基于全局灵敏度分析的改进微分进化算法

何小龙, 白俊强, 李宇飞

(西北工业大学 航空学院, 陕西 西安 710072)

摘 要:为了提高微分进化算法在问题维度较高、计算量有限的情况下的寻优能力,提出了基于全局灵敏度分析方法的改进微分进化算法。使用 Morris-One-at-a-Time (MOAT)全局灵敏度方法对典型函数进行灵敏度分析,与另一种全局灵敏度分析方法 Sobol 方法进行了对比,证明 MOAT 方法计算效率和精度较高。基于 MOAT 方法分别构造了考虑灵敏度信息的改进交叉算子和变异算子,从而得到 2 种改进微分进化算法 GSADE1 和 GSADE2。使用 5 个显式 50 维测试函数对 2 种改进算法进行了测试,并与基准微分进化算法进行了对比,发现 2 种改进算法在收敛速度和鲁棒性方面都有所提高。

关 键 词:算法; 全局优化; 微分进化; 计算机仿真; 灵敏度分析
中图分类号:TP301.6 **文献标志码**:A **文章编号**:1000-2758(2016)03-0411-07

微分进化算法(differential evolution, DE)与进化规划、遗传规划、遗传算法类似,都使用了优胜劣汰、适者生存的思想,是近年来发展的一种性能突出的进化算法(evolutionary algorithm)。DE 由 Rainer Stom 和 Kenneth Price 于 1995 年提出[1],实践发现其收敛速度快、可调参数少、鲁棒性好、算法简单,在计算机、控制工程、化工、电力等众多学科中得到了广泛应用。国内外有许多学者进行过研究,苏海军、刘波、Swagatam Das 等人[2-4]分别对 DE 算法研究进行了综述。

但是 DE 还存在一些问题,一方面是收敛性问题,DE 也存在着早熟和局部搜索能力弱的缺点[5],而且在问题维度较高时更为严重,需要极高的计算量来保证优化效果;另一方面是如何针对特定问题设置合理的参数来保证良好的算法性能。为了继续提高 DE 的性能,有许多学者致力于对 DE 进行改进,一种思路是对已有操作算子进行改进,如 Fan 等人提出的三角法变异算子[6],另一种思路是提出新的操作算子,如 Wang 等提出的加速和迁移操作[7]研究。

本文针对以上两方面问题,提出将全局灵敏度分析方法(global sensitivity analysis, GSA)与 DE 相结合的改进思路。首先,计算灵敏度的过程就是从问题中提取更多信息的过程,用灵敏度信息来和优化算法结合,有助于算法捕捉问题的特征,从而提高算法性能。其次,通过构造新的算子公式,使用灵敏度信息对算法参数进行控制,一定程度上解决了算法参数设置的问题。

1 基本微分进化算法

DE 的变异、交叉、选择 3 种算子是算法的核心操作,下面予以简要介绍。设种群大小为 N_P,D 为变量的维数, 第 n 代种群为 $\{x_1^{(n)}, x_2^{(n)}, \cdots x_i^{(n)} \cdots, x_{NP}^{(n)}\}$,第 n 代种群中的第 i 个个体可表示为 $x_i^{(n)} = (x_{i,1}^{(n)}, x_{i,2}^{(n)}, \cdots, x_{i,D}^{(n)})$。

1.1 变异算子

在 DE 中,变异个体的生成过程使用到了父代种群中多个个体的线性组合。微分进化算法使用如下公式产生变异个体 $v_i^{(n+1)}$

$$v_i^{(n+1)} = x_{r1}^{(n)} + F \cdot (x_{r2}^{(n)} - x_{r3}^{(n)}) \tag{1}$$

$x_{r1}^{(n)}$、$x_{r2}^{(n)}$、$x_{r3}^{(n)}$ 是从父代种群中按特定规则选择的 3 个不同个体,这里有一个暗含的要求是种群大小需要大于 4,否则这里的随机性就无法保证了。F 是一个常数,称为放缩因子,用来控制变异的比例。

收稿日期:2015-10-22 **基金项目**:国家“973”计划(2014CB744804)资助
作者简介:何小龙(1989—),西北工业大学博士研究生,主要从事飞行器气动外形优化设计及优化方法研究。

在种群进化过程中,如果已经靠近最优解,那么个体之间的差分值会逐渐减小,那么这种扰动就会自动变小。

放缩因子 F 较小会使得算法变异能力不足、对优化空间的探索能力不足,从而导致容易陷入局部最优;较大的 F 提高了算法搜索到全局最优的概率,但是当 F 较大时,算法的收敛速度会明显降低,这是因为当差分向量乘以 F 后会使变异个体距离父代较远,对父代个体的继承性较差,收敛速度会明显下降。根据一些学者针对测试函数的使用经验,放缩因子的选取范围为 0.5~0.9。

Price 和 Storn 对基本 DE 的变异公式进行微调,提出了几种不同的变异操作算子,具体细节的不同可用符号 DE/*X*/*Y*/*Z* 来区分。Price 曾提出 DE/BEST/2/BIN 是一种比较值得研究的算子,本文采用这种算子作为基础进行改进。

1.2 交叉算子

交叉运算的目的是通过变异个体和第 n 代个体的随机重组提高种群的多样性。标准微分进化算法的交叉公式如下

$$u_{i,j}^{(n)} = \begin{cases} v_{i,j}^{(n)}, \text{if}(\text{rand}_j[0,1] \leqslant C_R . \text{or}. j = j_{\text{rand}}) \\ x_{i,j}^{(n)}, \text{otherwise} \end{cases}$$
$$i = 1, \cdots, N_P, j = 1, \cdots, D \qquad (2)$$

式中,$u_i^{(n)} = (u_{i,1}^{(n)}, u_{i,2}^{(n)}, \cdots u_{i,D}^{(n)})$ 为交叉后产生的个体,rand_j 是[0,1] 之间的随机数;C_R 是[0,1] 之间的一个常数,在优化运行开始前给定,称为交叉常量。C_R 的取值越大,发生交叉的概率越大,$C_R = 0$ 则没有交叉的可能。j_{rand} 是在 r 之间随机取值的一个整数随机变量,保证了 $u_{i,j}^{(n)}$ 至少有一个分量是从 $v_{i,j}^{(n)}$ 中获取,以确保有新的个体生成,通过这种手段来保持群体的活跃性,维持进化不断进行。

1.3 选择算子

DE 的选择操作是一对一贪婪选择,当测试个体 $u_i^{(n)}$ 的适应度值比 $x_i^{(n)}$ 的适应度值更好时,$u_i^{(n)}$ 会被接受成为下一代种群的一员;反之,$x_i^{(n)}$ 则会被保留。假设待求优化问题为求最小值,则选择操作如下所示

$$x_i^{(n+1)} = \begin{cases} u_i^{(n+1)}, \text{if}(f(u_i^{(n+1)}) \leqslant f(u_i^{(n)})) \\ x_i^{(n)}, \text{otherwise} \end{cases} \qquad (3)$$

式中,$f(x_i^{(n)})$ 为 $x_i^{(n)}$ 的适应值。

2 MOAT 全局灵敏度分析方法

2.1 灵敏度分析方法

灵敏度分析方法的主要研究内容是:定性或定量地分析输入变量(或参数)的变化如何影响输出值,简单说来就是分析输入参数对输出值影响的程度大小[8]。灵敏度分析的对象可以是任意一个系统或模型,只需要输入和输出数据即可,本质上是通过系统的数学性质对系统的内在特性进行研究。灵敏度分析方法有很多种,各种方法出于不同的目的而使用了不同的构造思想和数学工具,导致这些方法各具特色。对特定的问题而言,可根据其适用条件和使用要求选取最合适的方法。

灵敏度分析方法大体上可分为局部灵敏度分析方法、全局灵敏度分析方法和基于人工神经网络的方法这几大类[9]。局部灵敏度分析方法把导数信息、梯度作为灵敏度信息,这种信息是局限于某个基准点附近的信息,包括手工求导、有限差分、自动微分、复变量方法等几种。全局灵敏度分析方法是在整个定义域内,研究设计变量对系统输出的影响,又分为筛选法、蒙特卡罗方法、基于方差的方法等几类。筛选法通常用来处理含有大量输入变量的模型,包括 OAT 方法、COTTER 方法、IFFD 方法等,计算量相对比较小,MOAT[10] 是其中较优秀的一种。蒙特卡罗方法包括散点图法、相关系数法、回归分析法,大多是使用统计学的方法,对于非线性、非单调的问题适用性较差。方差法包括重要性估计法、傅里叶幅值灵敏度测试法(FAST)等,是近年来研究发展最为迅速、应用最为广泛的一类方法。其中比较特殊的是 Sobol 方法[11],具有蒙特卡罗方法和方差法两方面的特点。关于这些方法的性能对比和分析,在若干文献[12] 里面都可以找到其他学者的研究结果。灵敏度分析已广泛应用于大气化学、气体排放、鱼群动力学、综合指标分析、金融投资、放射性废物处理、地质信息系统、固体物理学等等方向。

近年来全局灵敏度分析方法的发展和应用远远超过了局部灵敏度分析方法,而基于神经网络的方法较新但研究较少。其中 MOAT、Sobol 等几种方法的研究最为广泛,MOAT 方法的特点是适用于较高维度问题,计算量远远低于 Sobol,但精度稍低于 Sobol 方法。

2.2 MOAT 灵敏度分析方法

Morris 于 1991 年提出了一种 One-at-a-time (OAT)方法,以下简称为 MOAT 方法,在 GSA 中是一种相对高效的方法。OAT 方法的最主要特点是每次分析一个分量对输出的影响,如果需要分析所有分量的影响,那么需要多次计算。MOAT 实际上是对各种局部灵敏度分析方法的拓展,主要是在可行域内随机取值,从而在一定程度上消除了对初始点的依赖,最后通过取平均值体现了全局性。MOAT 方法可以用于识别具有多个输入变量的模型中的少数关键变量,一些学者把有这种作用的方法称为筛选法。

MOAT 首先定义了 Elementary Effect(EE),例如对于设计变量的第 i 个分量

$$EE_i(x) = [y(x_1,\cdots,x_{i-1},x_i+\Delta,x_{i+1},\cdots,x_k) - y(x)]/\Delta$$

EE 的定义式类似于偏导数,但是由于 Δ 的取值并非接近于无穷小,因此 EE 不同于偏导数。MOAT 方法中,求解 EE 时的基准点 x 可在整个定义域内随机选取,因此在一定程度上摆脱了对于 x 的依赖。假设设计变量 $\boldsymbol{x}=[x_1,x_2,\cdots x_k]$,共有 k 个分量。如果计算所有分量的影响,那么至少需要取 $k+1$ 个点 $x^1,x^2\cdots x^{k+1}$,最简单情况就是 $x^{i+1}-x^i=\Delta\cdot\boldsymbol{e}_i$,$\boldsymbol{e}_i=[0,0\cdots1\cdots0]$ 是第 i 个分量为 1 的单位向量,Δ 是 MOAT 方法选取的一个参数。

$$\boldsymbol{B}=\begin{bmatrix}0&0&0&\cdots&0\\1&0&0&\cdots&0\\1&1&0&\cdots&0\\1&1&1&\cdots&0\\\cdots&\cdots&\cdots&\cdots&\cdots\\1&1&1&\cdots&1\end{bmatrix}_{m\times k} \tag{4}$$

那么计算 $k+1$ 个函数值,两两相减然后除以 Δ 就可求得 k 个 EE。Δ 是影响 EE 的一个重要参数。MOAT 把设计变量的定义域离散化来求得 Δ。例如分量 x_i 的初始定义域是实数域,而现在使用 MOAT 只从 $\{0,1/(p-1),2/(p-1),\cdots,1\}$ 这 p 个值中随机选取某一个作为 x_i。这样一来,整个 k 维的定义域空间 Ω 被划分成 $k\times p$ 个网格点,x 只能从这些网格点中随机选取。Δ 为 $1/(p-1)$ 的倍数。而且需要保证,当 x 为区域 Ω 中的任意值,点 $x+\Delta$ 仍在 Ω 内。MOAT 的总计算量为 $r(k+1)$,每 $k+1$ 个点组成一个路径(path),重复 r 次然后取平均即得到灵敏度。

MOAT 方法的具体计算步骤如下:

实际使用中需要考虑基准点在定义域内的随机选取。随机化处理的步骤如下:

1) 令 $\boldsymbol{D}^*$ 是 k 维对角矩阵,对角元素是 1 或者 −1,概率各是 0.5。设 $\boldsymbol{J}_{m,k}$ 是 $m\times k$ 的全 1 矩阵,得到 $(1/2)[(2\boldsymbol{B}-\boldsymbol{J}_{m,k})\boldsymbol{D}^*+\boldsymbol{J}_{m,k}]$。

2) 设 x^* 是随机选择的基准值,每一个分量都是从 $\{0,1/(p-1),2/(p-1),\cdots,1-\Delta\}$ 中随机等概率地选择一个值。

3) 设 $\boldsymbol{P}^*$ 是一个 k 维随机扰动矩阵,每一列包含一个 1,其他所有元素都是 0,而且任意 2 列的 1 不在同一行。

其中 $\boldsymbol{D}^*$、$\boldsymbol{x}^*$、$\boldsymbol{P}^*$ 是各自独立地随机生成。最后就得到了

$$\boldsymbol{B}^*=\{\boldsymbol{J}_{m,1}x^*+(\Delta/2)[(2\boldsymbol{B}-\boldsymbol{J}_{m,k})\boldsymbol{D}^*+\boldsymbol{D}^*]\}\boldsymbol{P}^*$$

$\boldsymbol{B}^*$ 实质上就是前述矩阵 $\boldsymbol{B}$ 的随机形式。如果重复 r 次,就得到了 $[\boldsymbol{B}_1^*\quad\boldsymbol{B}_2^*\quad\cdots\boldsymbol{B}_r^*]^{\mathrm{T}}$。计算时,对 r 次的结果进行平均,这就是 MOAT 定义的灵敏度信息。

$$EE_i=\frac{1}{r}\sum_{m=1}^{r}\left(\frac{f(\boldsymbol{B}_{m,i+1}^*)-f(\boldsymbol{B}_{m,i}^*)}{\Delta x_i}\right) \tag{5}$$

式中,$f(\boldsymbol{B}_{m,i}^*)$ 是第 m 个 $\boldsymbol{B}^*$ 的第 i 行。

2.3 MOAT 方法与 Sobol 方法对比分析

本小节对 MOAT 与 Sobol 2 种方法进行对比,使用这 2 种方法对测试函数进行灵敏度计算并对比二者在计算效率、计算精度方面的问题。由于 MOAT 方法和 Sobol 方法的原理不同,这 2 种方法使用了 2 种不同的数学方式来度量灵敏度信息。但是 Campolongo 发现,MOAT 方法定义的 Elementary Effects 可以作为对 Sobol 方法定义的 S_{ti}(total sensitivity indice)一个较好的近似,理想情况下二者会逼近于同样一个期望,因此具有一定的可比性[13]。

此处给出的是 Sobol gstar function 的测试结果对比,这个函数是 Sobol 和很多学者在灵敏度研究时经常使用的测试函数中的一个。gstar 函数中的系数采用 $\boldsymbol{a}=[0,0.1,0.2,0.3,0.4,0.8,1,2,3,4]$。

由于 MOAT 和 Sobol 方法都依赖于样本进行计算,因此样本数量的影响十分重要。在具体操作过程中,应逐渐提高样本数量直至计算收敛,结果才是可信的。MOAT 设置参数为:level = 4,重复 10 次,总计 110 个样本。Sobol 方法使用拉丁超立方取样,样本总数 1 200 个。计算结果如图 1~图 2 所示,横坐标是设计变量的序号,纵坐标分别是 MOAT 和

Sobol 定义的灵敏度值。

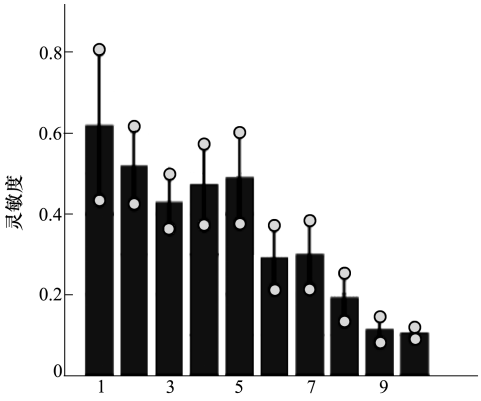


图 1 gstar 函数使用 MOAT 计算灵敏度结果

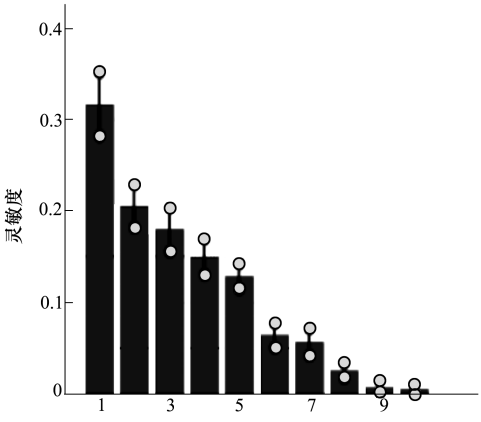


图 2 gstar 函数使用 Sobol 计算灵敏度结果

从这 2 种方法的计算结果可以看出,MOAT 的 Elementary Effects 和 Sobol 的 Total Sensitivity Indice 有近似相同的趋势,但是具体数值有所差异。g 函数的结果中对 x_2 和 x_4 的灵敏度计算差别较大。gstar 函数的结果 MOAT 不如 Sobol 的结果精确。从参数设置来看,x_1 至 x_{10}的权重系数依次递减,那么灵敏度应该也是依次递减的,Sobol 正确体现出了这种趋势。MOAT 由于样本生成方法的影响,在每个变量的区间中取定若干点,而不是完全随机取值,这样在灵敏度计算时,设计变量的每一维的取值比较固定。可能就是这种原因导致了 MOAT 的灵敏度精度比 Sobol 稍差一些。另外,图中显示的圆圈表示求解灵敏度方差。

测试算例表明:MOAT 方法的理论基础弱于定量分析,但优点是需要样本数量少。Sobol 方法理论较严密,但计算量需求极大。考虑到本研究的应用背景是针对计算量受到限制的问题,Sobol 方法实用性不强,而且对于改进 DE 而言,对灵敏度的精度要求并非很高,因此选择 MOAT 方法来展开后续研究。

3 改进微分进化算法

针对某些实际工程问题,需要在给定计算量约束的情况下,尽量获取更好的优化结果。这也就限制了种群的规模和迭代次数,对优化算法的性能提出了更苛刻的要求。

本文研究提出了 2 种改进思路,分别对微分进化算法的交叉和变异公式进行修改,主要考虑灵敏度信息是对设计变量的重要程度的一种度量,灵敏度值越大,表明对输出的影响越大,那么在算法迭代过程中,可以考虑将灵敏度信息耦合到变异算子或者交叉算子中去。本质上,这是通过灵敏度信息来控制交叉或者变异算子,其思路与自适应 DE 的一些方法类似。

第一种改进方式是修改交叉算子,改进后的 DE 以下简称为 GSADE1。根据灵敏度计算的结果,可以据此修改每个分量的交叉概率。具体操作方式为:将 C_R 扩展为矢量,$C_R = [C_{R1}, \cdots C_{RD}]$,其维度和设计变量的维度相同,$C_R$ 的每个分量的大小使用灵敏度信息来控制。设灵敏度矢量 $\boldsymbol{S} = [S_1, \cdots, S_i, \cdots S_D]$,其中灵敏度的最小值和最大值分别是 $\boldsymbol{S}_{\min} = \min(\boldsymbol{S})$、$\boldsymbol{S}_{\max} = \max(\boldsymbol{S})$,那么可以得到 C_R 的计算公式如下

$$C_{Ri} = (S_i - S_{\min})/(S_{\max} - S_{\min}) * \alpha + \beta \quad (6)$$

式中,α 和 β 相加等于 1,例如 $\alpha = 0.4$、$\beta = 0.6$,另外假设灵敏度的值是大于零的,那么 C_{Ri} 最终是在 [0.6,1] 之间变化,即式中的 S_i 首先被归一化,然后被缩放到指定的区间内。最后,微分进化算法的交叉公式如下,基本形式保持不变。考虑到 MOAT 方法计算的精度不是很高的问题,通过设置参数能保证交叉算子的值位于一定区间内,即使灵敏度精度不足,也能保证一定量的交叉概率,减弱了因灵敏度计算精度不足的影响。改进变异算子的思路也是同理。

$$u_{i,j}^{(n)} = \begin{cases} v_{i,j}^{(n)}, \text{if}(\text{rand}_j[0,1] \leqslant C_R(j).\text{or}.j = j_{\text{rand}}) \\ x_{i,j}^{(n)}, \text{otherwise} \end{cases}$$
$$i = 1, \cdots, N_P, j = 1, \cdots, D \quad (7)$$

第二种改进方式是修改变异算子,改进后的 DE 以下简称为 GSADE2。本文使用的变异算子形式如下,从中可以看出主要影响因素就是个体的选取和放缩因子 $\boldsymbol{F}$ 的影响。对 $\boldsymbol{F}$ 的修改方式与 C_R 类似,将 F 扩展为矢量,$\boldsymbol{F}=[F_1,\cdots F_D]$,其维度和设计变量的维度相同,$\boldsymbol{F}$ 的每个分量的大小使用灵敏度信息来控制。设灵敏度矢量 $\boldsymbol{S}=[S_1,\cdots,S_i,\cdots S_D]$,其中灵敏度的最小值和最大值分别是 $S_{\min}=\min(\boldsymbol{S})$、$S_{\max}=\max(\boldsymbol{S})$,那么可以得到 $\boldsymbol{F}$ 的计算公式如下

$$F_i=(S_i-S_{\min})/(S_{\max}-S_{\min})*\lambda+\omega \quad (8)$$

例如 $\lambda=0.3$、$\omega=0.7$,另外优于灵敏度的值是大于零的,那么 F_i 最终是在[0.7,1]之间变化,即式中的 S_i 被变换到指定的区间内。最终得到改进后的变异公式如下

$$v_i^{(n+1)}=x_{\text{best}}^{(n)}+F(i)\cdot(x_{r1}^{(n)}+x_{r2}^{(n)}-x_{r3}^{(n)}-x_{r4}^{(n)}) \quad (9)$$

4　函数测试及分析

本节使用 5 个解析函数作为测试问题,采用基本 DE 和改进算法分别寻优,以验证改进算法的性能。使用 2005 年 IEEE 进化计算大会的标准测试函数[14] 中的中的 Shifted Schwefel′s、Shifted Rotated High Conditioned Elliptic Function、Shifted Rosenbrock′s Function、Shifted Rotated Rastrigin′s Function、Shifted Rotated Expanded Scaffer′s,仿照该文献中的编号依次称为函数 2、3、6、10、14,设计变量范围和函数中的参数均使用文献中所给出的值。其中,函数 2 是单峰值函数,函数 3 是高条件数的单峰值函数,函数 6 是多峰值函数,函数 10 和函数 14 具有旋转性和极多局部峰值。这几个函数包含了单峰值、多峰值、带旋转等多种特性,因此具有一定的代表性。这几个函数的维度可以人为指定,考虑到本文面向的工程问题的维度在 50 维左右,因此针对 50 维的情况来进行测试。

各函数均使用 MOAT 方法计算灵敏度信息,样本总数 510 个,如果收敛较快则减少样本数目。

初始 DE 参数设置使用 DE/BEST/2/BIN 变异算子,放缩因子 $F=0.5$,交叉因子 $C_R=0.9$。GSADE1 参数设置是 $\alpha=0.1$、$\beta=0.9$、放缩因子 $F=0.5$。GSADE2 参数设置是 $\lambda=0.2$、$\omega=0.5$、交叉因子 $C_R=0.9$。除了以上区别,改进算法的其他参数都与初始 DE 相同。种群大小统一设定为 50 和 100,限定进化 50 代。

对每个函数重复测试 20 次,将收敛历史求和取平均,得到平均的收敛历史。收敛的最终结果的 20 次平均值如表 1 所示。

表 1　收敛平均值对比

标准测试函数	种群 50 收敛平均值			种群 100 收敛平均值		
	初始	改进 1	改进 2	初始	改进 1	改进 2
*F*2	1 188.65	369.95	1 124.80	390.03	−235.48	−194.02
*F*3	2 560.44	2 895.55	5 955.27	3 362.55	2 126.04	2 852.10
*F*6	478.27	390.85	444.53	422.99	393.57	411.60
*F*10	−328.41	−327.59	−328.73	−329.53	−329.62	−329.69
*F*14	−294.21	−294.66	−295.90	−295.95	−297.13	−297.58

图 3~图 7 显示了 3 个较复杂测试函数的最后 20 步平均收敛曲线。

种群数 100 的优化结果显示:2 种改进算法完全优于初始 DE。考虑到计算灵敏度信息使用的样本数量,GSADE1 和 GSADE2 的收敛速度也比基本 DE 更快。在函数 2、函数 6、函数 14 中改进算法优势明显。而在函数 3、函数 10 算例中,改进算法在优化前期优势不明显,但是在算法后期优于 DE,最终 GSADE1 和 GSADE2 的收敛结果都优于 DE。

在种群数 50 的情况下,GSADE1 只在函数 2、函数 6、函数 14 的测试中优于 DE。主要原因可能是在种群 50 严重不足地情况下,由于灵敏度对交叉概率的影响,导致在函数 3、函数 10 的测试中后期开发能力弱化。虽然在早期搜索中收敛较快,但是在算法后期开发能力不足。GSADE2 优于 GSADE1,只有函数 3 的测试中表现差于 DE。这个算例表明

交叉算子和变异算子对开发能力的影响有所不同。

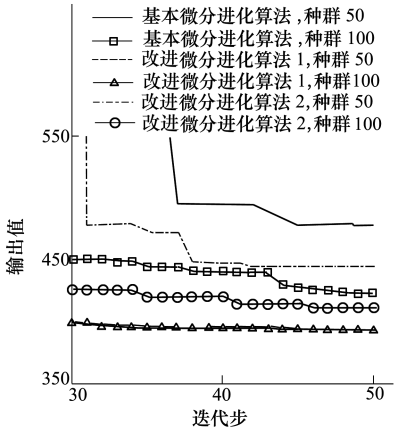


图 3 函数 6 收敛结果

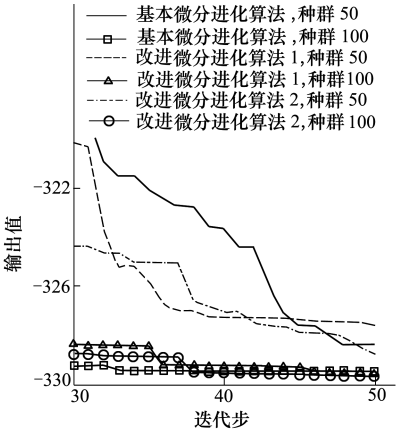


图 4 函数 10 收敛结果

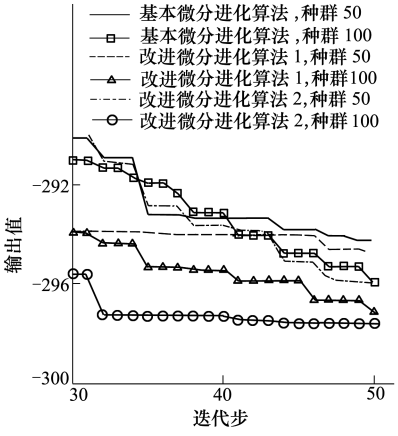


图 5 函数 14 收敛结果

5 结 论

本文提出 2 种基于全局灵敏度分析的改进 DE 算法 GSADE1 和 GSADE2 并进行了算例验证,得出以下结论:

1) MOAT 方法计算效率高,精度稍低于 Sobol 方法,将之与优化算法结合对于改善优化搜索结果、提高优化效率具有积极作用。

2) 2 种改进 DE 算法在种群数等于 100 的情况下完全优于初始 DE,而在种群数 50 的情况下 GSADE2 优于 GSADE1,但是仍有所不足,需要进一步针对极小种群的情况进行研究。

参考文献:

[1] Storn R, Price K. Differential Evolution-a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces[J]. Journal of Global Optimization, 1997, 11(4): 341-359

[2] 苏海军,杨煜普,王宇嘉. 微分进化算法的研究综述[J]. 系统工程与电子技术, 2008, 30(9): 1793-1797
Su Haijun, Yang Yupu, Wang Yujia. Research on Differential Evolution Algorithm: a Survey[J]. Systems Engineering and Electronic, 2008, 30(9): 1793-1797 (in Chinese)

[3] 刘波,王凌,金以慧. 差分进化算法研究进展[J]. 控制与决策, 2007, 22(7): 721-729
Liu Bo, Wang Ling, Jin Yihui. Advances in Differential Evolution[J]. Control and Decision, 2007, 22(7): 721-729 (in Chinese)

[4] 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

[5] 赵光权. 基于贪婪策略的微分进化算法及其应用研究[D]. 哈尔滨:哈尔滨工业大学, 2007
Zhao Guangquan. Differential Evolution Algorithm with Greedy Strategy and Its Applications[D]. Harbin, Harbin Institute of Technology, 2007 (in Chinese)

[6] Fan H Y, Lampinen J. A Trigonometric Mutation Operation to Differential Evolution[J]. Journal of Global Optimization, 2003, 27(1): 105-129

[7] Wang F S, Jing C H, Tsao G T. Fuzzy-Decision-Making Problems of Fuel Ethanol Production Using a Genetically Engineered Yeast[J]. Industrial & Engineering Chemistry Research, 1998, 37(8): 3434-3443

[8] Saltelli A, Ratto M, Andres T, et al. Global Sensitivity Analysis[M]. The Primer, John Wiley and Sons, 2008

[9] 朱亚涛. 基于敏度分析的飞行器气动/隐身综合优化设计策略研究[D]. 上海:上海交通大学, 2010
Zhu Yatao. The Strategy Investigation on Aircraft Aerodynamic——Stealth Composite Optimal Design Based on Sensitivity Analy-

sis[D]. Shanghai, Shanghai Jiaotong University, 2010 (in Chinese)
[10] Morris M D. Factorial Sampling Plans for Preliminary Computational Experiments[J]. Technometrics, 1991, 33: 161-174
[11] Sobol′ I M. Sensitivity Estimates for Nonlinear Mathematical Models[J]. Mat Model, 1990, 2(1): 112-118
[12] Gan Y, Duan Q, Gong W, et al. A Comprehensive Evaluation of Various Sensitivity Analysis Methods: A Case Study with a Hydrological Model[J]. Environmental Modelling & Software, 2014, 51: 269-285
[13] Campolongo F, Cariboni J, Saltelli A. An Effective Screening Design for Sensitivity Analysis of Large Models[J]. Environmental Modelling & Software, 2007, 22(10): 1509-1518
[14] Suganthan P N, Hansen N, Liang J J, et al. Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization[J]. Nanyang Technological University, 2005(1): 341-357

A Global Sensitivity Analysis Enhanced Differential Evolution Algorithm

He Xiaolong, Bai Junqiang, Li Yufei
(College of Aeronautics, Northwestern Polytechnical University, Xi′an 710072, China)

Abstract: An improved differential evolution (DE) algorithm based on global sensitivity analysis is proposed to enhance performance in high dimension problems with limited computation resources. Morris-One-at-a-Time(MOAT) method was firstly tested on a typical function and compared with Sobol sensitivity method, showing high efficiency and acceptable result. Then MOAT method is used to calculate sensitivity for each dimension of input vector, and new crossover and mutation operators are proposed to incorporate sensitivity for two improved algorithm GSADE1 and GSADE2. Five 50-dimension functions were used for test, showing both two new algorithms are better than the original DE.

Keywords: algorithm; global optimization; differential evolution; computer simulation; sensitivity analysis