灰狼优化与差分进化的混合算法及函数优化

张新明[1,2] 涂 强[1] 康 强[1] 程金凤[1]
(河南师范大学计算机与信息工程学院 新乡 453007)[1]
(河南省高校计算智能与数据挖掘工程技术研究中心 新乡 453007)[2]

摘 要 灰狼优化(Grey Wolf Optimization,GWO)算法是近年被提出的一种新型智能优化算法,具有收敛速度快和优化精度高的特点,但对于一些复杂优化问题易陷入局部最优。差分进化(Differential Evolution,DE)算法的全局搜索能力强,但其性能对参数敏感,且局部搜索能力不足。为了发挥二者各自的优点并弥补存在的缺陷,提出了一种灰狼优化与差分进化的混合优化算法。首先使用嵌入趋优算子的GWO算法搜索,以便在更短的过程中获得更高的优化精度和更快的收敛速度;然后采用自适应调节参数的差分进化策略来进一步提高算法对复杂优化函数的寻优性能,从而获得一种高性能的混合优化算法,以便能更高效地解决各种函数优化问题。对12个高维函数的优化结果表明,与标准GWO,ACS,DMPSO及SinDE相比,新的混合优化算法不仅具有更好的收敛速度和优化性能,而且具有更好的普适性,更适用于解决各种函数优化问题。

关键词 优化算法,混合优化算法,灰狼优化算法,差分进化,函数优化
中图法分类号 TP18 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.09.019

Hybrid Optimization Algorithm Based on Grey Wolf Optimization and Differential Evolution for Function Optimization

ZHANG Xin-ming[1,2] TU Qiang[1] KANG Qiang[1] CHENG Jin-feng[1]
(College of Computer and Information Engineering, Henan Normal University, Xinxiang 453007, China)[1]
(Engineering Technology Research Center for Computing Intelligence & Data Mining of Henan Province, Xinxiang 453007, China)[2]

Abstract Grey wolf optimizer (GWO) is a novel intelligent optimization algorithm which has proposed resently and it has such merits as fast convergence speed, high optimization precision, but easily entraps in local optima. The differential evolution (DE) algorithm has strong global search ability, but its local search ability is poor and its performance is sensitive to the parameters. To take advantage of the merits of GWO and DE and overcome their defects in dealing with function optimization problems, a hybrid optimization algorithm based on grey wolf optimization and differential evolution (GWODE) was proposed. First, the optima-inclinded operator embedded GWO is utilized which is benefit to improving the optimization precision and convergence rate of the algorithm in a shorter search process. Then, an adaptive differential strategy, which can automatically adjust the value of the parameters, is employed to further improve the optimization performance of the algorithm for complex optimization functions. Thus, a hybrid algorithm with high performance is obtained and it's more efficient to solve various function optimization problems. The optimization results on 12 benchmark functions show that the new hybrid optimization algorithm has higher search precision, better optimal performance and stronger applicability, and it's more suitable for solving a variety of optimization problems, compared with the standard GWO, ACS, DMPSO and SinDE.

Keywords Optimization algorithm, Hybrid optimization algorithm, Grey wolf optimization algorithm, Differential evolution, Function optimization

---

到稿日期:2016-07-02 返修日期:2016-11-13 本文受河南省重点科技攻关项目(132102110209),河南省基础与前沿技术研究计划项目(142300410295)资助。

张新明(1963—),男,教授,CCF会员,主要研究方向为智能优化算法、数字图像处理和模式识别等,E-mail:xinmingzhang@126.com;**涂 强**(1995—),男,硕士生,主要研究方向为数字图像处理和智能优化算法;**康 强**(1989—),男,硕士生,主要研究方向为数字图像处理和智能优化算法;**程金凤**(1990—),女,硕士生,主要研究方向为数字图像处理。

1 引言

科学计算、工程设计等领域中的许多问题都可归结为函数优化问题。这些优化问题中的大多数往往具有复杂性、非线性、多极值、建模困难等特点，依靠单一的智能优化算法往往不能取得令人满意的结果，人们逐渐从利用单一的智能优化算法过渡到利用混合的智能优化算法来解决各种复杂的函数优化问题。混合智能优化算法不是简单地将几种算法组合在一起，而是按照某种机制或策略进行混合，利用算法间的优势进行互补来克服单一算法的不足，进而增强算法的求解能力。目前，国内外学者针对特定问题提出了很多个性化的混合智能优化算法，如粒子群优化算法与人工鱼群算法的混合[1]、粒子群优化算法与蚁群算法的混合[2]。文献[3]将模拟退火算法、变邻域搜索算法和蚁群算法混合，用于解决平行机调度问题；文献[4]将细菌觅食优化算法中的趋化算子嵌入到粒子群优化算法中，用于解决多阈值图像分割问题。

群智能混合优化算法及应用虽取得了很多的研究成果，但仍存在尚未解决的问题，如算法的混合机制或混合策略有待进一步的深入研究；群智能混合优化算法的内部机理及用于解决实际复杂问题的应用也需研究[5]。考虑到综合利用不同智能优化算法的互补性与差异性，采用分而治之的策略，扬长避短，从而实现算法间的优化互补，本文提出了一种灰狼优化算法和差分进化算法的混合算法。

灰狼优化算法是 Mirjalili[6]于 2014 年提出的一种新型群体智能优化算法。该算法因具有结构简单、收敛迅速、可调参数少等优点，被广泛应用于多层传感器训练[7]、流动车间调度[8]、经济调度指派[9]、求解约束优化问题[10]等诸多方面，已经成为生物启发式计算领域的又一研究热点。但其在解决高维度、多模态复杂函数优化问题时，易陷入局部最优和出现早熟收敛的现象。差分进化算法作为一类具有代表性的进化算法[11]，因具有全局搜索能力和鲁棒性强等特点，在求解非连续、多峰值等复杂优化问题时效果很好，但存在局部搜索能力较弱、搜索效率较低和搜索性能对参数具有一定依赖性的缺陷。针对 GWO 和 DE 各自在应用上的优势和不足，本文提出了一种灰狼优化和差分进化的混合优化算法(Hybrid Optimization Algorithm Based on Grey Wolf Optimization and Differential Evolution，GWODE)。首先采用嵌入趋优算子的 GWO 算法来更新种群位置，以便算法在较短的搜索期间获得较好的优化精度和较快的收敛速度，更好地发挥 GWO 算法的优势；然后通过采用自适应调节参数的差分进化策略来进一步提高算法对复杂优化函数的寻优性能，实现算法之间的优势互补，进而使混合算法的整体性能得到改进和提高，增强普适性。

2 灰狼优化算法

灰狼优化算法是一种新的群体智能优化算法，它模拟了灰狼家族的社会等级制度和群体狩猎行为[6]。

(1)社会等级制度。自然界中的灰狼种群按社会地位从高到低依次被划分为 α,β,δ 和 ω 4 个等级[12]。在设计 GWO 时，为构建灰狼的等级制度模型，定义狼群中的当前最优解为 α 狼，次优解为 β 狼，第三优解为 δ 狼，剩下的解为 ω 狼。在 GWO 算法中，狩猎(优化)是由 α 狼、β 狼和 δ 狼负责引导 ω 狼进行猎物的跟踪围捕，最终完成狩猎任务。

(2)狩猎行为。狼群的主要狩猎过程为：跟踪、靠近猎物；追赶、骚扰猎物；包围、攻击猎物[13]。包围行为的数学模型如下[6]：

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

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

其中，$\boldsymbol{D}$ 表示灰狼与猎物间的距离，t 表示当前迭代次数，$\boldsymbol{X}_p$ 表示猎物的位置向量，$\boldsymbol{X}$ 表示灰狼的位置向量，$\boldsymbol{A}$ 和 $\boldsymbol{C}$ 是参数向量，且 $\boldsymbol{A}=2\boldsymbol{a}\cdot\boldsymbol{r}_1-\boldsymbol{a}$，$\boldsymbol{C}=2\cdot\boldsymbol{r}_2$，其中 $\boldsymbol{a}$ 在迭代过程中从 2 线性递减到 0，$\boldsymbol{r}_1$ 和 $\boldsymbol{r}_2$ 是[0,1]中的随机向量。

为模拟灰狼的狩猎行为，假设 α 狼、β 狼和 δ 狼对猎物的位置有更好的了解，因此灰狼群体可以利用这三者的位置来判断猎物所在的方位。灰狼群体根据 α 狼、β 狼和 δ 狼的位置信息来更新自身位置的公式如下[6]：

$$\boldsymbol{D}_\alpha=|\boldsymbol{C}_1\cdot\boldsymbol{X}_\alpha(t)-\boldsymbol{X}(t)| \tag{3}$$

$$\boldsymbol{D}_\beta=|\boldsymbol{C}_2\cdot\boldsymbol{X}_\beta(t)-\boldsymbol{X}(t)| \tag{4}$$

$$\boldsymbol{D}_\delta=|\boldsymbol{C}_3\cdot\boldsymbol{X}_\delta(t)-\boldsymbol{X}(t)| \tag{5}$$

$$\boldsymbol{X}_1=\boldsymbol{X}_\alpha(t)-\boldsymbol{A}_1\cdot\boldsymbol{D}_\alpha \tag{6}$$

$$\boldsymbol{X}_2=\boldsymbol{X}_\beta(t)-\boldsymbol{A}_2\cdot\boldsymbol{D}_\beta \tag{7}$$

$$\boldsymbol{X}_3=\boldsymbol{X}_\delta(t)-\boldsymbol{A}_3\cdot\boldsymbol{D}_\delta \tag{8}$$

$$\boldsymbol{X}(t+1)=(\boldsymbol{X}_1+\boldsymbol{X}_2+\boldsymbol{X}_3)/3 \tag{9}$$

为了说明上述模拟公式，图 1 示出了候选解根据 α 狼、β 狼和 δ 狼的位置信息来更新自身位置的原理图。由图 1 可知，候选解分布在由 α 狼、β 狼和 δ 狼定义的一个随机圆内，换句话说，GWO 算法的寻优过程就是先由 α 狼、β 狼和 δ 狼对猎物的位置进行评估定位，然后群内的其余个体以此为参考并在猎物周围随机更新位置。

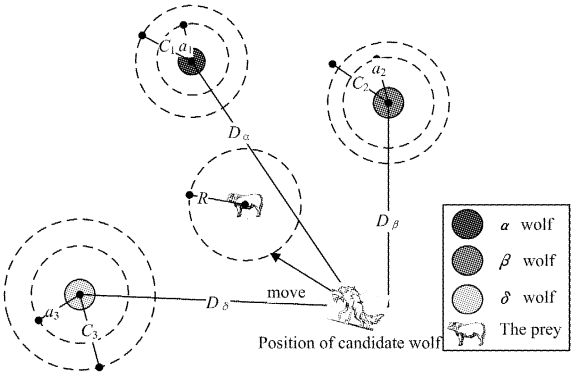


图 1 灰狼位置更新机制原理

3 差分进化算法

差分进化算法是由 Storm 和 Price 通过模拟生物进化机制提出的一种启发式全局搜索算法[11]，因具有结构简单、可调参数少、功能强大且鲁棒性强等优点而得到广泛的应用。DE 算法主要通过变异、交叉、选择这 3 个操作来求解最优解。

3.1 变异操作

在 DE 算法中，有多种变异操作；实现变异操作的基本原理是将一个差分向量加入到一个基向量中，即在种群中随机选取两个不同的个体，将其向量差进行缩放后再与待变异的

个体进行合成。DE/rand/1 的变异操作如式(10)所示：

$$\boldsymbol{D}_i(t+1)=\boldsymbol{X}_{r1}(t)+Fr\times(\boldsymbol{X}_{r2}(t)-\boldsymbol{X}_{r3}(t)) \tag{10}$$

其中，t 表示当前迭代次数；Fr 表示缩放因子，其取值范围通常被限定在[0,2]内；r_1，r_2 和 r_3 是区间[1,N]内与 i 不等的随机整数，且满足两两互不相等；N 为种群大小。

3.2　交叉操作

交叉操作的基本原理是将待变异个体与经变异操作后产生的新个体相互交换一些元素。对于第 i 个个体的第 j 维，其实现交叉操作的方法如式(11)所示：

$$\boldsymbol{U}_{i,j}(t+1)=\begin{cases}\boldsymbol{D}_{i,j}(t+1), & \text{if } rand<Cr \text{ or } j=sn\\ \boldsymbol{X}_{i,j}(t), & \text{otherwise}\end{cases} \tag{11}$$

其中，Cr 表示交叉概率，$rand$ 表示[0,1]区间内的随机数，sn 表示一个随机的维度。

3.3　选择操作

DE 算法采用贪婪算法来选择进入下一代的个体以确保种群的进化方向，即当新产生的子代个体优于父代个体时，保留子代个体至下一代，否则保留父代个体至下一代。其选择公式如式(12)所示：

$$\boldsymbol{X}_i(t+1)=\begin{cases}\boldsymbol{U}_i(t+1), & \text{if } f(\boldsymbol{U}_i(t+1))\leqslant f(\boldsymbol{X}_i(t))\\ \boldsymbol{X}_i(t), & \text{otherwise}\end{cases} \tag{12}$$

4　灰狼优化与差分进化混合优化算法

由 GWO 和 DE 算法可知：灰狼群体根据 α 狼、β 狼和 δ 狼的位置信息来更新自身位置，迅速向最优解聚集，使得优化精度高、收敛速度快，但 3 种位置信息对于其他个体的位置更新又起着绝对的引导作用，容易致使整个狼群过早聚集于群体当前最优位置的某一邻域内，种群的多样性得不到有效保持，容易陷入局部最优；而 DE 算法采用 DE/rand/1 的变异操作有利于保持种群的多样性，对于非连续、非可微、带噪声及多模态复杂优化函数有着较好的优化效果，但搜索效率偏低。鉴于此，文中提出灰狼优化与差分进化的混合优化算法。

4.1　混合策略

灰狼优化与差分进化混合算法采用一种简单的混合策略：将算法的整个搜索过程分为两个阶段，第一阶段即在算法前 1/3 的搜索过程中采用嵌入趋优算子的 GWO 搜索策略来寻找最优解；如果不能获得最优解，就转入第二阶段，即在后 2/3 的搜索过程中利用自调节参数的差分进化策略来获得全局最优解。同时，为确保种群的进化方向，两个阶段都将采用贪婪选择算子来选择是否更新新解。

4.2　嵌入趋优算子的灰狼优化算法

由于标准 GWO 模型中 α 狼、β 狼和 δ 狼代表群体最优的 3 个解，基于对这 3 个最优解的充分挖掘和利用，提出了一种趋优算子来更新种群个体的位置，其计算公式如下：

$$x_{i,j}(t+1)=(x_{a,num}(t)+x_{\beta,num}(t)+x_{\delta,num}(t))/3 \tag{13}$$

其中，num 表示随机的一个维度。

从式(13)中可以看出趋优算子充分考虑了群体当前 3 个历史最优解的位置信息，从解的角度向群体中的当前 3 个最优解趋近，故称为趋优算子；从维的角度发生突变，通过直接趋向最优解中随机选取的某一维度的位置来更新搜索因子的位置，以增强种群的多样性；此算子主要体现在局部搜索能力上，全局搜索能力有限，因此整体上更侧重于提高算法的优化精度和收敛速度。

嵌入趋优算子的 GWO 算法的伪代码如算法 1 所示。

算法 1　嵌入趋优算子的 GWO

```
输入：待更新狼群的位置信息 G=(X1,X2,…,XN)，趋优概率 Mr
输出：更新后狼群的位置信息 U=(X1,X2,…,XN)
for i=1 to N do
  if rand(0,1)<Mr then
    for j=1 to D do
    根据式(9)更新搜索算子的位置
    end for
  else
    for j=1 to D do
      根据式(13)更新搜索算子的位置
    end for
  end if
end for
```

其中，N 表示灰狼个数，D 表示搜索空间的维度，$rand$(0,1)表示均匀分布在[0,1]之间的随机实数。

4.3　灰狼优化与差分进化的混合优化算法

本文提出的差分进化算子的伪代码如算法 2 所示，其中 Fr 和 Cr 采用文献[14]中自适应调整的方式，见式(14)和式(15)：

$$Fr_i(t)=1/2\times((\sin(2\pi\times freq\times t))\times t/MaxDT+0.85) \tag{14}$$

$$Cr_i(t)=1/2\times((\sin(2\pi\times freq\times t+\pi))\times t/MaxDT)+1) \tag{15}$$

其中，$freq$ 为调节参数，$MaxDT$ 表示最大迭代次数。

算法 2　差分进化算子

```
输入：待更新狼群的位置信息 U=(X1,X2,…,XN)，调节参数 freq
输出：更新后狼群的位置信息 H=(X1,X2,…,XN)
for i=1 to N do
    随机选择 3 个个体 r1,r2,r3，并满足 r1≠r2≠r3≠i
    在区间[1,D]中随机选择一个整数 sn
  for j=1 to D do
      if rand (0,1)<Cri(t) or j==sn then
            Ut,j(t+1)=Xr1,j(t)+Frj(t)(Xr2,j(t)−Xr3,j(t))
        else
            Ut,j(t+1)=Xi,j(t)
    end if
  end for
end for
```

其中，N 表示灰狼个数，D 表示搜索空间的维度，$rand$(0,1)表示均匀分布在[0,1]之间的随机实数。

贪婪选择算子的伪代码如算法 3 所示。

算法 3　贪婪选择算子

```
输入：待更新狼群的位置信息 H=(X1,X2,…,XN)
输出：更新后狼群的位置信息 M=(X1,X2,…,XN)
for i=1 to N do
```

```
    if f(U_i(t))≤f(X_i(t)) then
      X_i(t+1)=U_i(t)
    else
      X_i(t+1)=X_i(t)
    end if
end for
```

GWODE 算法的流程如下：

Step 1 参数初始化，包括灰狼种群规模 N、最大迭代次数 $MaxDT$ 等基本参数。

Step 2 根据变量的上、下界初始化灰狼个体的位置 X_i。

Step 3 计算狼群中每一头狼的适应度值。

Step 4 找出当前具有最优、次优和第三优适应值的 3 类狼，并将其位置依次保存为 $\boldsymbol{X}_\alpha$，$\boldsymbol{X}_\beta$，$\boldsymbol{X}_\delta$。

Step 5 对狼群位置进行更新，即在前 1/3 搜索过程中按照算法 1 来更新搜索因子的位置，在后 2/3 的搜索过程中按照算法 2 来更新搜索因子的位置。

Step 6 依据算法 3，即按式(12)确定是否接受该新位置。

Step 7 更新参数 $\boldsymbol{a}$，$\boldsymbol{A}$ 和 $\boldsymbol{C}$。

Step 8 判断是否到达最大迭代次数 $MaxDT$，如果是则停止并返回 $\boldsymbol{X}_\alpha$ 的值作为最终得到的最优解，否则转 Step 3。

从 GWODE 算法的流程可以看出：GWO 由于有着优化精度高和收敛速度快的优点，只需较短的搜索过程便可获得精度较高的次优解，而趋优算子能使算法在更短的搜索过程(本文给定 1/3 的总搜索过程)中获得较好的优化精度和较快的收敛速度，为下一阶段的差分进化算法搜索提供了更多的搜索时间和计算资源以及良好的优化基础；同时自适应调节参数(见式(14)和式(15))的差分进化策略克服了其优化性能对参数过分依赖的问题，使得对于复杂函数优化问题有着更强的寻优能力，即将更多的搜索过程(2/3)分配至差分进化算法有利于其有充足的时间进行搜索，提高了算法对复杂优化问题的求解能力，实现了两种算法之间的优势互补。

5 实验结果与分析

5.1 测试函数和对比算法

为了验证 GWODE 算法的性能，将 GWODE 算法用于经典的高维函数优化计算中，并将计算结果与标准 GWO[6]，DMPSO[15]，SinDE[14] 以及 ACS[16] 算法的计算结果进行比较。其中，DMPSO 算法是由李景洋等人于 2014 年提出的一种较新的 PSO 的改进算法，由文献[15]可知，这种采用双模飞行模式的粒子群算法克服了标准 PSO 易陷入局部最优的缺点，具备比标准 PSO 更快的收敛速度和更高的搜索效率；SinDE 算法是由 Amer Draa 等人于 2015 年提出的一种有效的 DE 改进算法，其通过自适应的方式来动态调整缩放因子 Fr 和交叉概率 Cr，使得算法在探索过程和开发过程间达到平衡，其性能明显优于传统 DE；ACS 是由 Manoj Naik 于 2015 年提出的一种新颖的自适应调整步长的改进布谷鸟算法(Cuckoo search algorithm，CS)，其优化性能要明显优于标准 CS。同时，为考虑本文所提算法的普适性，特选取 12 个高维优化函数用于本次算法的性能测试分析，它们的表达式和全局最优解等情况如表 1 所列。其中 f_1-f_4 为单模态优化函数，由一些常见的单峰基准函数组成，主要用来考查算法的寻优精度和局部搜索能力。f_5-f_{12} 是复杂优化函数，包括非线性多峰函数、平移函数等，它们具有广泛的搜索空间、大量的局部极值点，再加上高维特性和强震荡性，可以用来检验算法避免早熟收敛和规避局部最优的能力以及对未知空间的探索能力[17-18]。本文所有实验均在 Intel Core i7-4790 主频为 3.60GHz 的 CPU 和内存为 8GB 的 PC 机上进行，操作系统为 Microsoft Windows 7，所有的算法都采用 MATLAB 2014A 编程实现。

表 1 测试函数

Name	Function	Dim	Range	F_{min}
Schwefel 2.21	$f_1(x)=\max_{i=1}^{n}\{\lvert x_i\rvert\}$	50	$[-100,100]$	0
Schwefel 2.22	$f_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	50	$[-10,10]$	0
Quadric	$f_3(x)=\sum_{i=1}^{n}ix_i^4+rand(0,1)$	50	$[-1.28,1.28]$	0
Rosenbrock	$f_4(x)=\sum_{i=1}^{n}[(x_i-1)^2+100(x_{i+1}+x_i^2)^2]$	50	$[-10,10]$	0
Ackley	$f_5(x)=20+e-20\exp[-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}]-\exp[\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)]$	50	$[-32,32]$	0
Rastrigin	$f_6(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	50	$[-5.12,5.12]$	0
Griewank	$f_7(x)=1+\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})$	50	$[-600,600]$	0
Levy	$f_8(y)=\sin^2(\pi y_1)+\sum_{i=1}^{n-1}[(y_i-1)^2(1+10\sin^2(\pi y_i+1))]+(y_n-1)^2(1+10\sin^2(2\pi y_n))$ $y_i=1+(x_i-1)/4$	50	$[-10,10]$	0
Shifted Sphere	$f_9(z)=\sum_{i=1}^{n}z_i^2-450, z=x-o$	50	$[-100,100]$	−450
Shifted Schwefel's Problem 2.21	$f_{10}(z)=\max_{i=1}^{n}\{\lvert z_i\rvert\}-450, z=x-o$	50	$[-100,100]$	−450
Shifted Griewank	$f_{11}(z)=1+\sum_{i=1}^{n}\frac{z_i^2}{4000}-\prod_{i=1}^{n}\cos(\frac{z_i}{\sqrt{i}})-180, z=x-o$	50	$[-600,600]$	−180
Shifted Ackley	$f_{12}(z)=20+e-20\exp[-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n}z_i^2}]-\exp[\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi z_i)]-140, z=x-o$	50	$[-32,32]$	−140

5.2 仿真实验结果

为确保测试的公平性，本实验设置 GWODE 的种群规模和最大迭代次数与 GWO，DMPSO，SinDE 和 ACS 算法的一致，即每种算法种群规模统一设置为 40，最大迭代次数都为 1500。5 种算法的其他参数设置如下：对于 DMPSO，SinDE，ACS 和 GWO 的参数设置请参见文献[6]和文献[15-16]；而本文提出的 GWODE 算法中差分进化算子中的 $freq=0.25$，其他设置与标准 GWO 相同。

5.2.1 算法参数分析

GWODE 算法中的参数对算法性能有较大影响，其中嵌

入趋优算子的灰狼优化算法中的趋优概率 Mr 对算法性能的影响最大，因此这里仅讨论 Mr 对 GWODE 算法的影响。将参数 Mr 递增地取 5 个不同的值，分别为 $Mr=0.5$，$Mr=0.6$，$Mr=0.7$，$Mr=0.8$ 和 $Mr=0.9$，在其他参数不变的情况下，选取单峰函数 f_1、多峰函数 f_6 和平移函数 f_{10} 作为测试函数，独立运行 GWODE 算法 30 次得到 3 个函数的最优平均值 Mean 和方差 Std(见表 2)，以此来考查参数 Mr 对算法优化性能的影响。

由表 2 可知，对于单峰函数，趋优概率越小表示采用趋优模式的概率越大，算法的寻优精度随着 Mr 的增大而不断降低，当 $Mr=0.5$ 时所取得的函数均值和方差均是最好的，此时趋优模式和原始 GWO 算法的狩猎模式共同作用使得算法具有较高的搜索精度；对于多峰函数和平移函数，算法均在 $Mr=0.7$ 时取得最优的函数均值和均方差。综合考虑对所有函数的普适性，折中选择趋优概率为 0.7 时较为合适，故以下实验中 Mr 取为 0.7。

表 2　参数 Mr 取不同值时 GWODE 算法的计算结果

fun	$Mr=0.5$		$Mr=0.6$		$Mr=0.7$		$Mr=0.8$		$Mr=0.9$	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
f_1	**4.4076e−25**	**2.4133e−24**	6.1889e−24	3.3889e−23	9.5399e−23	4.6045e−22	1.0101e−19	4.9508e−19	4.264e−16	9.0485e−16
f_6	23.4726	61.0079	32.379	74.46	**0**	**0**	**0**	**0**	**0**	**0**
f_{10}	−441.5153	5.27	−441.1632	4.919	**−444.0763**	**0.5373**	−442.5071	4.7608	−441.3131	4.8069

5.2.2　优化性能分析

为了便于比较，选取 5 个算法在指定最大迭代次数下对 12 个优化函数独立进行 30 次寻优计算的最优值(Best)、最差值(Worst)、平均值(Mean)、均方差(Std)及平均运行时间(Time/s)作为各算法寻优性能的评价指标。

表 3、表 4 列出各算法在优化函数上的最优值、最差值、平均值及均方差结果，表中最优值用黑体表示。

表 3　5 种算法在 4 个单模态优化函数上的实验结果对比

Fun	Algorithm	Best	Worst	Mean	Std
f_1	DMPSO	3.5894	35.0254	16.4167	6.7334
	SinDE	4.6238	22.3249	9.7500	3.6911
	ACS	3.9817	10.1489	6.3133	1.7465
	GWO	2.9397e−17	2.9321e−15	4.3869e−16	6.2488e−16
	GWODE	**2.3573e−34**	**2.6279e−18**	**9.5399e−23**	**4.6045e−22**
f_2	DMPSO	1.1796e−38	1.4959e−36	3.8454e−37	6.3431e−37
	SinDE	7.0641e−10	3.8208e−08	1.1252e−08	1.5827e−08
	ACS	2.2214	1.0000e+10	6.0000e+09	5.4772e+09
	GWO	1.7252e−44	1.4357e−43	8.2746e−44	5.4408e−44
	GWODE	**8.3000e−57**	**5.8494e−56**	**3.4598e−56**	**2.0408e−56**
f_3	DMPSO	0.0015	0.0532	0.0139	0.0118
	SinDE	0.0092	0.0319	0.0212	0.0060
	ACS	0.0683	0.1946	0.1125	0.0336
	GWO	2.3941e−04	0.0016	6.2352e−04	3.1583e−04
	GWODE	**4.2701e−05**	**0.0014**	**4.9263e−04**	**3.3974e−05**
f_4	DMPSO	43.9526	313.1685	55.5819	87.4313
	SinDE	46.5424	105.3206	48.5231	10.7372
	ACS	44.7071	105.9939	49.3372	10.7848
	GWO	44.0887	48.6067	47.0051	0.7621
	GWODE	**43.6988**	**46.2069**	**45.7717**	**0.3412**

表 3 列出了 5 种算法对 4 个单模态优化函数的寻优结果。由表 3 可知，对于这 4 个优化函数而言，GWODE 在最优值、最差值、平均值和均方差上均获得了最优秀解，表明 GWODE 对单峰优化函数有着更高的优化精度。同时由表 3 还可以看出，单一的 GWO 对 f_1-f_4 的寻优结果明显优于单一的 SinDE，体现了 GWO 在面对单峰优化函数时所具备的优化精度高的优点，而 GWODE 对这 4 个单峰优化函数的寻优结果又明显优于单一的 GWO，这正是由于趋优算子的引入促使了搜索因子通过对 3 个最优解位置信息的充分挖掘和利用，在短时间内有效地提高了算法的优化精度和收敛速度，弥补了差分进化策略局部搜索能力差的不足，也为 DE 提供了更多的计算资源来解决复杂优化问题。

表 4　5 种算法在 8 个复杂优化函数上的实验结果对比

Fun	Algorithm	Best	Worst	Mean	Std
f_5	DMPSO	8.8818e−15	4.8850e−14	1.4270e−14	1.3486e−14
	SinDE	1.8683e−08	6.3784e−06	4.4386e−07	1.3000e−06
	ACS	0.0194	2.1473	1.1816	0.6405
	GWO	1.3323e−14	2.0428e−14	1.6165e−14	3.4153e−15
	GWODE	**2.6645e−15**	**6.2172e−15**	**5.9804e−15**	**9.0135e−16**
f_6	DMPSO	**0**	49.7480	1.6583	9.0827
	SinDE	37.2530	82.5053	59.2999	10.4207
	ACS	333.8798	571.5183	466.9238	63.2617
	GWO	**0**	6.1819	0.2061	1.1287
	GWODE	**0**	**0**	**0**	**0**
f_7	DMPSO	**0**	2.1832e−09	7.2775e−11	3.9860e−10
	SinDE	3.4528e−14	5.3752e−10	3.6271e−11	1.2258e−10
	ACS	8.6113e−04	0.0705	0.0118	0.0130
	GWO	**0**	0.0159	5.2938e−04	0.0029
	GWODE	**0**	**0**	**0**	**0**
f_8	DMPSO	1.4998e−12	6.0203	1.7155	2.4830
	SinDE	6.0373e−14	1.1005e−09	6.1080e−11	2.1224e−10
	ACS	4.5934e−04	6.9891	0.8026	1.5783
	GWO	2.7283	8.0858	5.1607	1.5356
	GWODE	**8.2709e−16**	**6.6441e−15**	**2.2971e−15**	**1.2524e−15**
f_9	DMPSO	1.3109e+04	2.7166e+04	2.0123e+04	3.1056e+03
	SinDE	**−450**	**−450**	**−450**	4.6507e−10
	ACS	−449.9995	−449.9971	−449.9988	5.7990e−04
	GWO	1.0750e+3	1.7425e+04	5.1622e+03	3.3761e+03
	GWODE	**−450**	**−450**	**−450**	**3.1518e−11**
f_{10}	DMPSO	−415.6828	−407.8902	−412.6898	2.1598
	SinDE	−442.8037	−408.4436	−433.6905	6.5830
	ACS	−439.2979	−413.0132	−423.4827	6.4619
	GWO	−432.3399	−402.9861	−420.4177	4.9065
	GWODE	**−445.0394**	**−441.3629**	**−444.0763**	**0.5373**
f_{11}	DMPSO	−62.7543	−0.5956	−29.7880	16.7131
	SinDE	**−180**	−179.9926	−179.9998	0.0013503
	ACS	−179.9908	−179.8821	−179.9551	0.0316
	GWO	−174.4657	3.0343	−118.5781	40.7408
	GWODE	**−180**	**−180**	**−180**	**2.4862e−10**
f_{12}	DMPSO	−126.3366	−124.507	−125.3027	0.4166
	SinDE	**−140**	**−140**	**−140**	1.4798e−06
	ACS	−136.9241	−119.6098	−120.9831	3.9642
	GWO	−132.3540	−124.2466	−128.5874	1.7735
	GWODE	**−140**	**−140**	**−140**	**3.2749e−11**

由表 4 可以看出，在 f_5-f_{12} 这 8 个复杂优化函数中，相对于其他 4 种算法，GWODE 在最优值、最差值、平均值和均方差上都获得了最好的寻优结果，尤其是在 f_6，f_7，f_9，f_{11} 和 f_{12} 函数上，GWODE 甚至均获得了全局最优解，表明 GWODE 对复杂优化函数也有着较好的寻优性能；而与 DE

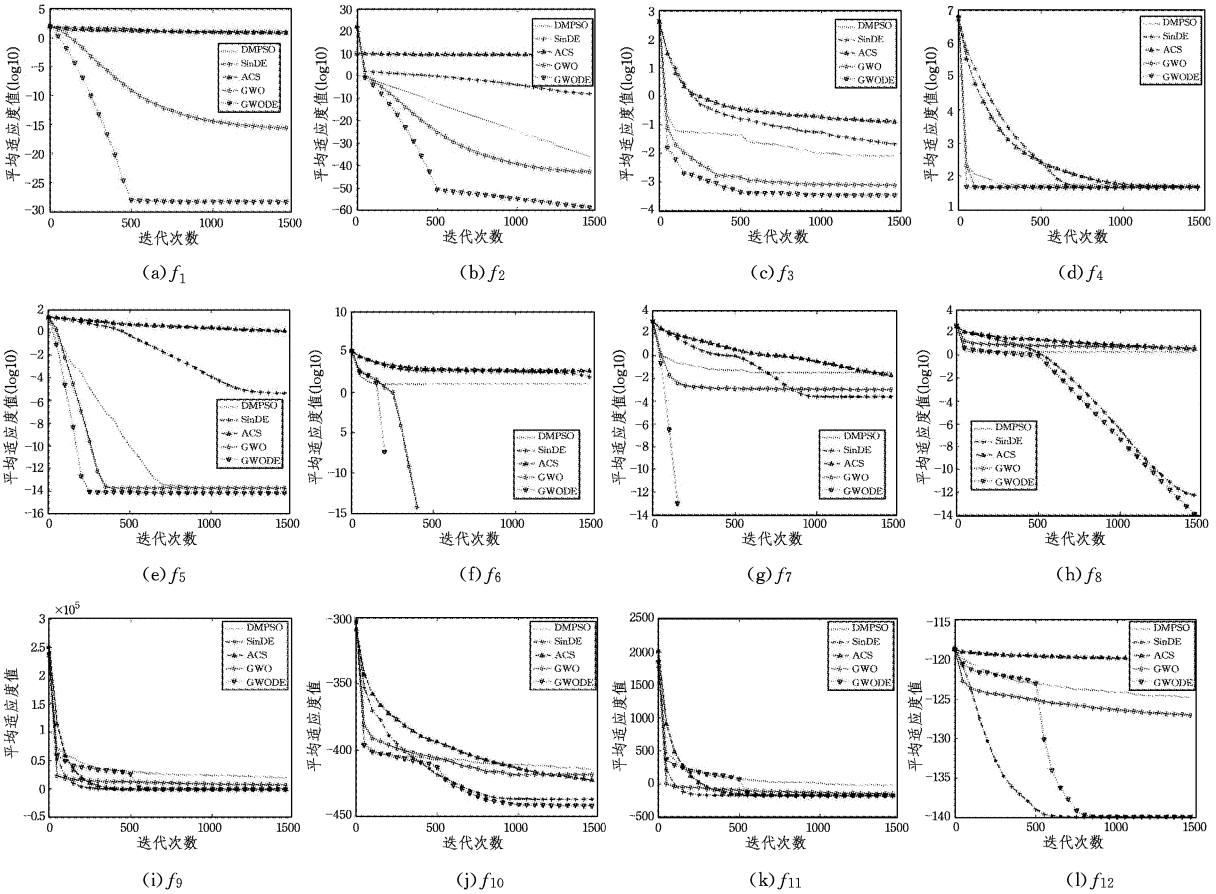
相比，GWO 在一些复杂优化函数如 f_8-f_{12} 上的优化性能较差。由此可以看出，GWODE 对复杂函数有着较强的规避局部最优的能力。主要原因为：1)嵌入趋优算子的 GWO 奠定了良好的优化基础；2)DE 本身对非连续、多模态等复杂优化函数有着较好的优化性能，而采用的自调节的缩放因子 Fr 和交叉概率 Cr 消除了算法对参数的依赖性；3)二者的有效融合促进了全局搜索与局部搜索达到平衡，提高了算法的搜索精度和跳出局部最优解的能力，使 GWODE 的整体性能优于 GWO 和 DE。

5.2.3 运行时间分析

表 5 列出了各算法在优化函数上的运行时间对比结果。从表 5 可以看出，对于函数 f_1-f_{12}，ACS 耗时最少，DMPSO 耗时最长，GWODE 的耗时虽然多于 ACS，但 ACS 的寻优精度却明显低于 GWODE。图 2 为 12 个函数的收敛曲线图，从图也可以看出 ACS 极易出现早熟收敛的现象，因此在优化精度要求较高的同等情况下，GWODE 能更快地获得全局最优解。虽然 GWODE 在面对 f_5-f_{12} 这 8 个复杂优化函数时的运行时间多于 SinDE，但对于 f_1-f_4 这 4 个单模态优化函数的运行时间却少于 SinDE，且无论是面对单峰优化函数，还是面对复杂多峰优化函数，GWODE 的运行时间均少于 GWO。而根据图 2 的收敛图也会发现，在前 500 次的迭代过程中，GWODE 在 f_1-f_4 上相对于其他 4 种算法有着最快的收敛速度，这在一定程度上也要归功于 GWODE 中所采用的灰狼优化策略具有优化精度高和收敛速度快的优点，且趋优算子的引入也进一步提高了算法的寻优性能，以便为差分进化策略提供更多的搜索时间和计算资源以及更好的优化基础，进而大幅提高了算法的搜索效率。

表 5 5 种算法在 12 个复杂优化函数上的运行时间对比/s

Fun	DMPSO	SinDE	ACS	GWO	GWODE
f_1	1.4697	1.0995	**0.9407**	1.2914	0.9493
f_2	1.6981	1.6041	**1.0285**	1.5103	1.1974
f_3	2.3324	1.8586	**1.7582**	2.0188	1.8102
f_4	1.6791	1.1832	**1.0898**	1.4565	1.1671
f_5	3.4419	1.0801	**0.9203**	2.2573	1.8657
f_6	3.2681	0.8792	**0.7831**	2.1096	1.7847
f_7	2.5997	0.0496	**2.0000**	2.2577	1.1562
f_8	4.1878	2.2163	**1.9774**	2.7694	2.3416
f_9	2.5315	2.0198	**1.8258**	2.2117	2.0532
f_{10}	2.5465	2.0243	**1.8271**	2.1762	2.0295
f_{11}	3.4684	2.8964	**2.7116**	3.1049	2.9493
f_{12}	2.9675	2.3505	**2.2468**	2.5611	2.4064



(a) f_1 (b) f_2 (c) f_3 (d) f_4

(e) f_5 (f) f_6 (g) f_7 (h) f_8

(i) f_9 (j) f_{10} (k) f_{11} (l) f_{12}

图 2 5 种算法在 f_1-f_{12} 上的收敛曲线

综上所述，本文提出的 GWODE 算法充分发挥了 GWO 所具备的搜索精度高、收敛速度快和 DE 算法在处理复杂优化函数时所具备的较好寻优效果的特点，并有效克服了各自的缺陷，使得新混合算法不论在面对单峰函数优化还是多峰及平移复杂函数优化问题时，都比 DMPSO，SinDE，GWO 和 ACS 具有更高的搜索精度和搜索效率以及更强的规避局部最优的能力，对于高维复杂函数的求解也体现出了更好的优化性能和普适性。

结束语 鉴于 GWO 优化精度高、收敛速度快，但对一些复杂优化函数却容易陷入局部最优，而 DE 对复杂优化函数有着较好的优化效果但局部搜索能力差和搜索效率偏低的情况，提出了灰狼优化与差分进化的混合优化算法。GWODE 首先采用嵌入趋优算子的 GWO 算法来更新种群位置，使其

(下转第 124 页)

[8] YIN R, YU G D, ZHONG C J, et al. Distributed resource allocation for D2D communication underlaying cellular networks[C]// IEEE International Conference on Communications Workshops (ICC). Budapest, 2013: 138-143.

[9] ZHANG H, WANG T, SONG L, et al. Graph-based resource allocation for D2D communications underlaying cellular networks [C]// IEEE/CIC International Conference on Communications in China-Workshops (CIC/ICCC). Xi'an, 2013: 187-192.

[10] GU Y, ZHANG Y, PAN M, et al. Matching and cheating in device to device communications underlying cellular networks[J]. IEEE Journal on Selected Areas in Communications, 2015, 33 (10): 2156-2166.

[11] GALE D, SHAPLEY L S. College admissions and the stability of marriage[J]. American Mathematical Monthly, 1962, 69(1): 9-15.

[12] IRVING R W, LEATHER P, GUSFIELD D. An efficient algorithm for the "optimal" stable marriage [J]. Journal of the ACM, 1987, 34(3): 532-543.

[13] 卢开澄, 卢华明. 图论及其应用(第 2 版)[M]. 北京: 清华大学出版社, 1995: 189-202.

[14] ITU-R. Guidelines for evaluation of radio interface technologies for IMT-Advanced: ITU-RM. 2135-1[R]. 2009.

(上接第 98 页)

在更短的时间内获得更好的优化精度和更快的收敛速度，以便将更多的搜索过程留给差分进化算法，有利于其对复杂优化问题的处理；然后通过采用自适应调节参数的差分策略来进一步提高算法对复杂优化函数的寻优性能，实现算法之间的优势互补，获得了一种全局搜索能力和局部搜索能力兼顾的高效混合优化算法。实验结果表明，GWODE 具有更快的收敛速度、更好的寻优性能和更强的普适性，更适用于求解各种函数优化问题。

参 考 文 献

[1] JIANG J Q, BO Y L, SONG C Y, et al. Hybrid algorithm based on particle swarm optimization and artifical fish swarm algorithm [J]. Lecture Notes in Computer Science, 2012, 7367: 607-614.

[2] LI Q, ZHAO C, CHEN P, et al. Improved ant colony optimization based on particle swarm optimization [J]. Control and Decision, 2013, 28(6): 873-878. (in Chinese)
李擎, 张超, 陈鹏, 等. 一种基于粒子群参数优化的改进蚁群算法[J]. 控制与决策, 2013, 28(6): 873-878.

[3] BEHNAMIAN J, ZANDIEH M, FATEMI G S M T. Parallel machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm [J]. Expert Systems with Applicatoins, 2009, 36(6): 9637-9644.

[4] ZHANG X M, TU Q, YIN X X, et al. Chemotaxis operator embedded particle swarm optimization algorithm and its application to multilevel thresholding[J]. Computer Science, 2016, 43(2): 311-315. (in Chinese)
张新明, 涂强, 尹欣欣, 等. 嵌入趋化算子的 PSO 算法及其在多阈值分割中的应用[J]. 计算机科学, 2016, 43(2): 311-315.

[5] ZHOU Y L. Hybrid strategy of intelligent optimization algorithm: analysis, design and model[J]. Application Research of Computers, 2010, 27(12): 4423-4426. (in Chinese)
周雅兰. 智能优化算法的混合策略分析、设计和建模[J]. 计算机应用研究, 2010, 27(12): 4423-4426.

[6] MIRJALILI S, MIRJALILI S M, LEWIS A. Grey wolf optimizer [J]. Advances in Engineering Software, 2014, 69(3): 46-61.

[7] MIRJALILI S. How effective is the grey wolf optimizer in training multi-layer perceptrons[J]. Applied Intelligence, 2015, 43 (1): 150-161.

[8] SULAIMAN M H, MUSTAFFA Z, MOHAMED M R, et al. Using the grey wolf optimizer for solving optimal reactive power dispatch problem[J]. Applied Soft Computing, 2015, 32: 286-292.

[9] SONG H M, SULAIMAN M H, MOHAMED M R. An application of grey wolf optimizer for solving combined economic emission dispatch problems[J]. International Review on Modelling and Simulations, 2014, 7(5): 838-844.

[10] LONG W, ZHAO D Q, XU S J. Improved grey wolf optimization algorithm for constrained optimization problem[J]. Journal of Computer Applications, 2015, 35(9): 2590-2595. (in Chinese)
龙文, 赵东泉, 徐松金. 求解约束优化问题的改进灰狼优化算法[J]. 计算机应用, 2015, 35(9): 2590-2595.

[11] GAO Y L, LIU J M. Dynamic differential evolution algorithm with random mutation[J]. Journal of Computer Applications, 2009, 29(10): 2719-2722. (in Chinese)
高岳林, 刘俊梅. 一种带有随机变异的动态差分进化算法[J]. 计算机应用, 2009, 29(10): 2719-2722.

[12] PILOT M, BRANICKI W, JEDRZEJEWSKI W, et al. Phylogeographic history of grey wolves in Europe [J]. Bmc Evolutionary Biology, 2010, 10(1685): 1-11.

[13] MURO C, ESCOBEDO R, SPECTOR L, et al. Wolf-pack (Canis lupus) hunting strategies emerge from simple rules in computational simulations [J]. Behavioural Processes, 2011, 88(3): 192-197.

[14] DRAA A, BOUZOUBIA S, BOUKHALFA I. A sinusoidal differential evolution algorithm for numerical optimization[J]. Applied Soft Computing, 2015, 27: 99-126.

[15] LI J Y, WANG Y, LI C L. Particle Swarm Optimization Algorithm with Double-Flight Modes[J]. Pattern Recognition and Artificial Intelligence, 2014, 27(6): 533-539. (in Chinese)
李景洋, 王勇, 李春雷. 采用双模飞行的粒子群优化算法[J]. 模式识别与人工智能, 2014, 27(6): 533-539.

[16] NAIK M, NATH M R, WUNNAVA A, et al. A new adaptive Cuckoo search algorithm [C] // 2015 IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS). IEEE, 2015: 1-5.

[17] HAKLI H, UGUZ H. A novel swarm optimization algorithm with Levy flight[J]. Applied Soft Computing, 2014, 23(5): 333-345.

[18] TANWEER M R, SURESH S, SUNDARARAJAN N. Self regulating particle swarm optimization algorithm [J]. Information Sciences, 2015, 294: 182-202.