文章编号: 1001-0920(2017)11-1985-06 DOI: 10.13195/j.kzyjc.2016.1250

基于参数动态调整的多目标差分进化算法

侯 莹†, 韩红桂, 乔俊飞

(1. 北京工业大学 信息学部, 北京 100124; 2. 计算智能与智能系统北京市重点实验室, 北京 100124)

摘 要: 针对多目标差分进化算法最优解难以获取的问题,提出一种基于参数动态调整的多目标差分进化(AMODE)算法. AMODE算法通过设计变异率和交叉率的自适应调整策略,实现进化过程中变异率和交叉率的动态调整,均衡多目标差分进化算法的局部搜索能力和全局探索能力,获得收敛性、多样性和均匀性较好的最优解. 实验结果表明,基于参数动态调整的AMODE算法能够有效改善多目标差分进化算法的逼近能力(IGD)和均匀性(SP),具有较好的优化效果.

关键词: 多目标优化; 差分进化算法; 参数动态调整; 自适应

中图分类号: TP173 **文献标志码:** A

Adaptive multi-objective differential evolution algorithm based on the dynamic parameters adjustment

HOU Ying†, *HAN Hong-gui*, *QIAO Jun-fei*

(1. Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China; 2. Beijing Key Laboratory of Computational Intelligence and Intelligent System, Beijing 100124, China)

Abstract: To obtain the optimal solutions of the multi-objective differential evolution algorithm, an adaptive multi-objective differential evolution(AMODE) algorithm based on the dynamic parameters adjustment strategies is developed, in which the adaptive adjustment strategies are designed to select the scaling factor and crossover rate. Then, the suitable scaling factor and crossover rate can be calculated in the mutation and crossover processes to balance the local search and the global exploration abilities of the multi-objective differential evolution algorithm. Thus, the integrity and uniformity optimal solutions are able to be obtained in the evolutionary process. The experimental results show that this proposed AMODE algorithm has a better effect to improve the inverted generational distance(IGD) and spacing(SP).

Keywords: multi-objective optimization; differential evolution algorithm; dynamic parameters adjustment; adaptive

0 引 言

差分进化算法(DE)由于其机理简单、受控参数少、鲁棒性强等特点,已成功应用于非线性优化控制、约束优化计算、神经网络优化等问题[1-3]. 近年来,随着科学研究与工程实践中系统的规模不断增大、约束条件不断增多,优化问题变得越来越复杂,需要同时对多个目标进行优化. 由于多目标优化中各目标之间往往相互矛盾,相互冲突,对其中一个目标优化必须以牺牲其他目标为代价,单目标DE已很难满足实际的需要[4-5].

为了对多目标优化问题进行求解,多目标差分进化算法(MODE)已成为国际学术界的一个研究热点[6-7]. 优化问题的复杂化对MODE性能提出了更高的要求,Elsayed等提出了一种改进型自适应MODE,通过使用混合变异算子策略提高算法的全局探索能力,并引入自适应进化策略改善了其局部搜索能力,取得了较好的优化结果[8]; Bandyopadhyay等提出了一种α-DEMO算法,α-DEMO算法通过记录多目标间的冲突状态,并对冲突目标进行参数补偿,有效地改善了MODE的局部搜索能力[9]; 通过将混沌引入种群的初始化,章萌等提出了一种多目标强度Pareto混沌差分进化算法,同时设计出均匀排挤机制和混沌替换方式进行种群的选择,并利用变缩放因子变异策略对优化过程进行了改进,保证了Pareto最优解

收稿日期: 2016-10-03; **修回日期:** 2017-02-17.

基金项目: 国家自然科学基金项目(61533002, 61622301); 中国博士后科学基金项目(2014M550017); 教育部博士点基金项目(20131103110016); 北京市教委项目(KM201410005001, KZ201410005002).

作者简介: 侯莹(1982–), 女, 博士生, 从事智能计算、复杂过程智能优化的研究; 韩红桂(1983–), 男, 教授, 博士生导师, 从事污水处理过程建模、优化与控制等研究.

†通讯作者. E-mail: houying17@sina.com

的均匀分布性[10];姚峰等提出了一种新型MODE,利用非支配排序和拥挤度距离进行综合考虑,提高了最优解分布的多样性,并保证了算法收敛到Pareto最优解[11]. 以上几种改进型MODE在一定程度上提高了算法的搜索能力,改善了算法的性能. 然而,由于以上几种MODE的改进策略需要引入较多的参数,从而增加了优化算法的计算量,降低了优化求解的速度[12-13]. 为了在提高搜索能力的同时,降低MODE的计算量,Chen等提出了一种MODE-RMO,利用有效解的信息设计了一种变异算子排序策略,加快了MODE的求解速度,获得了较好的效果[14]. 另外,基于种群多样性的特点,Wang等提出了一种基于多目标排序变异策略的MODE,利用当前种群个体的非支配排序贡献值选择合适的子代种群,由于同时考虑适应度值和多样性信息,改进后的MODE算法具有较快的收敛速度[15];围绕存在噪音的多目标优化问题,在综合考虑算法适应度值和计算量的情况下,Rakshit等提出了一种扩展型MODE,利用Pareto最优解的概率排序选择合适的变异和交叉策略,提高了MODE算法的寻优速度[16]. 以上几种改进型MODE在一定程度上减少了算法的计算量,提高了算法的搜索速度. 然而,如何快速获得完整和均匀的最优解依然是一个开放的问题[17-20].

本文针对MODE存在的问题,提出一种基于参数动态调整的多目标差分进化(AMODE)算法,通过设计变异率和交叉率的自适应调整策略,实现局部搜索和全局探索能力的平衡. 最后利用标准的测试函数对其性能进行分析,并通过实验验证所提出算法的有效性.

1 多目标差分进化算法

1.1 多目标优化问题描述

多目标优化问题中的优化目标通常定义为

$$\begin{aligned}&\boldsymbol{G}(\boldsymbol{x})=g_k(\boldsymbol{x}),k=1,2,\cdots,K,\\&\boldsymbol{x}=[x_1,x_2,\cdots,x_D]\in U;\\&\text{s.t. }x_{jL}<x_j<x_{jH},\ j=1,2,\cdots,D.\end{aligned}\tag{1}$$

其中:$\boldsymbol{G}(\boldsymbol{x})$是优化目标向量;$g_k(\boldsymbol{x})$是优化目标向量中第$k$个目标函数;$K$是优化目标个数;$\boldsymbol{x}$是在可行域$U$中的最优解集;$x_j$是$\boldsymbol{x}$中的第$j$个分量,即第$j$个决策变量,$x_{jL}$是其最小值,$x_{jH}$是其最大值.

由于多目标优化问题中各个目标间是相互冲突的,最优解一般是一个解集,称为Pareto解集(或非劣解集). 以求最小值优化问题为例,Pareto解集的定义为:若$\boldsymbol{x}^*$在所有目标函数上的值均不大于$\boldsymbol{x}$在相应目标函数上的值,而且$\boldsymbol{x}^*$至少在一个目标函数上的值严格小于$\boldsymbol{x}$在此目标函数上的值,则称$\boldsymbol{x}^*$Pareto占优$\boldsymbol{x}$. 如果$\boldsymbol{x}^*$在整个可行域U中,没有被任何变量Pareto占优,则$\boldsymbol{x}^*$为Pareto解. 所有的Pareto解构成多目标优化的最优解集,即Pareto前沿.

1.2 多目标差分进化算法

多目标差分进化算法主要用于对包含两个或两个以上的待优化目标函数进行优化,是一种并行直接搜索方法,其主要操作如下:

1) 参数设置及种群初始化. 设定种群规模NP,最大进化代数$T_{\max}$,初始化变异率F,交叉率C_r. 令初始进化代数$T=0$,随机产生D维初始化种群

$$\boldsymbol{x}_i^0=[x_{1,i}^0,x_{2,i}^0,\cdots,x_{D,i}^0],\ i=[1,2,\cdots,\text{NP}].\tag{2}$$

2) 变异操作. 变异操作在生物学中相当于基因突变,在进化计算模式背景下,是指对某个随机因素的改变或者扰动. 变异策略表达式为

$$\boldsymbol{x}_i^{\text{T}+1}=\boldsymbol{x}_i^{\text{T}}+F(\boldsymbol{x}_{r_1}^{\text{T}}-\boldsymbol{x}_{r_2}^{\text{T}}).\tag{3}$$

其中:$\boldsymbol{x}_i^{\text{T}+1}$是子代第$i$个个体,$\boldsymbol{x}_i^{\text{T}}$是父代第$i$个个体,$\boldsymbol{x}_{r_1}^{\text{T}}$和$\boldsymbol{x}_{r_2}^{\text{T}}$是两个不同的个体,$F$是变异率,$r_1$和$r_2$是在$[1,\text{NP}]$中随机选取的不同于$i$的两个互不相同的实数.

3) 交叉操作. 交叉操作能够增加种群的多样性,是生成新个体的主要方法,其策略为

$$\bar{x}_{ji}^{\text{T}+1}=\begin{cases}x_{ji}^{\text{T}+1},\ \text{rand}[0,1]\leqslant C_r;\\x_{ji}^{\text{T}},\ \text{others}.\end{cases}\tag{4}$$

其中:$\bar{x}_{ji}^{\text{T}+1}$是产生的实验个体;rand$[0,1]$是$[0,1]$之间均匀分布的随机数;$j=1,2,\cdots,D$是随机选择的一个标志位,以便保证进化过程中能够从父代中继承至少一位元素.

4) 选择操作. 通过比较子代与父代个体的目标函数值,判断父代个体是否需要被子代个体所替代,以此推动种群的进化. 当求取最小化问题时,选择操作如下:

$$\boldsymbol{x}_i^{\text{T}+1}=\begin{cases}\bar{\boldsymbol{x}}_i^{\text{T}+1},\ g_k(\bar{\boldsymbol{x}}_i^{\text{T}+1})\leqslant g_k(\boldsymbol{x}_i^{\text{T}});\\\boldsymbol{x}_i^{\text{T}},\ g_k(\bar{\boldsymbol{x}}_i^{\text{T}+1})>g_k(\boldsymbol{x}_i^{\text{T}}).\end{cases}\tag{5}$$

当子代个体的适应度值不高于父代个体的适应度值时,子代个体取代父代个体,否则保持原父代个体不变.

5) 终止条件判断. 若$T<T_{\max}$,则返回2),否则将生成的子代种群中优劣等级最高的所有个体构成多目标优化问题的最优解集,输出并结束运行.

2 自适应多目标差分进化算法

多目标差分进化算法中,最优解的性能主要取决于变异和交叉两个环节. 其中,变异和交叉过程中的

可变参数包括变异率F和交叉率C_r. 变异率F影响多目标差分进化算法的搜索范围,交叉率C_r决定多目标差分进化算法的搜索方向. 为了选择合理的搜索范围和方向,平衡多目标差分进化算法的局部搜索与全局探索能力,加快算法的收敛速度并提高最优解的性能,通过自适应动态调整变异率和交叉率来提高多目标差分进化算法的搜索精度和收敛性.

2.1　变异率动态调整策略设计

多目标差分进化算法中,变异率用以对差分向量进行放大或缩小,一般在算法进化初期希望有较大的变异率,保证种群的多样性,避免算法陷入局部最优;在算法进化的后期则希望有较小的变异率,使得进化过程中优良个体得以有效保留,提高算法的搜索效率. 因此,变异率的有效调整能促进局部搜索与全局探索能力达到平衡. 考虑到在多目标优化过程中,每一代产生多个个体,由于整个种群及个体自身进化程度不同,每个个体采用不同的变异率,更有利于种群的进化. 对于求解最小化问题,变异率动态调整策略可表示为

$$F_i^{\mathrm{T}+1} = F_i^{\mathrm{T}}[\mu_L + (\mu_H - \mu_L)\theta_i]. \tag{6}$$

其中

$$\theta_i = f_w^{\mathrm{T}} - f_a^{\mathrm{T}}, \tag{7}$$

f_a^{T}和f_w^{T}分别是父代个体中最优和最差的个体适应度,$F_i^{\mathrm{T}+1}$是子代中第i个个体的变异率,F_i^{T}是父代中第i个个体的变异率,μ_H和μ_L分别是变异率的上限和下限. 同时,AMODE算法中子代变异率可表示为

$$\boldsymbol{F}^{\mathrm{T}+1} = [F_1^{\mathrm{T}+1}, F_2^{\mathrm{T}+1}, \cdots, F_D^{\mathrm{T}+1}]. \tag{8}$$

根据式(6),在AMODE算法的优化过程中,多目标差分进化算法的变异率$\boldsymbol{F}$不但能保证进化初期具有较大的值,使种群具有较好的多样性,同时能保证进化后期变异率具有较小的值,保留种群中的优良个体,使得多目标差分进化算法具有较好的搜索效率.

2.2　交叉率动态调整策略设计

多目标差分进化算法中,交叉率C_r对算法的搜索能力和收敛性也有较大的影响,进化过程中动态调整多目标差分进化算法的交叉率,能够有效地改进多目标差分进化算法的优化效果.

AMODE算法进化过程中,交叉率C_r主要基于父代个体适应度与群体平均适应度的相对值来调整,对于求解最小化问题,其更新策略为

$$C_{ri}{}^{\mathrm{T}+1} = \begin{cases} C_{ri}{}^{\mathrm{T}}, \ f_i^{\mathrm{T}} < f_m^{\mathrm{T}}; \\ C_{ri}{}^{\mathrm{T}}[\rho_L + (\rho_H - \rho_L)\theta_i], \ f_i^{\mathrm{T}} \geqslant f_m^{\mathrm{T}}. \end{cases} \tag{9}$$

其中:$C_{ri}{}^{\mathrm{T}+1}$是子代中第i个个体的交叉率,$C_{ri}{}^{\mathrm{T}}$是父代中第i个个体的交叉率,f_i^{T}是父代第i个个体的适应度值,f_m^{T}是父代个体的平均适应度值,ρ_H和ρ_L分别是交叉率的上限和下限.

多目标差分进化算法的交叉率C_r服从高斯分布$N(C_{rm}, \delta^2)$时具有较好的性能. 其中:C_{rm}是高斯分布的中心,是一个进化周期内多目标差分进化算法产生的所有优秀个体交叉率C_r的均值,决定了概率密度函数的位置;δ是高斯分布的宽度,决定了高斯分布的幅度. 为了进一步提高算法的搜索精度,所提出的AMODE算法对交叉率的高斯分布宽度δ进行设计,根据种群进化程度的过程信息对δ进行调整:当进化产生的优秀个体数量增加时,减少高斯分布的宽度δ,缩小交叉率C_r的取值范围;当优秀个体数量不变时,高斯分布宽度δ不变;当优秀个体数量减少时,增大高斯分布的宽度δ,提高交叉率C_r的取值范围,扩大最优解的搜索区间.

基于以上分析,所提出的AMODE算法在进化过程中,交叉率高斯分布宽度δ的自适应调整策略为

$$\delta^{\mathrm{T}+1} = \begin{cases} \alpha\delta^{\mathrm{T}}, \ N^{\mathrm{T}+1} - N^{\mathrm{T}} > 0; \\ \delta^{\mathrm{T}}, \ N^{\mathrm{T}+1} - N^{\mathrm{T}} = 0; \\ \beta\delta^{\mathrm{T}}, \ N^{\mathrm{T}+1} - N^{\mathrm{T}} < 0. \end{cases} \tag{10}$$

其中:$\delta^{\mathrm{T}+1}$是子代交叉率C_r的高斯分布宽度;δ^{T}是父代交叉率C_r的高斯分布宽度;$N^{\mathrm{T}+1}$和N^{T}分别是子代和父代的优秀个体数量;α和β是修正参数,$0 < \alpha < 1, \beta > 1$.

2.3　自适应多目标差分进化算法的实现流程

有效利用种群个体适应度值及种群整体进化程度信息,动态调整进化过程中的变异率和交叉率能够最大限度地提高对进化过程信息的利用率,自适应调节优化搜索的范围和方向.

式(6)和(9)根据进化状态对变异率和交叉率进行自适应调整,增强了多目标差分进化算法的搜索能力和收敛速度. 同时,式(10)对多目标差分进化算法交叉率的高斯分布宽度进行了自适应调整,有利于提高多目标差分进化算法的搜索精度.

所提出的AMODE算法主要流程如下.

Step 1: 参数设置及种群初始化:选定种群规模NP,最大进化代数$T_{\max}$,确定变异率F_j,交叉率$C_{rj}(j = 1, 2, \cdots, D)$,根据式(2)随机产生$D$维初始化种群,令$T = 0$进行优化.

Step 2: 变异操作:利用进化过程信息,根据式(6)和(7)对变异率进行调整;同时,根据式(3)进行变异操作.

Step 3: 交叉操作:根据式(9)和(10)对交叉率进行调整;同时,根据式(4)进行交叉操作.

Step 4: 选择操作: 根据式(5)比较新个体的适应度值, 基于个体的适应度值进行选择.

Step 5: 终止判断: 若$T < T_{\max}$, 则返回Step 2继续寻优; 否则终止计算, 并输出结果.

由式(6)和(9)可见, 所提出的AMODE算法搜索到的空间更为广阔, 更易跳出局部最优, 且求解精度有所提高, 更容易得到完整和均匀的Pareto解集.

在算法进化过程中, AMODE算法的变异率和交叉率可根据进化信息自适应调整, 利用种群信息实时确定变异率和交叉率, 不但可以避免进化过程早熟收敛, 而且有效减小了参数初始设定值对算法性能的影响, 提高了多目标差分进化算法求解问题的通用性.

3 仿真实验

为了验证本文所提出的变异率和交叉率自适应策略的有效性, 应用3个标准测试函数ZDT1、ZDT2和DTLZ2对所设计的AMODE算法性能进行测试. 其中: 测试函数ZDT1和ZDT2为双目标测试函数, 测试函数DTLZ2为三目标测试函数.

为了保证测试结果的公平性, 测试函数决策变量的维数与比较算法一致, 双目标测试函数ZDT1和ZDT2的决策变量维数取为30; 三目标测试函数DTLZ2的决策变量维数取为12.

3.1 实验设计

为了描述多目标差分进化算法的性能, 从收敛性、多样性和均匀性3个角度对算法进行评价. 收敛性和多样性采用反转世代距离(IGD)指标度量, 均匀性采用间距(SP)指标度量, 即

$$\mathrm{IGD}(\boldsymbol{P}^*, \boldsymbol{P}) = \sum_{x\in p^*} \min \mathrm{dis}(\boldsymbol{x}, \boldsymbol{P})/|\boldsymbol{P}^*|, \tag{11}$$

$$\mathrm{SP} = \sqrt{(Z-1)^{-1}\sum_{i=1}^{k}(\bar{d}-d_i)}. \tag{12}$$

其中: $\boldsymbol{P}$是真实Pareto最优解集, $\boldsymbol{P}^*$是进化计算获得的近似Pareto最优解集, $\min \mathrm{dis}(\boldsymbol{x}, \boldsymbol{P})$是$\boldsymbol{x}$与$\boldsymbol{P}$的最小欧氏距离, d_i是第i个解与其最近解之间的欧氏距离, Z是非支配解的个数, $\bar{d}$是所有解欧氏距离的平均距离.

IGD指标是表征进化计算获得的近似Pareto最优解与真实Pareto最优解之间的距离指标, 该指标数值越小, 表明近似Pareto最优解越接近真实Pareto最优解, 算法的收敛性和多样性越好. SP指标是表征最优解集分布均匀性的性能指标, 该指标数值越小, 表明解集的均匀性越好.

实验选取的3个标准测试函数ZDT1、ZDT2和DTLZ2具体描述如下.

测试函数ZDT1为

$$\begin{cases} f_1(x) = x_1, \\ f_2(x) = g(x)(1-\sqrt{(x_1/g(x))}), \\ g(x) = 1 + 9\sum\limits_{i=2}^{m} x_i/(m-1); \\ \text{s.t. } m = 30,\ 0 \leqslant x_i \leqslant 1,\ i = 1, 2, \cdots, m. \end{cases} \tag{13}$$

测试函数ZDT2为

$$\begin{cases} f_1(x) = x_1, \\ f_2(x) = g(x)(1-(x_1/g(x))^2), \\ g(x) = 1 + 9\sum\limits_{i=2}^{m} x_i/(m-1); \\ \text{s.t. } m = 30,\ 0 \leqslant x_i \leqslant 1,\ i = 1, 2, \cdots, m. \end{cases} \tag{14}$$

测试函数DTLZ2为

$$\begin{cases} f_1(x) = \cos\left(\dfrac{\pi}{2}x_1\right)\cos\left(\dfrac{\pi}{2}x_2\right)(1+g(x)), \\ f_2(x) = \cos\left(\dfrac{\pi}{2}x_1\right)\sin\left(\dfrac{\pi}{2}x_2\right)(1+g(x)), \\ f_3(x) = \sin\left(\dfrac{\pi}{2}x_1\right)(1+g(x)), \\ g(x) = \sum\limits_{i=3}^{m}(x_i-0.5)^2; \\ \text{s.t. } m = 12,\ 0 \leqslant x_i \leqslant 1,\ i = 1, 2, \cdots, m. \end{cases} \tag{15}$$

为了有效评价所提出的AMODE算法的性能, 对每个测试函数分别进行20次独立实验, 将实验的平均值作为实验结果进行比较.

为了使结果公平, 测试函数的种群规模和最大进化代数与比较算法一致. 其中: 对于双目标测试函数, 种群规模NP设为100; 对于三目标测试函数, 种群规模NP设为300; 所有优化算法的最大进化代数$T_{\max}$设为250; AMODE算法的初始交叉率设为0.5, 变异率为0.2; 其他算法的初始参数和原文中一致.

3.2 实验结果

基于以上实验设计, 使用所提出的AMODE算法对3个标准测试函数分别进行实验, 图1 ~ 图3给出了AMODE算法寻找到的最优解集与真实Pareto最优解集的对比结果. 在图3中: × 表示AMODE, · 表示Pareto front. 由图1 ~ 图3可知, 所提出的AMODE算法找到的最优解能较好地逼近真实Pareto最优解,

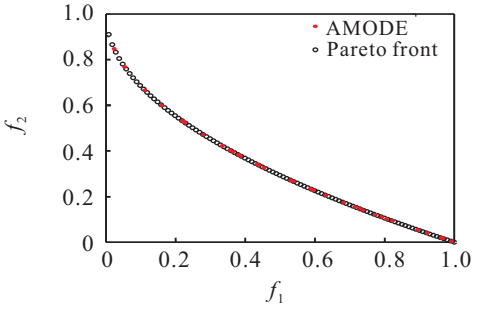

图 1 ZDT1函数最优解集与真实Pareto最优解集对比

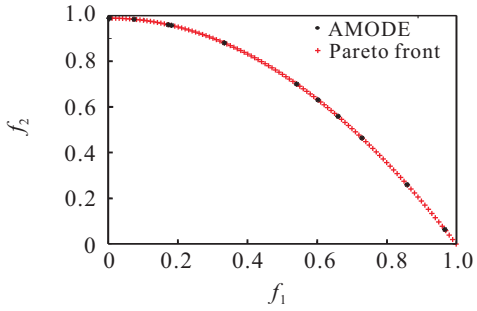

图 2　ZDT2 函数最优解集与真实 Pareto 最优解集对比

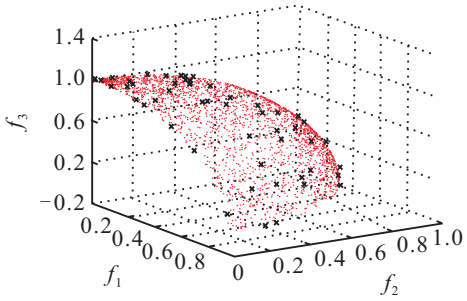

图 3　DTLZ2 函数最优解集与真实 Pareto 最优解集对比

AMODE 算法找到的最优解的值与真实 Pareto 最优解的值基本吻合. 同时, 由图 1 ～ 图 3 可知, AMODE 算法寻找到的最优解基本均匀地布满 Pareto 最优前沿, 具有较好的收敛性、多样性和均匀性.

为进一步分析 AMODE 算法的性能, 将该算法对 3 个标准测试函数的寻优结果与其他典型多目标优化算法进行比较: α-DEMO 算法[9]、MODE-RMO 算法[14]、非支配排序遗传算法 (NSGA-II)[21] 和多目标粒子群算法 (MOPSO)[22], 比较结果如表 1 和表 2 所示.

表 1 和表 2 分别给出了 5 种多目标优化算法的收敛性指标 IGD 和均匀性指标 SP 的均值和方差, 其中 $*$ 表示与原文的数据相同. 由表 1 可以看出: 所提出的 AMODE 算法对 3 个标准测试函数 ZDT1、ZDT2 和 DTLZ2 优化时的 IGD 平均值分别为 1.32×10^{-3}、2.26×10^{-4} 和 3.15×10^{-3}, 该值比 α-DEMO 算法、NSGA-II 算法和 MOPSO 算法的 IGD 平均值都小, 在对 DTLZ2 优化时的 IGD 平均值略大于 MODE-RMO 算法对 DTLZ2 优化时的 IGD 平均值; 同时, IGD 的方差分别为 1.52×10^{-4}、2.01×10^{-4} 和 4.58×10^{-5}, 比 α-DEMO 算法和 NSGA-II 算法的 IGD 方差都小. 因此, 所提出的 AMODE 算法具有较好的收敛性. 由表 2 可以看出: 所提出的 AMODE 算法对 3 个标准测试函数 ZDT1、ZDT2 和 DTLZ2 优化时的 SP 平均值分别为 2.57×10^{-3}、1.12×10^{-3} 和 6.51×10^{-2}, 比 α-DEMO 算法、MODE-RMO 算法、NSGA-II 算法和 MOPSO 算法的 SP 平均值都小; 同时, SP 的方差分别为 5.43×10^{-4}、3.34×10^{-4} 和 4.74×10^{-3}, 比 α-DEMO 算法、MODE-RMO 算法和 NSGA-II 算法的 SP 方差都小, 只有对 DTLZ2 优化时的 SP 平均值略大于 MOPSO 算法对 DTLZ2 优化时 SP 平均值. 因此, 所提出的 AMODE 算法具有较好的均匀性.

以上实验结果表明, 所提出的 AMODE 算法具有较好的收敛性和多样性, 能够有效地获得双目标函数和三目标函数的最优解, 获得的解具有较好的收敛性、多样性和均匀性.

表 1　不同优化算法的 IGD 对比

Function	Index	AMODE	α-DEMO[9]	MODE-RMO[14]	NSGA-II[21]	MOPSO[22]
ZDT1	Mean	1.32×10^{-3}	1.63×10^{-3}	$1.62 \times 10^{-3*}$	$5.74 \times 10^{-3*}$	$2.14 \times 10^{-3*}$
	SD	1.52×10^{-4}	1.89×10^{-4}	$1.11 \times 10^{-4*}$	$3.39 \times 10^{-4*}$	$1.69 \times 10^{-4*}$
ZDT2	Mean	2.26×10^{-4}	2.97×10^{-3}	$2.95 \times 10^{-3*}$	$5.35 \times 10^{-3*}$	$4.13 \times 10^{-3*}$
	SD	2.01×10^{-4}	2.81×10^{-4}	$2.78 \times 10^{-4*}$	$2.02 \times 10^{-4*}$	$4.66 \times 10^{-4*}$
DTLZ2	Mean	3.15×10^{-3}	5.01×10^{-3}	$1.01 \times 10^{-3*}$	$2.06 \times 10^{-1*}$	$1.03 \times 10^{-2*}$
	SD	4.58×10^{-5}	7.25×10^{-5}	$3.25 \times 10^{-4*}$	$5.78 \times 10^{-3*}$	$4.56 \times 10^{-5*}$

表 2　不同优化算法的 SP 对比

Function	Index	AMODE	α-DEMO[9]	MODE-RMO[14]	NSGA-II[21]	MOPSO[22]
ZDT1	Mean	2.57×10^{-3}	$2.69 \times 10^{-2*}$	$3.77 \times 10^{-2*}$	$5.28 \times 10^{-2*}$	$3.73 \times 10^{-2*}$
	SD	5.43×10^{-4}	$6.83 \times 10^{-4*}$	1.13×10^{-3}	$9.38 \times 10^{-3*}$	$2.67 \times 10^{-3*}$
ZDT2	Mean	1.12×10^{-3}	$8.62 \times 10^{-3*}$	1.67×10^{-2}	$7.24 \times 10^{-3*}$	$1.09 \times 10^{-2*}$
	SD	3.34×10^{-4}	$5.94 \times 10^{-4*}$	5.31×10^{-4}	$7.41 \times 10^{-4*}$	$1.04 \times 10^{-3*}$
DTLZ2	Mean	6.51×10^{-2}	$5.61 \times 10^{-1*}$	5.23×10^{-1}	$5.06 \times 10^{-1*}$	$3.52 \times 10^{-1*}$
	SD	4.74×10^{-3}	$5.65 \times 10^{-3*}$	4.82×10^{-3}	$5.78 \times 10^{-3*}$	$4.22 \times 10^{-3*}$

4　结　论

本文围绕多目标差分进化算法最优解的获取问题, 提出了一种基于变异率和交叉率动态调整的 AMODE 算法. 该算法根据进化过程信息设计变异率和交叉率的动态调整策略, 实现了 AMODE 算法在变异和交叉过程中参数的自适应调整, 提高了算法的

局部搜索能力和全局探索能力,获得了收敛性、多样性和均匀性较好的最优解. 利用标准测试函数进行仿真实验,实验所提出的AMODE算法具有如下特点: 1) AMODE算法的变异率和交叉率能够根据上一代个体进化程度和群体进化状态,进行自适应调整,充分利用了优化算法的过程信息,解决了初始参数难以设定的问题; 2) AMODE算法在保证最优解收敛性、多样性的同时,改善了最优解分布的均匀性,平衡了多目标差分进化算法的局部搜索能力和全局探索能力,提高了算法的寻优能力; 3) 在双目标函数和三目标函数的最优解求解过程中, AMODE算法的IGD和SP值优于其他常用的几种多目标优化算法,验证了AMODE算法具有较好的收敛性和多样性.

整体而言,基于进化过程信息的变异率和交叉率动态调整策略能够有效地解决多目标差分进化算法最优解的获取问题,所提出的AMODE算法能够有效地用于双目标和三目标问题的求解. 同时,根据进化过程信息设计变异率和交叉率动态调整策略的思路,可为其他优化算法提供有益的尝试. 但是,由于变异率和交叉率自适应策略需要增加一定的计算量,所提出的AMODE算法运行时间较长,未来将在保证算法现有优势的前提下,降低其计算复杂度.

参考文献(References)

[1] Tang L X, Dong Y, Liu J Y. Differential evolution with an individual-dependent mechanism[J]. IEEE Trans on Evolutionary Computation, 2015, 19(4): 560-574.

[2] Das S, Suganthan P N. Differential evolution: A Survey of the state-of-the-art[J]. IEEE Trans on Evolutionary Computation, 2011, 15(1): 4-31.

[3] Yang Q W, Cai L, Xue Y C. Asurvey of differential evolution algorithms[J]. Pattern Recognition and Artificial Intelligence, 2008，21(4): 506-513.

[4] Guo S M, Yang C C. Enhancing differential evolution utilizing eigenvector-based crossover operator[J]. IEEE Trans on Evolutionary Computation, 2015, 19(1): 31-49.

[5] Wang X Z, Li P, Yu G Y. Multi-objective chaotic differential evolution algorithm with grading second mutation[J]. Control and Decision, 2011, 26(3): 457-463.

[6] Gong M G, Jiao L C, Yang D D, et al. Research on evolutionary multi-objective optimization algorithms[J]. J of Software, 2009, 20(2): 271-289.

[7] Tang L X, Zhao Y, Liu J Y. An improved differential evolution algorithm for practical dynamic scheduling in steelmaking-continuous casting production[J]. IEEE Trans on Evolutionary Computation, 2014, 18(2): 209 -225.

[8] Elsayed S M, Sarker R A, Essam D L. An improved self-adaptive differential evolution algorithm for optimization problems[J]. IEEE Trans on Industrial Informatics, 2013, 9(1): 89-99.

[9] Bandyopadhyay S, Mukherjee A. An algorithm for many-objective optimization with reduced objective computations: A study in differential evolution[J]. IEEE Trans on Evolutionary Computation, 2015, 19(3): 400-413.

[10] Zhang M, Zhang W G, Sun Y. Mutli-objective strength Pareto chaotic differential evolution algorithm[J]. Control and Decision, 2012, 27(1): 41-46.

[11] Yao F, Yang W D, Zhang M. Multi-objective differential evolution used for load distribution of hot strip mills[J]. Control Theory & Applications, 2010, 27(7): 897-902.

[12] Ye H T, Luo F, Xu Y G. Differential evolution for solving multi-objective optimization problems: A survey of the state-of-the-art[J]. Control Theory & Applications, 2013, 30(7): 922-928.

[13] Abaci K, Yamacli V. Differential search algorithm for solving multi-objective optimal power flow problem[J]. Int J of Electrical Power & Energy Systems, 2016, 79(1): 1-10.

[14] Chen X, Du W L, Qian F. Multi-objective differential evolution with ranking-based mutation operator and its application in chemical process optimization[J]. Chemometrics & Intelligent Laboratory Systems, 2014, 136(16): 85–96.

[15] Wang J H, Liao J J, Zhou Y, et al. Differential evolution enhanced with multiobjective sorting-based mutation operators[J]. IEEE Trans on Cybernetics, 2014, 44(12): 2792-2805.

[16] Rakshit P, Konar A. Extending multi-objective differential evolution for optimization in presence of noise[J]. Information Sciences, 2015, 305(1): 56-76.

[17] Rakshit P, Konar A. Differential evolution for noisy multiobjective optimization[J]. Artificial Intelligence, 2015, 227(1): 165-189.

[18] Qi Y T, Hou Z T, Yin M L, et al. An immune multi-objective optimization algorithm with differential evolution inspired recombination[J]. Applied Soft Computing, 2015, 29(1): 395-410.

[19] Chen Y G, Mahalec V, Chen Y W, et al. Reconfiguration of satellite orbit for cooperative observation using variable-size multi-objective differential evolution[J]. European J of Operational Research, 2015, 242(1): 10-20.

[20] Yang M, Li C H, Cai Z H, et al. Differential evolution with auto-enhanced population diversity[J]. IEEE Trans on Cybernetics, 2014, 45(2): 302-315.

[21] Martin D, Rosete A, Alcala F J, et al. A new multiobjective evolutionary algorithm for mining a reduced set of interesting positive and negative quantitative association rules[J]. IEEE Trans on Evolutionary Computation, 2014, 18(1): 54-69.

[22] Elhossini A, Areibi S, Dony R. Strength pareto particle swarm optimization and hybrid EA-PSO for multi-objective optimization[J]. Evolutionary Computation, 2010, 18(1): 127-156.

（责任编辑：闫 妍）