文章编号: 1001-0920(2019)10-2073-12 DOI: 10.13195/j.kzyjc.2018.0146

改进的灰狼优化算法及其高维函数和FCM优化

张新明[1,2†], 王 霞[1], 康 强[1]

(1. 河南师范大学 计算机与信息工程学院，河南 新乡 453007；
2. 河南省高校计算智能与数据挖掘工程技术研究中心，河南 新乡 453007)

摘 要: 灰狼优化算法(GWO)具有较强的局部搜索能力和较快的收敛速度,但在解决高维和复杂的优化问题时存在全局搜索能力不足的问题. 对此,提出一种改进的GWO,即新型反向学习和差分变异的GWO(ODGWO). 首先,提出一种最优最差反向学习策略和一种动态随机差分变异算子,并将它们融入GWO中,以便增强全局搜索能力;然后,为了很好地平衡探索与开采能力以提升整体的优化性能,对算法前、后半搜索阶段分别采用单维操作和全维操作形成ODGWO;最后,将ODGWO用于高维函数和模糊C均值(FCM)聚类优化. 实验结果表明,在许多高维Benchmark函数(30维、50维和1 000维)优化上,ODGWO的搜索能力大幅度领先于GWO,与state-of-the-art优化算法相比,ODGWO具有更好的优化性能. 在7个标准数据集的FCM聚类优化上, 与GWO、GWOepd和LGWO相比,ODGWO表现出了更好的聚类优化性能,可应用在更多的实际优化问题上.

关键词: 智能优化算法；灰狼优化算法；反向学习；差分变异；模糊C均值(FCM)聚类；高维函数优化

中图分类号: TP181 **文献标志码:** A

Improved grey wolf optimizer and its application to high-dimensional function and FCM optimization

ZHANG Xin-ming[1,2†], *WANG Xia*[1], *KANG Qiang*[1]

(1. College of Computer and Information Engineering，Henan Normal University，Xinxiang 453007，China；
2. Engineering Technology Research Center for Computing Intelligence & Data Mining of Henan Province，Xinxiang 453007，China)

Abstract: The grey wolf optimizer (GWO) algorithm proposed recently has strong local search ability and fast convergence, but it has some defects, such as poor global search ability in solving high-dimensional functions and complex optimization problems. Therefore, this paper proposes an improved GWO, namely GWO with opposition-learning and differential mutation (ODGWO). Firstly, a max-min opposition learning strategy and a dynamical and random differential mutation operator are proposed, which are integrated with GWO to enhance its global search ability. Then, in order to balance exploration and exploitation well to improve the overall performance, one-dimensional operation and full-dimensional operation are applied to the first half and the latter one of the search phase respectively to form the ODGWO consequently. Finally, the ODGWO is used for the high-dimensional function and fuzzy C-means (FCM) clustering optimization. The experimental results on many high-dimensional (30, 50 and 1 000 dimension) benchmark functions show that the ODGWO has significantly higher global search ability than the GWO does, and the ODGWO outperforms the state-of-the-art algorithms. As for FCM optimization on seven standard datasets, the ODGWO shows better clustering optimization performance compared with the GWO, the GWOepd and the LGWO, and it will be applied to more real-world optimization problems.

Keywords: intelligent optimization algorithm；grey wolf optimization；opposition-learning；differential mutation；fuzzy C-means (FCM) clustering；high-dimensional function optimization

0 引 言

群智能优化算法受启发于自然界生物的社会行为等,相比于传统优化算法,其结构简单,易于实现,从而受到广泛关注. 2014年,Mirjalili等提出了一种新型的群智能优化算法——灰狼优化算法(Grey wolf optimizer, GWO)[1]. GWO模拟了自然界中灰狼的社会等级和狩猎行为. 该算法因采用新型的搜索机制,在解决一些优化问题时表现出良好的性能,如

收稿日期: 2018-01-29；**修回日期:** 2018-03-22.
基金项目: 河南省高等学校重点科研项目(19A520026).
责任编委: 巩敦卫.
†通讯作者. E-mail: xinmingzhang@126.com.

局部搜索能力较强,收敛速度较快等等. 但GWO在解决一些复杂问题时仍存在全局搜索能力较差等缺陷. 因此,许多学者对其进行了改进研究. 文献[2]提出了融合动态进化种群(Evolutionary population dynamics, EPD)的GWO(GWOepd),提高了GWO的探索能力. 文献[3]提出了一种基于Powell局部优化方法的GWO,并将该算法用于聚类分析. 文献[4]将差分进化算法的交叉和变异算子引入GWO中,提出一种混合GWO,提高了GWO的优化性能. 文献[5]提出了一种基于混沌理论和精英反向学习策略的混合GWO(Hybrid GWO, HGWO),用于解决高维函数优化问题. 文献[6]将Levy flight和贪婪选择策略与改进的狩猎阶段相结合,提出一种嵌入Levy flight的GWO(Levy-embedded GWO, LGWO),提高了GWO的种群多样性. 文献[7]提出了一种生物地理学优化算法与GWO的混合算法,以帮助GWO跳出局部最优点. 文献[8]以强化学习的方式设置个体行为,以神经网络的方式学习个体历史经验,提高了GWO的优化性能,从而提出一种Experienced GWO. 文献[9]将位置更新方程和非线性控制参数策略相结合,提出了一种提高探索能力的GWO. 以上算法很好地解决了GWO存在的一些问题,但GWO提出时间较短,有许多改进和应用拓展之处,尤其在解决一些复杂优化问题时,仍存在很大的改进空间.

目前,高维优化问题一直是进化计算领域的一个焦点[5],因为近年来许多现实世界的问题都涉及多个变量的优化. 然而,维数的增加会导致大多数可用优化算法的性能下降. 因此,传统的智能优化算法已经无法有效地解决该问题. 模糊C均值(Fuzzy C-means, FCM)聚类[10]是一种无监督的学习方法,FCM聚类的最终目标是划分到同一组内数据点的相似性最大,不同组间数据点的相异性最大. FCM聚类使用模糊目标函数来评估聚类结果,模糊目标函数的值越低表示聚类效果越好. FCM聚类作为一种软聚类算法已被广泛应用于多个领域[10],但仍存在对初始聚类中心敏感、易陷于局部最优等问题.

本文针对GWO在解决高维和复杂的优化问题时存在的全局搜索能力不足的问题,提出最优最差反向学习策略与动态随机差分变异策略,构建一种基于这两种策略的新型灰狼优化算法(GWO with opposition-learning and differential mutation, ODGWO). 首先,为了增加GWO的全局搜索能力,嵌入最优最差反向学习策略; 其次,为了进一步增强算法的全局搜索能力,融入动态随机差分变异策略;最后,为了平衡探索与开采能力,搜索前期采用单维操作,搜索后期采用全维操作,实现两种操作的优势互补. 为了验证ODGWO处理高维和复杂优化问题的能力,测试其在30维、50维和1 000维的高维函数优化和FCM聚类优化的性能,结果表明,与GWO以及多个state-of-the-art算法相比,ODGWO在高维函数优化问题上取得了较好的结果,并能更好地处理FCM聚类优化问题.

1 灰狼优化算法

GWO是受灰狼群的社会等级和狩猎行为启发而提出的算法[1,11]. 在自然界中,灰狼群的社会等级分为4个级别,由上至下依次为α狼、β狼、δ狼和ω狼. 狼群的狩猎行为主要分为3个步骤,分别是跟踪并靠近猎物、追踪包围猎物和攻击猎物. 包围行为的数学模型为

$$\boldsymbol{D} = |\boldsymbol{C}\boldsymbol{X}_p(t) - \boldsymbol{X}(t)|, \tag{1}$$

$$\boldsymbol{X}(t+1) = \boldsymbol{X}_p(t) - \boldsymbol{A}\boldsymbol{D}. \tag{2}$$

其中:t为当前迭代标示; $\boldsymbol{A}$和$\boldsymbol{C}$为系数向量,其具体取值见文献[1]; $\boldsymbol{X}$和$\boldsymbol{X}_p$分别为灰狼和猎物的位置向量. 攻击猎物行为的数学模型为

$$\boldsymbol{D}_\alpha = |\boldsymbol{C}_1\boldsymbol{X}_\alpha - \boldsymbol{X}|, \tag{3}$$

$$\boldsymbol{D}_\beta = |\boldsymbol{C}_2\boldsymbol{X}_\beta - \boldsymbol{X}|, \tag{4}$$

$$\boldsymbol{D}_\delta = |\boldsymbol{C}_3\boldsymbol{X}_\delta - \boldsymbol{X}|, \tag{5}$$

$$\boldsymbol{X}(t+1) = \frac{\boldsymbol{X}_\alpha - \boldsymbol{A}_1\boldsymbol{D}_\alpha + \boldsymbol{X}_\beta - \boldsymbol{A}_2\boldsymbol{D}_\beta + \boldsymbol{X}_\delta - \boldsymbol{A}_3\boldsymbol{D}_\delta}{3}. \tag{6}$$

其中:式(3)~(5)分别给出了ω狼与α狼、β狼、δ狼之间的距离, $\boldsymbol{X}_\alpha$、$\boldsymbol{X}_\beta$和$\boldsymbol{X}_\delta$分别表示α狼、β狼和δ狼的位置向量;式(6)给出了ω狼的位置更新方式,两个参数$\boldsymbol{A}$和$\boldsymbol{C}$在灰狼狩猎过程中起着关键作用,共同决定了GWO的探索与开采阶段[11]. 当$|\boldsymbol{A}| > 1$时,GWO更强调探索能力,当$|\boldsymbol{A}| < 1$时,GWO更强调开采能力,$\boldsymbol{C}$的取值是为了能随机增加($|\boldsymbol{C}| > 1$)或减轻($|\boldsymbol{C}| < 1$)灰狼靠近猎物的难易程度.

算法1 GWO伪代码.

1) 随机初始化$\boldsymbol{N}$头灰狼的位置,初始化参数$\boldsymbol{A}$和$\boldsymbol{C}$

2) 计算每头灰狼的适应度值

3) 初始化$\boldsymbol{X}_\alpha, \boldsymbol{X}_\beta, \boldsymbol{X}_\delta$,且令$t = 0$

4) while $t <$ MaxDT

5) for i to N do

6) 按照式(3)～(6)更新第i头灰狼的位置

7) end for

8) 计算每头灰狼的适应度值

9) 贪心算法更新$\boldsymbol{X}_\alpha, \boldsymbol{X}_\beta, \boldsymbol{X}_\delta$并保存,更新$\boldsymbol{A}$和$\boldsymbol{C}$

10) $t = t + 1$

11) end while

12) return $\boldsymbol{X}_\alpha$

GWO伪代码如算法1所示,其中t表示当前迭代次数,N表示种群大小,MaxDT表示最大迭代次数.

用GWO与粒子群优化算法(PSO)进行对比,GWO具有以下优势: 1)搜索方式独特,所有灰狼均依赖3头不同位置的最优灰狼更新其位置,即所有灰狼朝着3个最优点趋近,而PSO中所有粒子向着两个最优位置趋近,故在解决一些简单优化问题时,GWO比PSO具有更好的全局搜索能力. 2)探索与开采平衡方式不同,GWO采用$\boldsymbol{A}$和$\boldsymbol{C}$两个参数向量进行平衡,故能够获得更快的收敛速度和更强的局部搜索能力[1]. 3)每次迭代只保留更新后的α狼、β狼和δ狼,而PSO保留N个历史最优位置向量和速度向量,GWO具有更低的空间复杂度. 4) PSO采用迭代贪心算法进行更新,迭代贪心算法指的是每次得到一个新解之后,对其计算适应度值,将新解与原解进行对比,新解优于原解则替换原解,加快了收敛速度,一般更适用于单峰优化问题,但对于多峰优化问题,会使算法易陷于局部最优. GWO更新所有灰狼位置之后才计算其适应度值,然后采用贪心算法更新α狼、β狼和δ狼的位置. 与迭代贪心算法相比,GWO的贪心算法使得适应度值的计算可采用并行计算方式以便提高运行速度. 5) GWO原理简单,参数少,易于操作和实现. 另外,与采用单维操作的人工蜂群算法[12]相比,GWO采用全维操作更新灰狼位置等,具有较高的搜索效率.

2　改进的灰狼优化算法(ODGWO)

2.1　GWO存在的问题及改进的动机

主要问题: 由式(6)可知,ω狼在α狼、β狼和δ狼的带领下更新自身位置,当α狼、β狼和δ狼均处于局部最优时,狼群中的每一头狼因受这3头狼的影响,有可能趋近于局部最优,种群多样性降低;因此,GWO在解决复杂优化问题时,易陷于局部最优,全局搜索能力不足.

为解决该问题,ODGWO在GWO上进行了一系列改进: 1)为避免算法陷入局部最优和增加全局搜索能力,引入新型的反向学习策略. 2)为进一步增加全局搜索能力,动态融合随机差分变异策略,借助差分变异增加种群的多样性. 3)依据现代智能优化理论,在处理优化问题时,需追求探索和开采的平衡. 因此,在算法搜索前半阶段采用有助于探索的单维操作,后半阶段采用有助于开采的全维操作,以平衡探索和开采,且能够做到单维和全维操作的优势互补,从而从整体上提升算法的优化性能.

2.2　最优最差反向学习策略

Tizhoosh[13]在2005年提出反向学习的概念,其目的在于避免算法陷入局部最优. 许多改进的优化算法引入了反向学习策略,如文献[5]将精英反向学习策略融入到GWO中. 本文提出了一种最优最差反向学习策略,并嵌入到GWO中.

最优最差反向学习策略是将一般反向学习机制与随机反向学习机制相融合. 主要思想是寻找狼群中的某一可行解的反向解[14],从而更好地引导狼群寻找到全局最优解. 当GWO陷入局部最优时,当前种群中的最优灰狼得不到有效的更新,因此对其采用一般反向学习机制,即

$$\boldsymbol{X}_{\mathrm{best}}(t+1) = \boldsymbol{a} + (\boldsymbol{b} - \boldsymbol{X}_{\mathrm{best}}(t)). \tag{7}$$

其中: $\boldsymbol{X}_{\mathrm{best}}$表示当前全局最优灰狼位置向量,$\boldsymbol{a}$和$\boldsymbol{b}$分别表示灰狼位置下边界向量和上边界向量.

为增加最差灰狼的全局搜索能力,采用随机反向学习机制,即

$$\boldsymbol{X}_{\mathrm{worst}}(t+1) = \boldsymbol{a} + \mathrm{rand}(\boldsymbol{b} - \boldsymbol{X}_{\mathrm{worst}}(t)). \tag{8}$$

其中: $\boldsymbol{X}_{\mathrm{worst}}$表示当前全局最差灰狼位置向量,rand表示在区间[0, 1]中的均匀分布随机数.

与文献[5]中精英反向学习策略相比,最优最差反向学习策略主要有如下特点:

1) 执行对象不同. 精英反向学习策略的执行对象是当前种群中前m个精英个体,而本文中一般反向学习策略和随机反向学习策略的执行对象分别是当前种群中的最优灰狼和最差灰狼(两个个体). 原因在于最优灰狼若陷入局部最优,则可能引导其他灰狼陷入局部最优,因此采用一般反向学习策略使其有机会跳出局部最优;而最差灰狼处于最差的位置,因此采用随机反向学习得到一个反向最差灰狼的随机位置,更加提高了种群的多样性,增加了获得全局最优解的概率. 采用两个个体进行反向学习既有利于避免算法陷于局部最优,又不至于影响收敛速度,因为过多

的个体采用反向学习虽然能增加种群的多样性,但会影响局部搜索能力.

2) 参数含义不同. 精英反向学习策略中的$\boldsymbol{a}$和$\boldsymbol{b}$值分别表示m个精英个体位置每一维上的最小值和最大值,每次迭代需要计算它们,如此增加了计算复杂度.而本文反向学习策略中的$\boldsymbol{a}$和$\boldsymbol{b}$值分别表示灰狼种群位置的下边界和上边界,每次迭代不变,无需计算.

3) 无参数调整. 精英反向学习需要确定精英个数m,需要在实验中多次调整,而本文的新型反向学习策略无需设置参数.

2.3 动态随机差分变异策略

为进一步解决GWO在解决复杂优化问题时全局搜索能力不足的问题,在2.2节的基础上又融入一种新型的差分变异策略,即动态随机差分变异策略. 这种策略来自于差分进化算法的典型差分变异思想[4],先随机选择不同的3个个体,然后缩放2个个体的差分向量,随后将缩放的差分向量与第3个个体结合以实现差分扰动,以便增强全局搜索能力. 本文首先采用如下所示的差分变异策略:

$$\boldsymbol{X}_i(t+1)=\boldsymbol{X}_{r_1}(t)+F(\boldsymbol{X}_{r_2}(t)-\boldsymbol{X}_{r_3}(t)). \quad (9)$$

其中:$\boldsymbol{X}_{r_1}$、$\boldsymbol{X}_{r_2}$和$\boldsymbol{X}_{r_3}$是从狼群中随机选择的3头灰狼的位置向量,满足r_1、r_2和$r_3\in[1,N]$,且$r_1\neq r_2\neq r_3\neq i$. 为进一步增强全局搜索能力,采用随机缩放参数,形成随机差分变异策略,即设置缩放因子为随机值$F=0.5+0.5\,\mathrm{rand}$.

将随机差分变异策略与GWO搜索方式交替执行,实现了动态随机差分变异策略.

算法2 随机差分变异策略与GWO的动态融合.

1) if rand$<P_d$

2) 按照式(9)执行随机差分变异策略

3) else

4) 按照式(3)~(6)执行GWO的更新策略

5) end

其中:P_d是随机差分变异策略的选择概率,其值采用从1到0的线性动态递减方式,即

$$P_d=1-t/\mathrm{MaxDT}. \quad (10)$$

由式(10)可知,搜索前期P_d的值较大,主要采用随机差分变异策略,有利于提高探索能力;在搜索后期P_d的值较小,主要采用GWO的搜索方式,有利于提高开采能力. 由算法2可知,与GWO更新方式(式(3)~(6))相比,随机差分变异策略仅使用式(9)进行位置更新,其更新表达式简单,计算量较小,故计算复杂度较小. 算法2是两种策略的动态融合,故与GWO相比,从整体上算法2降低了计算复杂度.

动态随机差分变异策略有如下主要特点: 1)其执行对象是当前种群中除最优灰狼和最差灰狼之外的所有灰狼,而不是当前种群中的全部个体. 2)随机差分变异策略与GWO搜索方式以概率P_d随机选择形式交替进行,有利于提高优化性能. 3)差分变异策略采用随机缩放参数F,无需人为的调整,可操作性增强;且由于F的随机性,从而整体上增加了种群多样性,进一步增强了算法的全局搜索能力.

2.4 单维和全维分段操作

为了平衡算法的探索与开采能力,根据算法的迭代次数将搜索过程分为两个阶段,前半搜索阶段(即迭代次数的取值范围为[1, MaxDT/2])采用单维操作,后半搜索阶段采用全维操作. 文献[12]对单维和全维操作的优化性能进行了对比实验,结果表明单维操作的优势在于:具有较高的种群多样性,有助于提高算法的全局搜索能力,且单维操作是解决不可分优化问题的一种有效操作. 但是,单维操作具有以下缺陷: 1)因只选择一维进行更新,故搜索效率相对较低; 2)因随机性较强,收敛速度较慢. 全维操作的优势在于:对全部维度进行更新,搜索效率较高,收敛速度较快. 因此,本文提出了单维与全维分段操作的方式. 算法的前半搜索阶段更强调全局搜索能力,因此采用单维操作. 单维操作即在当前灰狼的位置向量上随机选择一维执行更新操作. 为了弥补单维操作的缺陷,在后半搜索阶段更需强调开采能力,即采用全维操作进行优势互补. 全维操作即对当前灰狼的位置向量的每一维都进行更新操作.

将最优最差反向学习策略和动态随机差分变异策略融合到GWO中,并在算法前半搜索阶段采用单维操作,后半搜索阶段采用全维操作,如此构建ODGWO,其算法流程如图1所示.

由图1可以看出: 1)在算法中嵌入最优最差反向学习策略和动态随机差分变异策略,增加了种群多样性,有利于避免算法陷入局部最优,从而提高全局搜索能力; 2)采用单维全维分段操作,有助于更好地平衡探索与开采能力和两种操作的优势互补; 3)保留了GWO的优势,即收敛速度快和所有个体更新过后再计算适应度值,这种方式有助于并行计算目标函数值以提高运行速度.

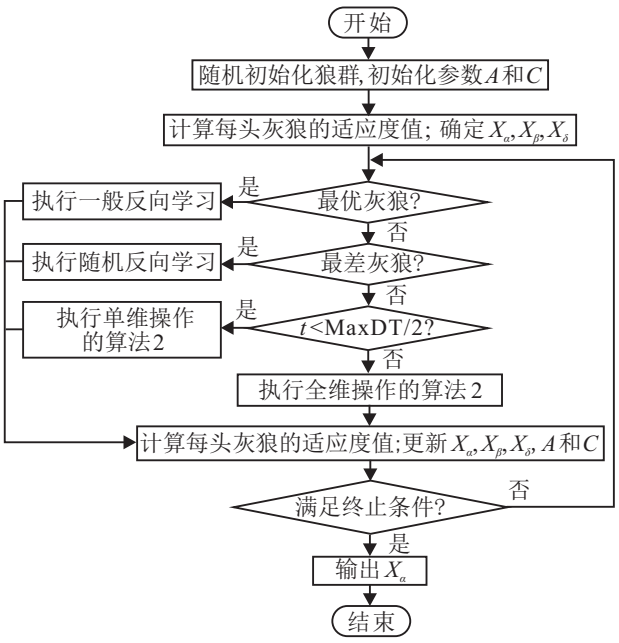

图 1　ODGWO流程

3　函数优化仿真实验及结果分析

为了验证ODGWO的性能,本节采用两组实验: 实验1是ODGWO在24个30维和50维Benchmark函数上的优化实验; 实验2是其在10个1 000维Benchmark函数上的优化实验. 实验环境采用Windows 7操作系统,3.10 GHz CPU和4 GB内存的PC,编程语言为Matlab R2014a.

3.1　在30维和50维函数上优化性能比较

3.1.1　测试函数及对比算法

表1给出了24个Benchmark函数的信息,包含6个单峰函数($f_1 \sim f_6$)、6个多峰函数($f_7 \sim f_{12}$)、6个平移函数($f_{13} \sim f_{18}$)和6个旋转函数($f_{19} \sim f_{24}$),其中Range表示函数定义域,Fmin表示函数的理论最优值. 单峰函数用于评估算法的开采能力,多峰函数用于评估算法的探索能力,平移和旋转函数用于评估算法解决复杂优化问题的能力. 本节选取GWO[1]和6种state-of-the-art算法(GWOepd[2]、MEABC[15]、CSPSO[16]、SinDE[17]、RLPSO[18]和DELLU[19])作对比. MEABC(Multi-strategy ensemble ABC)是一种多策略集成的人工蜂群算法. CSPSO(PSO using crisscross search)是一种改进粒子群算法: 交叉搜索的粒子群优化算法. SinDE(Sinusoidal differential evolution)是正弦差分进化算法,该算法使用新的正弦表达式来自动调整差分进化算法主要参数(缩放因子和交叉概率)的值,从而促进算法探索能力与开采能力之间的平衡. RLPSO(Reverse-learning and local-learning PSO)是一种具备反向学习和局部学习能力的粒子群优化算法. DELLU(Differential evolution algorithm based on local lipschitz underestimate supporting hyperplanes)是在差分进化算法的框架下,结合Lipschitz估计理论提出的一种基于局部Lipschitz下界估计支撑面的差分进化算法.

3.1.2　与GWO、GWOepd、MEABC、CSPSO和SinDE性能对比

为了公平起见,对GWO、GWOepd、MEABC、CSPSO、SinDE和ODGWO设置相同的基本参数: 种群大小$N = 20$,独立运行次数$\mathrm{Num} = 30$,当维度$D = 30$时最大迭代次数$\mathrm{MaxDT} = 2\,500$,当$D = 50$时$\mathrm{MaxDT} = 3\,500$,其他参数设置同其相应的参考文献. 同时,为更好地显示ODGWO与5种对比算法之间的差异性,本文采用Wilcoxon符号秩和检验法[20]进行差异性检验. Wilcoxon符号秩和检验法主要采用成对检验,目的在于检验两个算法之间的显著性差异. 表2、表3分别给出了6种算法在24个Benchmark函数上30维和50维的结果对比以及ODGWO与5种对比算法之间的Wilcoxon符号秩和检验结果. 其中包含均值(Mean)、标准差(Std)及排名(Rank),排名依据为: 均值越小,排名越高,如果均值相同,则标准差越小排名越高;如果二者相同,则排名相同. 最优者用下划线标注. "$n/w/t/l$"分别表示测试的函数个数、ODGWO优于(win)对比算法的函数个数、ODGWO等于(tie)对比算法的函数个数和ODGWO劣于(lose)对比算法的函数个数. 置信水平$\alpha = 0.05$.

由表2的实验结果可知,与GWO相比,ODGWO在10个Benchmark单峰和多峰函数上的优化结果均优于GWO,且在函数f_7和f_{11}上ODGWO与GWO获得相同的结果. 从均值和标准差来看,在单峰函数上,GWO在5个函数上的均值仅次于ODGWO,并且优于其余4种对比算法,说明GWO的局部搜索能力强于4种对比算法,即开采能力较强. 但在多峰函数上,GWO仅在函数f_7和f_{11}上与ODGWO同时取得排名第1,其他多峰函数上的均值劣于ODGWO,说明GWO的全局搜索能力有待提高. 在平移和旋转函数上,ODGWO的优化性能大幅度领先,尤其在f_{16}、f_{17}和$f_{19} \sim f_{23}$上,ODGWO的结果优势明显,这也说明GWO解决复杂优化问题的能力较差,表明了在GWO中引入新型搜索策略是必要的. 由Wilcoxon符号秩和检验结果可知,ODGWO优于、等于和劣于GWO的个数分别是22、2和0,说明ODGWO在24个函数上表现出的优化性能明显优于GWO. 与GWOepd、MEABC、CSPSO和SinDE相比,从排名来看,ODGWO获得了19次第1名,GWOepd、MEABC、

表 1 Benchmark测试函数信息表

Function	Formulation	Range	Fmin
Sphere	$f_1(x)=\sum_{i=1}^{D}x_i^2$	$[-100,100]^D$	0
Tablet	$f_2(x)=10^6x_1^2+\sum_{i=2}^{D}x_i^2$	$[-100,100]^D$	0
Schwefel2.22	$f_3(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	$[-10,10]^D$	0
Schwefel1.2	$f_4(x)=\sum_{i=1}^{D}\Big(\sum_{j=1}^{i}x_j\Big)^2$	$[-100,100]^D$	0
Zakharow	$f_5(x)=\sum_{i=1}^{D}x_i^2+\Big(\sum_{i=1}^{D}0.5ix_i\Big)^2+\Big(\sum_{i=1}^{D}0.5ix_i\Big)^4$	$[-5,10]^D$	0
Rosenbrock	$f_6(x)=\sum_{i=1}^{D-1}[100(x_i^2-x_{i+1})^2+(x_i-1)^2]$	$[-10,10]^D$	0
Girewank	$f_7(x)=\frac{1}{4\,000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\Big(\frac{x_i}{\sqrt{i}}\Big)+1$	$[-600,600]^D$	0
Ackley	$f_8(x)=20+e-20\exp\Big(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\Big)-\exp\Big(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\Big)$	$[-32,32]^D$	0
Schwefel2.26	$f_9(x)=418.982\,887\,272\,433\,69D-\sum_{i=1}^{D}x_i\sin(\sqrt{\lvert x_i\rvert})$	$[-500,500]^D$	0
Rastrigin	$f_{10}(x)=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$	$[-5.12,5.12]^D$	0
Sum of different power	$f_{11}(x)=\sum_{i=1}^{D}\lvert x_i\rvert^{(i+1)}$	$[-1,1]^D$	0
Exponential	$f_{12}(x)=\exp\Big(0.5\sum_{i=1}^{D}x_i\Big)-1$	$[-1.28,1.28]^D$	0
Shift Sphere	$f_{13}(x)=\sum_{i=1}^{D}z_i^2-450,\ z=x-o$	$[-100,100]^D$	−450
Shift Schwefel2.21	$f_{14}(x)=\max_{i=1}^{D}\{\lvert z_i\rvert\}-450,\ z=x-o$	$[-100,100]^D$	−450
Shift Rosenbrock	$f_{15}(x)=\sum_{i=1}^{D-1}[100(z_i^2-z_{i+1})^2+(z_i-1)^2]+390,\ z=x-o+1$	$[-100,100]^D$	390
Shift Rastrigin	$f_{16}(x)=\sum_{i=1}^{D}[z_i^2-10\cos(2\pi z_i)+10]-330,\ z=x-o$	$[-5.12,5.12]^D$	−330
Shift Girewank	$f_{17}(x)=1+\sum_{i=1}^{D}\Big(\frac{z_i^2}{4\,000}\Big)-\prod_{i=1}^{D}\cos\Big(\frac{z_i}{\sqrt{i}}\Big)-180,\ z=x-o$	$[-600,600]^D$	−180
Shift Ackley	$f_{18}(x)=20+e-20\exp\Big[-\frac{1}{5}\sqrt{\frac{1}{D}\sum_{i=1}^{D}z_i^2}\Big]-\exp\Big[\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)\Big]-140$ $z=x-o$	$[-32,32]^D$	−140
Rotated Sphere	$f_{19}(x)=\sum_{i=1}^{D}z_i^2,\ z=xM$	$[-100,100]^D$	0
Rotated Elliptic	$f_{20}(x)=\sum_{i=1}^{D}(10^6)^{\frac{i-1}{D-1}}z_i^2,\ z=xM$	$[-100,100]^D$	0
Rotated Rosenbrock	$f_{21}(x)=\sum_{i=1}^{D}[100(z_i^2-x_i+1)^2+(z_i-1)^2],\ z=xM$	$[-2.048,2.048]^D$	0
Rotated Rastrigin	$f_{22}(x)=\sum_{i=1}^{D}[y_i^2-10\cos(2\pi y_i)+10],\ y=xM$	$[-5.12,5.12]^D$	0
Rotated Ackley	$f_{23}(x)=20+e-20\exp\Big(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}y_i^2}\Big)-\exp\Big(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi y_i\Big)$ $y=xM$	$[-32,32]^D$	0
Rotated Girewank	$f_{24}(x)=\frac{1}{4\,000}\sum_{i=1}^{D}y_i^2-\prod_{i=1}^{D}\cos\Big(\frac{y_i}{\sqrt{i}}\Big)+1,\ y=xM$	$[-600,600]^D$	0

表 2 6种算法在30维函数上的优化结果

Fun		ODGWO	GWO	GWOepd	MEABC	CSPSO	SinDE
f_1	Mean	0	1.475 3e-130	1.095 2e-92	2.606 1e-30	2.940 0e-53	1.992 6e-13
	Std	0	4.067 5e-130	5.909 5e-92	3.207 6 e-30	1.536 2e-52	6.908 4e-13
	Rank	1	2	3	5	4	6
f_2	Mean	0	3.913 6e-130	1.153 9e-94	1.139 1e-29	1.560 9e-41	1.234 0e-06
	Std	0	9.884 1e-130	4.574 5e-94	1.673 2e-29	6.612 3e-41	6.757 6e-06
	Rank	1	2	3	5	4	6
f_3	Mean	0	5.411 4e-76	2.610 3e-55	2.040 8e-16	3.773 2e-33	2.705 7e-11
	Std	0	1.164 8e-75	4.658 5e-55	9.491 4e-17	7.856 0e-33	1.480 4e-10
	Rank	1	2	3	5	4	6
f_4	Mean	0	1.368 6e-34	6.550 0e-18	1.118 0e+04	3.416 5e-04	4.891 7e+02
	Std	0	5.887 0e-34	3.031 4e-17	2.091 7e+03	5.270 3e-04	3.213 5e+02
	Rank	1	2	3	6	4	5
f_5	Mean	0	1.802 2e-37	5.541 0e-14	5.070 8e+02	9.799 1e-05	1.029 8e+01
	Std	0	8.161 8e-37	1.801 4e-13	5.651 8e+01	1.116 1e-04	5.003 7e+00
	Rank	1	2	3	6	4	5
f_6	Mean	1.683 2e-02	2.682 2e+01	2.727 3e+01	2.885 7e+00	2.217 4e+01	6.118 3e+01
	Std	2.729 3e-02	9.108 5e-01	8.739 2e-01	4.328 8e+00	8.820 7e+00	4.838 8e+01
	Rank	1	4	5	2	3	6
f_7	Mean	0	0	2.345 6e-03	6.581 3e-06	0	7.405 9e-04
	Std	0	0	5.374 1e-03	2.589 2e-05	0	2.256 4e-03
	Rank	1	1	6	4	1	5
f_8	Mean	3.256 7e-15	7.756 8e-15	8.585 7e-15	5.169 2e-14	3.730 3e-15	4.516 7e-08
	Std	8.030 4e-15	2.412 0e-15	2.999 1e-15	7.772 4e-15	1.655 9e-15	2.163 3e-07
	Rank	1	3	4	5	2	6
f_9	Mean	8.281 7e-09	6.620 3e+03	6.054 7e+03	3.553 2e+01	5.145 4e+02	1.270 6e+02
	Std	3.948 1e-08	8.751 9e+02	1.720 7e+03	6.336 3e+01	1.115 4e+03	1.133 3e+02
	Rank	1	6	5	2	4	3
f_{10}	Mean	2.368 5e-16	2.352 5e-01	1.257 9e-01	4.643 1e-01	1.090 3e-06	9.515 1e+00
	Std	1.014 9e-15	1.288 5e+00	6.890 0e-01	6.780 1e-01	4.873 8e-06	3.125 9e+00
	Rank	1	4	3	5	2	6
f_{11}	Mean	0	0	0	1.640 3e-61	1.536 7e-48	2.663 3e-1
	Std	0	0	0	5.010 9e-61	5.398 4e-48	1.451 1e-10
	Rank	1	1	1	4	5	6
f_{12}	Mean	0	2.115 7e-06	2.657 2e-06	3.763 4e-06	1.598 6e-05	2.318 1e-13
	Std	0	1.634 7e-06	3.557 9e-06	4.854 8e-06	1.707 0e-05	5.201 1e-13
	Rank	1	3	4	5	6	2
f_{13}	Mean	2.743 6e-12	2.995 0e+03	3.187 8e+00	1.402 1e-13	1.179 5e-08	2.340 3e-04
	Std	3.264 7e-12	3.487 4e+03	1.173 5e+00	2.884 3e-14	1.774 4e-08	1.281 9e-03
	Rank	2	6	5	1	3	4
f_{14}	Mean	3.437 9e+00	1.750 2e+01	9.695 8e-01	1.060 9e+01	3.570 6e+00	2.440 9e+01
	Std	1.681 3e+00	9.196 1e+00	3.788 1e-01	1.650 3e+00	1.698 0e+00	1.427 2e+01
	Rank	2	5	1	4	3	6
f_{15}	Mean	4.591 7e+02	3.218 8e+08	1.087 1e+03	2.695 0e+01	5.607 3e+02	1.185 6e+02
	Std	4.136 2e+01	3.510 3e+08	5.408 2e+02	2.414 6e+01	7.316 5e+02	9.903 5e+01
	Rank	3	6	5	1	4	2
f_{16}	Mean	1.993 6e-10	1.304 0e+02	7.671 8e+01	6.301 4e-01	1.360 7e+02	8.192 2e+00
	Std	2.466 4e-10	3.397 8e+01	2.228 0e+01	7.147 7e-01	3.002 4e+01	2.499 3e+00
	Rank	1	5	4	2	6	3
f_{17}	Mean	3.975 8e-07	2.710 6e+01	8.759 4e-01	3.299 5e-04	4.047 2e-03	1.478 6e-03
	Std	2.101 4e-06	2.198 8e+01	1.275 9e-01	1.800 0e-03	8.560 6e-03	3.578 9e-03
	Rank	1	6	5	2	4	3
f_{18}	Mean	2.149 6e-06	1.109 2e+01	2.226 9e+00	3.514 8e-13	3.432 4e-05	1.806 5e-07
	Std	1.413 3e-06	3.117 8e+00	6.269 3e-01	6.354 6e-14	2.489 3e-05	7.503 4e-07
	Rank	3	6	5	1	4	2
f_{19}	Mean	0	4.420 7e-115	5.152 7e-86	2.401 1e-06	1.766 8e-52	1.200 3e-06
	Std	0	1.877 7e-114	1.134 7e-85	5.350 0e-06	9.169 6e-52	6.481 5e-06
	Rank	1	2	3	6	4	5
f_{20}	Mean	1.347 8e-26	3.737 8e+05	9.675 4e+05	1.198 7e+07	1.993 1e+05	4.091 8e+06
	Std	3.878 4e-26	5.322 0e+05	1.257 7e+06	2.924 5e+06	2.280 9e+05	1.786 7e+06
	Rank	1	3	4	6	2	5
f_{21}	Mean	2.773 5e+01	2.870 6e+01	2.874 4e+01	3.250 6e+03	2.837 4e+01	2.883 4e+03
	Std	5.823 9e-01	4.623 8e-02	3.422 6e-02	3.024 3e+03	1.974 1e-02	7.213 8e+03
	Rank	1	3	4	5	2	6
f_{22}	Mean	2.368 5e-16	4.531 4e+01	3.561 8e+01	2.509 3e+02	9.552 1e+01	5.740 5e+01
	Std	7.712 5e-16	1.338 1e+01	3.956 6e+01	3.857 3e+01	3.157 4e+01	1.453 3e+01
	Rank	1	3	2	6	5	4
f_{23}	Mean	1.528 6e+01	2.102 9e+01	2.102 4e+01	2.094 7e+01	2.102 3e+01	2.104 1e+01
	Std	8.841 5e+00	4.487 3e-02	5.764 8e-02	6.374 1e-02	4.733 9e-02	3.974 8e-02
	Rank	1	5	4	2	3	6
f_{24}	Mean	3.700 7e-18	5.364 6e-03	0	4.539 3e-01	0	4.940 6e-02
	Std	2.027 0e-17	2.938 3e-02	0	1.605 0e-01	0	5.963 8e-02
	Rank	3	4	1	6	1	5
P-value			0.001	0.001	0.001	0.001	0.001
n/*w*/*t*/*l*			24/22/2/0	24/21/1/2	24/21/0/3	24/22/1/1	24/22/0/2

表 3　6种算法在50维函数上的优化结果

Fun		ODGWO	GWO	GWOepd	MEABC	CSPSO	SinDE
f_1	Mean	0	2.187 6e-140	2.490 0e-104	3.460 7e-23	8.990 9e-49	2.561 5e-06
	Std	0	5.487 1e-140	5.384 3 e-104	3.870 4e-23	4.247 6e-48	1.402 2e-05
	Rank	1	2	3	5	4	6
f_2	Mean	0	1.991 5e-140	4.327 9e-104	4.934 7e-23	1.100 6e-37	2.184 0e-04
	Std	0	4.473 4e-140	1.308 9e-103	3.132 7e-23	2.243 8e-37	1.196 1e-03
	Rank	1	2	3	5	4	6
f_3	Mean	0	3.796 1e-82	5.460 1e-61	8.397 7e-13	1.162 3e-33	9.078 3e-10
	Std	0	6.475 2e-82	8.096 9e-61	3.363 2e-13	4.218 9e-33	2.431 6e-09
	Rank	1	2	3	5	4	6
f_4	Mean	0	5.944 7e-22	2.305 9e-08	3.422 5e+04	6.117 5e-02	5.596 8e+03
	Std	0	3.242 6e-21	1.228 6e-07	4.139 8e+03	7.725 1e-02	2.133 1e+03
	Rank	1	2	3	6	4	5
f_5	Mean	0	8.143 8e-28	2.419 8e-05	1.085 5e+03	2.083 2e-02	1.178 5e+02
	Std	0	3.113 9e-27	8.441 0e-05	1.108 2e+02	1.624 3e-02	2.892 9e+01
	Rank	0	2	3	6	4	5
f_6	Mean	5.284 6e-02	4.711 0e+01	4.724 5e+01	3.177 1e+00	3.775 3e+01	8.241 2e+01
	Std	6.897 0e-02	7.964 6e-01	7.786 7e-01	3.585 2e+00	1.726 4e+01	3.322 9e+01
	Rank	1	4	5	2	3	6
f_7	Mean	7.401 5e-18	0	3.710 8e-04	1.136 7e-10	0	3.675 1e-04
	Std	4.054 0e-17	0	2.032 5e-03	6.217 4e-10	0	1.804 9e-03
	Rank	3	1	6	4	1	5
f_8	Mean	7.519 9e-15	1.083 6e-14	1.237 5e-14	9.141 2e-12	5.743 6e-15	6.278 5e-05
	Std	8.265 4e-15	2.972 4e-15	2.272 6e-15	1.308 7e-11	2.029 8e-15	2.529 2e-04
	Rank	2	3	4	5	1	6
f_9	Mean	1.290 8e-06	1.194 1e+04	1.255 9e+04	5.527 1e+01	7.690 1s+03	9.407 1s+02
	Std	2.727 1e-06	1.170 1e+03	3.196 9e+03	9.191 8e+01	2.511 6s+03	3.927 2s+02
	Rank	1	5	6	2	4	3
f_{10}	Mean	1.184 2e-16	0	6.837 9e-02	4.974 8e-01	3.166 3e-02	4.010 7e+01
	Std	4.506 8e-16	0	3.745 3e-01	6.788 5e-01	1.734 3e-01	8.378 7e+00
	Rank	2	1	4	5	3	6
f_{11}	Mean	0	0	0	2.324 6e-53	9.798 8e-41	9.343 7e-08
	Std	0	0	0	8.775 3e-53	4.909 0e-40	5.069 9e-07
	Rank	1	1	1	4	5	6
f_{12}	Mean	0	1.527 5e-06	1.820 1e-06	1.496 3e-06	1.201 7e-05	4.649 6e-12
	Std	0	1.847 5e-06	1.585 5e-06	1.276 2e-06	1.250 8e-05	1.831 4e-11
	Rank	1	4	5	3	6	2
f_{13}	Mean	1.041 4e-08	9.512 2e+03	1.479 0e+01	2.880 1e-13	3.179 8e-06	3.914 2e-07
	Std	1.313 7e-08	5.089 4e+03	4.008 5e+00	1.828 3e-13	2.049 6e-06	2.079 5e-06
	Rank	2	6	5	1	4	3
f_{14}	Mean	1.272 2e+01	4.246 5e+01	3.019 7e+00	3.555 9e+01	1.107 0e+01	4.710 0e+01
	Std	4.276 2e+00	7.670 8e+00	6.334 5e-01	3.420 1e+00	2.606 6e+00	1.248 4e+01
	Rank	3	5	1	4	2	6
f_{15}	Mean	5.039 1e+01	1.666 3e+09	3.107 1e+04	1.051 8e+01	3.403 0e+02	2.022 5e+03
	Std	3.134 5e+01	1.451 7e+09	2.398 1e+04	7.917 2e+00	4.188 9e+02	7.274 9e+03
	Rank	2	6	5	1	3	4
f_{16}	Mean	3.316 8e-02	3.055 9e+02	2.238 0e+02	7.628 0e-01	2.262 5e+02	3.595 2e+01
	Std	1.816 5e-01	4.994 2e+01	7.327 3e+01	8.540 3e-01	6.868 9e+01	6.640 3e+00
	Rank	1	6	4	2	5	3
f_{17}	mean	4.881 6e-07	1.087 7e+02	1.137 4e+00	1.151 0e-03	1.242 3e-03	1.966 2e-03
	Std	1.039 4e-06	4.765 3e+01	3.530 4e-02	3.018 8e-03	3.2859e-03	6.6163e-03
	Rank	1	6	5	2	3	4
f_{18}	Mean	7.618 9e-05	1.439 0e+01	3.634 5e+00	2.525 0e-11	1.200 6e-03	2.425 7e-04
	Std	4.812 8e-05	2.167 0e+00	6.212 3e-01	1.547 3e-11	2.194 7e-03	1.194 5e-03
	Rank	2	6	5	1	4	3
f_{19}	Mean	0	1.403 3e-125	7.375 5e-97	4.116 8e-03	2.339 1e-39	6.236 8e-02
	Std	0	6.638 6e-125	2.812 9e-96	6.680 2e-04	1.007 9e-38	2.065 1e-01
	Rank	1	2	3	5	4	6
f_{20}	Mean	4.006 7e-27	3.550 7e+05	7.820 6e+05	2.866 3e+07	2.687 2e+04	1.207 9e+07
	Std	1.627 3e-26	7.275 6e+05	1.346 7e+06	3.104 2e+06	4.554 0e+04	3.894 1e+06
	Rank	1	3	4	6	2	5
f_{21}	Mean	4.826 4e+01	4.858 3e+01	4.863 7e+01	2.771 4e+04	4.820 9e+01	1.336 9e+04
	Std	1.779 9e-01	4.478 2e-02	4.543 7e-02	6.569 0e+04	2.290 9e-02	3.462 7e+04
	Rank	2	3	4	6	1	5
f_{22}	Mean	0	1.122 4e+02	5.275 1e+01	6.130 9e+02	2.048 3e+02	1.273 0e+02
	Std	0	2.274 4e+01	1.481 1e+01	5.280 2e+01	9.355 4e+01	2.779 5e+01
	Rank	1	3	2	6	5	4
f_{23}	Mean	1.820 7e+01	2.119 2e+01	2.117 8e+01	2.109 7e+01	2.119 6e+01	2.117 8e+01
	Std	7.277 3e+00	4.374 2e-02	5.175 0e-02	5.015 5e-02	3.821 9e-02	3.949 8e-02
	Rank	1	5	4	2	6	3
f_{24}	Mean	2.176 4e-03	2.641 7e-02	0	5.845 1e-01	0	2.329 1e-01
	Std	1.192 0e-02	7.124 3e-02	0	1.617 2e-01	0	4.994 9e-01
	Rank	3	4	1	6	1	5
P-value			0.001	0.001	0.001	0.019	0.001
n/*w*/*t*/*l*			24/21/1/2	24/21/1/2	24/21/0/3	24/18/0/6	24/23/0/1

CSPSO和SinDE分别获得3次、3次、2次和0次第1名,表明ODGWO在30维的Benchmark函数上具有较好的优化性能. 由Wilcoxon符号秩和检验结果可知,ODGWO优于GWOepd、MEABC、CSPSO和SinDE的个数分别为21、21、22和22,说明ODGWO在24个函数上明显优于这4种对比算法.

由表3的数据可知,与GWO相比,从均值和标准差来看,ODGWO得到与30维类似的结果. 由Wilcoxon符号秩和检验结果可知,ODGWO优于、等于和劣于GWO的个数分别是21、1和2,同样说明ODGWO在50维函数上明显优于GWO. 与GWOepd、MEABC、CSPSO和SinDE相比,在单峰和多峰函数上,ODGWO在函数$f_1 \sim f_5$、f_{11}和f_{12}上表现出了最好的优化性能,得到了理论最优值0.在平移和旋转函数上,ODGWO在函数f_{19}和f_{22}上表现出了最优的性能,取得了理论最优值0. 在函数f_{16}、f_{17}、f_{20}和f_{23}上, ODGWO虽然没有取得最优值0,但是其优化性能远远优于其他4种算法. 从排名来看,ODGWO共获得15次第1名,GWOepd、MEABC、CSPSO和SinDE分别获得3次、3次、4次和0次第1名,说明ODGWO在50维Benchmark函数上同样具有较好的优化性能. 由Wilcoxon符号秩和检验结果可知,ODGWO优于GWOepd、MEABC、CSPSO和SinDE 的个数分别为21、21、18和23,也说明ODGWO在24个函数上明显优于这4种优秀的对比算法.

另外,由表2、表3可以看出,因f_6是不可分函数,只有采用单维处理的MEABC和ODGWO获得了较好的结果,足以证明单维操作在此类优化问题上的优势所在. 由Wilcoxon符号秩和检验结果可知,无论是在30维还是在50维,P-value值均小于0.05,说明ODGWO与5种对比算法之间存在显著性差异.

总的来说,ODGWO在不同维度的4种类型Benchmark函数上获得了较为满意的结果,验证了ODGWO能很好地解决GWO存在的问题,同时表明了本文提出的最优最差反向学习策略、动态随机差分变异策略以及单维全维分段操作是有效的.

3.1.3　运行时间对比及时间复杂度分析

为了考查ODGWO的运行速度,对GWO、SinDE、GWOepd、MEABC、CSPSO和ODGWO的运行时间进行了测试,图2给出了这6种算法在30维和50维上的平均运行时间结果(单位s).

由图2可以看出,在30维和50维两种情况下ODGWO的耗时最少,运行速度最快,其在30维函数上的平均耗时分别是GWO、GWOepd、MEABC、CSPSO和SinDE的52%、48%、26%、48%和27%,在50维函数上的平均耗时是其他算法的1/3至1/2.

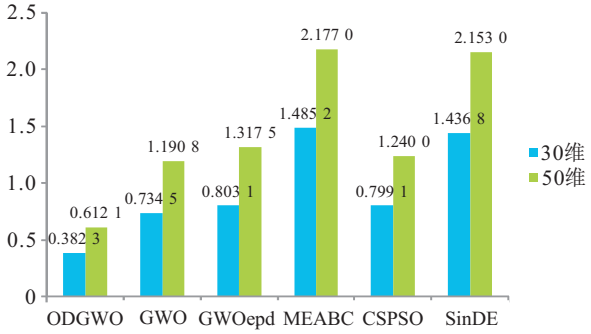

图2　30维和50维平均运行时间对比

对ODGWO的计算复杂度进行分析. 一般而言,算法的计算复杂度主要体现在两个方面: 目标函数的计算复杂度和算法本身的计算复杂度. 以一次迭代为例,ODGWO的运行速度快的原因主要有两个: 1) 虽然6种算法的最大函数评价次数(maxFEs)相同,但SinDE和MEABC采用了串行目标函数评价方式,在每得到一个新种群位置之后就对其进行目标函数评价,这样串行的评价方式较为耗时;而ODGWO使用了并行目标函数评价方式,对N个新解同时进行目标函数评价,从而提高了运行速度,节省了运行时间; 2) GWO、GWOepd和CSPSO虽然也采用了并行的函数评价方式,但算法的整个搜索阶段采用了全维操作,当对某一候选解向量进行更新时,需要通过循环更新其中的每一维,其时间复杂度为$O(D)$,运行时间相对稍长. 另外,ODGWO采用算法2的搜索方式,正如2.3节所述,算法2的计算复杂度少于GWO和GWOepd. 而ODGWO在前半搜索阶段采用单维操作,当对某一灰狼的位置向量进行更新时,只更新其中的一维,其时间复杂度为$O(1)$. MEABC虽然也采用了单维操作,但正如原因1)所知,MEABC采用了串行的目标函数评价方式,且采用多种策略协同处理,所以MEABC的运行时间较长. 虽然ODGWO的更新方式融入了随机差分变异策略,增加了一些判断和计算步骤,但没有额外增加循环步骤,而对算法的一半搜索阶段使用单维操作和目标函数并行计算使计算复杂度大幅度降低,从而整体上节省了算法的运行时间. 上述分析也验证了图2平均运行时间的对比结果.

3.1.4　与RLPSO和DELLU性能对比

为了进一步验证ODGWO的性能,将其分别与RLPSO[18]和DELLU[19]进行对比. RLPSO与ODGWO均使用了反向学习策略,DELLU与ODGWO均采用了差分变异方式更新种群,选用

这两种算法进行对比是因为它们具有较强的可比性. RLPSO和DELLU的实验数据分别取自于文献[18]和文献[19],由于不同文献使用的函数测试集不同,只取ODGWO与这两种算法共同的测试函数上的结果进行对比.为使结果对比可靠,保证ODGWO的maxFEs总是小于两种对比算法,即ODGWO优化条件更苛刻. 它们在30维函数上的结果对比如表4所示,其中表4的第1行括号中的数据是对应的maxFEs, NA表示相应的参考文献没有提供该数据.

表4 ODGWO与RLPSO、DELLU的结果对比

Fun	ODGWO(5e+04)		RLPSO(1e+05)		DELLU(6e+04)	
	Mean	Std	Mean	Std	Mean	Std
f_1	0	0	1.23e-13	2.82e-13	8.98e-21	5.23e-21
f_2	0	0	NA	NA	2.65e-12	1.71e-12
f_3	0	0	9.02e-08	2.92e-08	8.12e-11	3.06e-11
f_4	0	0	NA	NA	1.07e+01	6.38e+00
f_5	0	0	NA	NA	1.74e+00	2.38e+00
f_6	1.68e-02	2.73e-02	2.03e+02	5.86e+02	3.89e-02	3.49e-02
f_7	0	0	1.63e-02	1.98e-02	3.60e-08	2.13e-08
f_8	3.26e-15	8.03e-15	4.99e-08	2.66e-08	4.24e-05	4.24e-05
f_9	8.28e-09	3.95e-08	4.91e+03	7.08e+02	0	0
f_{10}	2.37e-16	1.01e-15	1.56e+00	1.37e+00	4.06e-08	2.84e-08
f_{11}	0	0	1.97e-26	9.34e-26	NA	NA

由表4可以看出,与RLPSO相比,在ODGWO的maxFEs更少的前提下,其在所有函数上的优化性能大幅度优于RLPSO. 与DELLU相比,在maxFEs更少的前提下,除了在f_9上ODGWO的结果稍差于DELLU,在其他9个函数上,ODGWO的结果明显优于DELLU. 这说明将动态随机差分进化策略和最优最差反向学习方案等有机融合到GWO中是有效的,既发挥了差分变异和反向学习增强全局搜索能力的优点,又承袭了GWO较强的局部搜索能力的优势.

3.2 在1 000维函数上优化性能比较

为了验证ODGWO在更高维函数优化上的性能,本节给出了ODGWO在1 000维Benchmark函数上的优化测试结果. 选取文献[5]中的GWO、HGWO、GSA、PSO的实验数据直接作为对比数据,为公平起见,设置ODGWO与文献[5]中的maxFEs相同,均为5e+04,独立运行次数相同$\mathrm{Num}=30$,10个相同的Benchmark函数. 在这4种对比算法中,HGWO和ODGWO中均嵌入了反向学习策略,同为GWO的改进算法,更具有可比性.

表5给出了ODGWO与GWO、HGWO、GSA和PSO在1 000维的优化结果对比. 由表5可知: 与GWO相比,ODGWO在10个1 000维函数上的优化结果更优; 与最先进的GWO改进算法HGWO相比,在除f_{10}外的9个函数上,ODGWO取得了更好的结果,ODGWO也大幅度优于经典的GSA和PSO. 综上所述,ODGWO适用于求解高维函数的优化问题,也说明本文对GWO的改进是可行的.

表5 5种算法在1 000维函数上的优化结果

Fun		ODGWO	GWO	HGWO	GSA	PSO
f_1	Mean	0	6.14e-01	4.62e-10	5.73e+04	1.37s+04
	Std	0	2.58e-01	2.73e-10	3.43s+03	9.31s+02
f_2	Mean	0	4.46e+00	1.81e-05	4.36s+04	1.15s+03
	Std	0	4.60e+00	1.00e-05	3.88s+03	2.61s+01
f_3	Mean	6.20e-03	6.97e+01	5.77e-01	2.14e+01	3.09e+01
	Std	2.61e-02	2.29e+00	7.03e-02	9.34e-01	1.96e+00
f_4	Mean	2.07e+01	4.08e+03	9.96e+02	6.02e+06	9.35e+07
	Std	2.75e+01	3.61e+03	0.00e+00	5.60e+05	8.60e+06
f_5	Mean	0	2.02e+02	8.49e+00	6.03e+04	1.33e+04
	Std	0	2.30e+00	2.75e+00	2.61e+03	6.54e+02
f_6	Mean	7.12e-04	1.44e-02	6.17e-03	1.32e+03	2.43e+05
	Std	4.58e-04	5.30e-03	3.53e-03	1.41e+02	5.08e+03
f_7	Mean	8.29e-16	7.89e+01	3.95e-07	4.60e+03	1.29e+04
	Std	2.46e-15	1.64e+01	2.52e-07	1.19e+02	4.08e+02
f_8	Mean	3.70e-14	7.79e-02	5.45e-07	8.42e+00	1.37e+01
	Std	2.13e-14	2.03e-02	7.08e-08	7.70e-02	4.23e-01
f_9	Mean	6.29e-17	2.17e-02	2.59e-11	1.35e+04	8.84e+00
	Std	7.54e-17	2.09e-03	1.16e-11	2.77e+02	1.93e-01
f_{10}	Mean	2.87e-03	8.48e-01	2.83e-03	2.00e+01	1.43e+06
	Std	4.80e-03	1.50e-02	4.10e-03	1.41e+00	3.39e+05

4 在FCM聚类优化上的应用

FCM聚类采用模糊目标函数的值来评估聚类结果[10]. 当模糊目标函数值达到最小值时,聚类中心和隶属度最优,即模糊目标函数的值越低表示聚类效果越好. 模糊目标函数表达式如下:

$$J(\boldsymbol{U},\boldsymbol{V})=\sum_{i=1}^{C}\sum_{j=1}^{N}u_{ij}^{m}\|\boldsymbol{x}_j-\boldsymbol{v}_i\|^2. \tag{11}$$

其中: $\boldsymbol{U}$表示模糊隶属度矩阵, u_{ij}表示第i个数据点属于第j个聚类的隶属度, $\boldsymbol{v}_i$表示第i个聚类中心, "C"和"N"表示将N个点划分为C个簇, m表示加权指数, $\|\boldsymbol{x}_j-\boldsymbol{v}_i\|$表示第$i$个聚类中心与第$j$个数据点之间的欧几里德距离. 将ODGWO用于FCM聚

类优化,就是将式(11)作为目标函数.

FCM通过不断迭代求解模糊目标函数的最小值,每次迭代中使用下式计算隶属度和聚类中心:

$$u_{ij}=\frac{1}{\sum\limits_{k=1}^{C}\left(\dfrac{\|\boldsymbol{x}_j-\boldsymbol{v}_i\|}{\|x_j-v_k\|}\right)^{2/(m-1)}},\tag{12}$$

$$\boldsymbol{v}_i=\frac{\sum\limits_{j=1}^{N}u_{ij}^m\boldsymbol{x}_j}{\sum\limits_{j=1}^{N}u_{ij}^m}.\tag{13}$$

本文采用7组标准数据集进行聚类优化测试,选取GWO[1]、GWOepd[2]和LGWO[6]作为对比算法. 7组数据集直接来自于UCI机器学习数据库(http://archive.ics.uci.edu/ml). 4种算法的参数设置相同: 种群大小$N=20$,独立运行次数$\mathrm{Num}=30$,最大迭代次数$\mathrm{MaxDT}=100$. 表6给出了数据集的相关说明以及4种算法在不同数据集上的聚类优化结果. 其中: Mean表示一种算法独立运行Num次获取最小目标函数值的平均值, Std表示标准差, Time表示平均时间,首列数据集名称下面括号中的数据分别表示样本数、属性数和类数.

表6　FCM聚类优化的结果对比

数据集		ODGWO	GWO	GWOepd	LGWO
Blance (625, 4, 3)	Mean	1 666.667	1 666.667	1 666.667	1 666.667
	Std	2.151 3e-07	4.627 3e-05	9.073 9e-05	1.089 9e-04
	Time/s	2.256 1	2.352 5	2.356 4	2.386 9
Newthyroid (215, 5, 3)	Mean	17 757.884	17 759.080	17 759.108	17 761.470
	Std	2.545 2e-03	4.261 4e+00	4.588 6e+00	8.729 5e+00
	Time/s	1.057 4	1.035 8	1.084 2	1.102 0
Livedisorder (345, 6, 2)	Mean	333 107.699	333 107.701	333 107.702	333 107.702
	Std	9.298 2e-11	2.803 7e-03	4.627 5e-03	4.294 0e-03
	Time/s	1.082 3	1.091 8	1.097 5	1.144 0
Wine (178, 13, 3)	Mean	1 796 082.760	1 796 280.931	1 796 264.956	1 796 223.775
	Std	9.161 5e-10	4.978 9e+02	4.665 8e+02	5.104 8e+02
	Time/s	1.019 4	1.035 8	1.056 2	1.096 2
Lonosphere (351, 34, 2)	Mean	1 597.169	1 597.169	1 597.169	1 597.169
	Std	7.090 3e-13	9.853 0e-09	2.069 8e-08	1.143 8e-08
	Time/s	1.775 8	1.835 5	1.816 0	1.846 1
Iris (150, 4, 3)	Mean	60.576	60.576	60.576	60.582
	Std	5.198 0e-13	1.442 5e-05	1.139 3e-05	2.127 0e-02
	Time/s	0.814 65	0.788 55	0.803 01	0.847 28
Heart (270, 13, 2)	Mean	798.657	798.657	798.657	798.657
	Std	3.410 6e-13	6.476 0e-07	1.169 1e-06	1.058 6e-06
	Time/s	0.992 27	1.013 2	1.011 1	1.060 7

由表6可知, ODGWO在7个数据集上均获得了最优的Mean值和Std值,虽然在Newthyroid和Iris上ODGWO的运行时间比GWO略长,但ODGWO在其他5个数据集上的运行时间总是最少. 综合而言, ODGWO的聚类优化效果优于3种对比算法,且运行速度较快.

综合高维函数和聚类优化的结果, ODGWO的性能优于各组的对比算法,说明本文提出的ODGWO是有效的,在高维函数优化上取得了较好的结果,能够有效地处理FCM聚类优化问题.

5　结　论

针对GWO在解决高维和复杂优化问题时存在的全局搜索能力不足的问题,提出了一种新型反向学习和差分变异的GWO(ODGWO). 首先,引入最优最差反向学习策略增加种群的多样性,通过对最优灰狼使用一般反向学习机制,对最差灰狼使用随机反向学习机制,提升了算法的全局搜索能力;然后,引入动态随机差分变异策略,进一步增强了全局搜索能力;最后,在算法前半搜索阶段使用单维操作,后半搜索阶段使用全维操作,很好地平衡了算法的探索与开采能力和使两种操作优势互补,从而整体上提升了算法性能. 在30维、50维和1 000维Benchmark函数及7个数据集聚类上的优化结果表明,本文对GWO的改进是有效的,其性能大幅度优于GWO,与state-of-the-art算法相比,在整体上具有更好的优化性能,不仅适用于高维函数优化,而且能够有效处理FCM聚类优化问题,可望用于其他实际的优化问题. 另外,本文提出的最优最差反向学习与动态随机差分变异两种策略和

单维与全维分段操作方式可推广到其他优化算法的改进中.

参考文献(References)

[1] Mirjalili S, Mirjalili S M, Lewis A. Grey wolf optimizer[J]. Advances in Engineering Software, 2014, 69: 46-61.

[2] Saremi S, Mirjalili S Z, Mirjalili S M. Evolutionary population dynamics and grey wolf optimizer[J]. Neural Computing and Applications, 2015, 26(5): 1257-1263.

[3] Zhang S, Zhou Y. Grey wolf optimizer based on powell local optimization method for clustering analysis[J]. Discrete Dynamics in Nature & Society, 2015, 2015: 1-17.

[4] Jayabarathi T, Raghunathan T, Adarsh B R, et al. Economic dispatch using hybrid grey wolf optimizer[J]. Energy, 2016, 111: 630-641.

[5] 龙文, 蔡绍洪, 焦建军, 等. 求解高维优化问题的混合灰狼优化算法[J]. 控制与决策, 2016, 31(11): 1991-1997.
(Long W, Cai S H, Jiao J J, et al. Hybrid grey wolf optimization algorithm for high-dimensional optimization[J]. Control and Decision, 2016, 31(11): 1991-1997.)

[6] Heidari A A, Pahlavani P. An efficient modified grey wolf optimizer with lévy flight for optimization tasks[J]. Applied Soft Computing, 2017, 60: 115-134.

[7] Zhang X M, Kang Q, Cheng J F, et al. A novel hybrid algorithm based on biogeography-based optimization and grey wolf optimizer[J]. Applied Soft Computing, 2018, 67: 197-214.

[8] Emary E, Zawbaa H M, Grosan C. Experienced gray wolf optimization through reinforcement learning and neural networks[J]. IEEE Trans on Neural Networks and Learning Systems, 2018, 29(3): 681-694.

[9] Long W, Jiao J J, Liang X M, et al. An exploration-enhanced grey wolf optimizer to solve high-dimensional numerical optimization[J]. Engineering Applications of Artificial Intelligence, 2018, 68: 63-80.

[10] Nayak J, Naik B, Behera H S, et al. Hybrid chemical reaction based metaheuristic with fuzzy c-means algorithm for optimal cluster analysis[J]. Expert Systems with Applications, 2017, 79: 282-295.

[11] 张新明, 涂强, 康强, 等. 双模狩猎的灰狼优化算法在多阈值图像分割中应用[J]. 山西大学学报: 自然科学版, 2016, 39(3): 378-385.
(Zhang X M, Tu Q, Kang Q, et al. Grey wolf optimization algorithm with double-hunting modes and its application to multi-threshold image segmentation[J]. J of Shanxi University: Natural Science Edition, 2016, 39(3): 378-385.)

[12] Shi Y, Pun C M, Hu H, et al. An improved artificial bee colony and its application[J]. Knowledge-Based Systems, 2016, 107: 14-31.

[13] Ouyang H B, Gao L Q, Li S, et al. Improved global-best-guided particle swarm optimization with learning operation for global optimization problems[J]. Applied Soft Computing, 2017, 52: 987-1008.

[14] Wang H, Wu Z, Rahnamayan S, et al. Enhancing particle swarm optimization using generalized opposition-based learning[J]. Information Sciences, 2011, 181(20): 4699-4714.

[15] Wang H, Wu Z, Rahnamayan S, et al. Multi-strategy ensemble artificial bee colony algorithm[J]. Information Sciences, 2014, 279: 587-603.

[16] Meng A, Li Z, Yin H, et al. Accelerating particle swarm optimization using crisscross search[J]. Information Sciences, 2016, 329(SI): 52-72.

[17] Draa A, Bouzoubia S, Boukhalfa I. A sinusoidal differential evolution algorithm for numerical optimization[J]. Applied Soft Computing, 2015, 27: 99-126.

[18] 夏学文, 刘经南, 高柯夫, 等. 具备反向学习和局部学习能力的粒子群算法[J]. 计算机学报, 2015, 38(7): 1397-1407.
(Xia X W, Liu J N, Gao K F, et al. Particle swarm optimization algorithm with reverse-learning and local-learning behavior[J]. Chinese J of Computers, 2015, 38(7): 1397-1407.)

[19] 周晓根, 张贵军, 郝小虎, 等. 一种基于局部Lipschitz下界估计支撑面的差分进化算法[J]. 计算机学报, 2016, 39(12): 2631-2650.
(Zhou X G, Zhang G J, Hao X H, et al. Differential evolution algorithm based on local Lipschitz underestimate supporting hyperplanes[J]. Chinese J of Computers, 2016, 39(12): 2631-2650.)

[20] 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 & Evolutionary Computation, 2011, 1(1): 3-18.

作者简介

张新明(1963−), 男, 教授, 从事智能优化算法、数字图像处理、模式识别等研究, E-mail: xinmingzhang@126.com;

王霞(1993−), 女, 硕士生, 从事智能优化算法、数字图像处理的研究, E-mail: wangxiagzyx@126.com;

康强(1989−), 男, 硕士生, 从事智能优化算法、数字图像处理的研究, E-mail: 2813985282@qq.com.

(责任编辑：孙艺红)