算法和 GSO 算法作为对比算法，考察 3 种对等比较在 9 个测试函数(F1－F9)上前 1000 代获得的求解结果，通过观察各算法获得结果的变化趋势可以直观地判断出算法收敛速度的快慢。需要说明的是，选取 DE 算法和 GSO 算法作为对比算法的原因在于：OBDGSO 算法综合利用了 DE 变异策略和 GSO 的基本搜索模式，本文算法与这两种基本算法进行实验比较，有利于判定 OBDGSO 算法的有效性；此外，图 2 所使用的数据均来自各算法在 30 次独立运行中首次获得的结果。从图 2(a)可以看出，本文的 OBDGSO 算法较基本 DE 算法和基本 GSO 算法能以较快的速度收敛，即 OBDGSO 算法运行近 200 代时就接近于收敛状态，其次是基本 GSO 算法，收敛性表现较差的是基本 DE 算法。从图 2(b)可以看出，OBDGSO 算法在仅执行 8 次迭代左右即接近收敛状态，反映了本文算法在 F2 函数上较好的收敛性能。本文算法在 F3，F5，F6，F7，F8 和 F9 等 6 个测试函数上较基本 DE 算法和基本 GSO 算法的收敛速度亦是最好的。综合图 2 可知，本文的 OBDGSO 算法在 9 个测试函数上都表现出了较好的收敛性，其收敛性能明显优于基本 DE 算法和基本 GSO 算法。究其原因，OBDGSO 算法一方面通过对当前种群和反向种群的并集执行精英选择策略，加快了算法的收敛速度；另一方面，对种群中较好的个体执行差分进化的变异操作，促进算法在较优解附近进行局部开采，这些策略有利于算法较快地发现全局最优解。



(a)3 种算法在 F1 上的收敛曲线　(b)3 种算法在 F2 上的收敛曲线　(c)3 种算法在 F3 上的收敛曲线

(d)3 种算法在 F4 上的收敛曲线　(e)3 种算法在 F5 上的收敛曲线　(f)3 种算法在 F6 上的收敛曲线

(g)3 种算法在 F7 上的收敛曲线　(h)3 种算法在 F8 上的收敛曲线　(i)3 种算法在 F9 上的收敛曲线

图 2　3 种算法在 9 个测试函数上的收敛曲线

结束语　群搜索算法在解决一些高维多峰优化问题时面临收敛速度慢、求解精度不高以及容易陷入局部极值等问题，本文将一般动态反向学习机制和差分进化的变异操作引入到 GSO 算法中，以改善 GSO 算法在求解复杂优化问题时的性能。实验结果表明，应用反向学习和差分进化的群搜索算法的性能显著优于其他 4 种对等比较算法，说明本文的 OBDGSO 算法较好地弥补了标准 GSO 算法的不足，因此 OBDGSO 算法是一种有前途的群智能优化算法。

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 Congress on Evolutionary Computation. 2006：1272-1278.

[2] SAUNDERS J R，LI X. Application of a group search optimization based artificial neural network to machine condition monitoring[C]// The 13th IEEE International Conference on Emerging Technologies and Factory Automation. Hamburg，2008：15-18.

[3] TANG W J，LI M S，HE S，et al. Optimal power flow with dynamic loads using bacterial foraging algorithm[C]// International Conference on Power Systems Technology. 2006，10：22-26.

[4] 李丽娟，徐小通，刘锋. 基于群智能的群搜索算法及其在离散变量设计中的应用[J]. 钢结构，2008(增刊)：592-596.

[5] 李丽娟，张雯雯，徐小通，等. 改进的群搜索优化算法及其应用[J]. 空间结构，2016，16(2)：13-24.

[6] 庞艳娟. 混合群搜索优化算法及其应用[D]. 太原：太原科技大学，2010.

[7] 易卜拉欣. 基于文化框架的群搜索和粒子群的混合算法及其应用[D]. 上海：华东理工大学，2014.

[8] 汪慎文，丁立新，谢承旺，等. 群搜索优化算法中角色分配策略的研究[J]. 小型微型计算机系统，2012，33(9)：1938-1943.

[9] 汪慎文，丁立新，谢大同，等. 应用反向学习策略的群搜索优化算法[J]. 计算机科学，2012，39(9)：183-187.

[10] TIZHOOSH H. Opposition-based learning：A new scheme for machine intelligence[C]// Proceedings of the International Conference on Computational Intelligence for Modeling Control and Automation. 2005：695-701.

[11] 王立平，谢承旺. 一种带反向学习机制的自适应烟花爆炸算法[J]. 计算机科学，2016，43(11A)：103-107.

[12] STORN R，PRICE K. Differential evolution：A simple and efficient adaptive scheme for global optimization over continuous spaces：Technical Report TR-95-012[R]. ICSI，USA，1995.

[13] 周新宇，吴志健，王晖，等. 一种精英反向学习的粒子群优化算法[J]. 电子学报，2013，11(8)：1647-1652.

[14] 周新宇，吴志健，王明文. 基于正交实验设计的人工蜂群算法[J]. 软件学报，2015，26(9)：2167-2190.

[15] TANG K，LI X D，SUGANTHAN P N，et al. Benchmark Functions for the CEC's 2010 Special Session and Competition on Large-Scale Global Optimization[D]. Hefei：Nature Inspired Computation and Applications Laboratory，USTC，2009.