二阶段循环优化差分演化算法

周雅兰[1],王甲海[2],林 琛[2]
(1.广东商学院信息学院,广东广州 510320;2.中山大学计算机科学系,广东广州 510006)

摘 要: 差分演化算法具有结构简单容易实现,收敛速度快和鲁棒性强等优点,但是存在早熟和进化停滞的现象.提出的二阶段循环优化差分演化算法框架能够很好地保持算法局部开采能力和全局勘探能力的平衡.在差分演化的变异操作中,以马氏距离矩阵为依据分别在目标向量的近邻或者远邻中选择父辈个体参与变异,这样分别形成偏重局部开采或者偏重全局勘探的搜索阶段,此二阶段循环迭代,使得局部开采和全局勘探能力得到震荡平衡.在 CEC2005 标准函数集上的测试结果显示了提出算法框架的有效性.

关键词: 差分演化;二阶段循环优化;局部开采;全局勘探;连续优化
中图分类号: TP18 **文献标识码:** A **文章编号:** 0372-2112 (2013) 12-2456-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2013.12.021

Framework of Recurring Two-Stage Differential Evolution

ZHOU Ya-lan[1], WANG Jia-hai[2], LIN Chen[2]
(1. *Information Science School, Guangdong University of Business Studies, Guangzhou, Guangdong* 510320, *China*;
2. *Department of Computer Science, Sun Yat-sen University, Guangzhou, Guangdong* 510006, *China*)

Abstract: The advantages of differential evolution(DE) are its simple structure, easiness of implement, fast convergence and robustness. However, DE often suffers from premature convergence and stagnation problems. A framework of the recurring two-stage DE is proposed to balance global exploration and local exploitation. The proposed framework is based on repeated and alternated execution of two different stages, namely, the local exploitation and global exploration stages. The parent individuals for the mutation operation at each stage are selected from neighbors or strangers of the target vector, respectively, based on the Mahalanobis distance matrix. The simulation results on the CEC2005 real-parameter optimization benchmark functions show that the proposed framework can make DE more efficient.

Key words: differential evolution; recurring two-stage optimization; local exploitation; global exploration; continuous optimization

1 引言

差分演化(Differential Evolution, DE)算法是一种简单有效地求解连续优化问题的算法[1],因其结构简单容易实现、收敛速度快、鲁棒性好等优点,迅速引起了国内外研究学者的关注,提出了许多改进的算法,并被广泛地应用于科学工程领域[2].DE 的改进算法大致可以分为两类[3]:一类是在 DE 中引入其他的组件,如将三角变异嵌入到 DE[4],与局部搜索和单点交叉相结合的 DE[5],免缩放因子的 DE[6];另一类是对 DE 结构进行修改,如自适应参数的 DE(Self-adapting control parameters in DE, SaDE)[7],逆向 DE(Opposition-based DE, ODE)[8],基于邻域变异算子的 DE[9],多个变异算子协作的 DE[10].

DE 算法的主要特点是利用不同父辈个体间的差异来产生下一代解,但它采用完全随机的方式来选择参与变异的父辈个体,没有利用个体间的距离信息,这限制了算法全局搜索的广度和局部搜索的深度,导致算法容易出现早熟和进化停滞现象.演化算法要取得好的性能,关键是要平衡好全局勘探能力和局部开采能力[11~13].提出的二阶段循环优化差分演化(Recurring Two-stage DE, RTDE)框架分为偏重局部开采与偏重全局勘探二阶段,二阶段分别在目标向量的近邻或远邻中选择父辈个体参与变异,搜索过程中二阶段循环迭代,局部开采和全局勘探能力在反复循环中得到震荡平衡.实验表明 RTDE 框架能够取得好的性能.

收稿日期:2012-09-27;修回日期:2013-01-18;责任编辑:覃怀银
基金项目:国家自然科学基金(No.60905038, No.60805026, No.61070076);广州市珠江科技新星;广东商学院科研创新团队建设计划

2 DE 算法

DE 算法有以下三个主要算子:变异,交叉和选择[1].

变异:假设在 D 维空间,第 g 代的解向量(也称为目标向量)为 $\boldsymbol{X}_i^g=(x_{i1}^g,\cdots,x_{id}^g,\cdots,x_{iD}^g)$, $i=1,2,\cdots,N$,变异向量为 $\boldsymbol{V}_i^g=(v_{i1}^g,\cdots,v_{id}^g,\cdots,v_{iD}^g)$,常用的变异算子有[13]:(1) rand/1: $\boldsymbol{V}_i^g=\boldsymbol{X}_{r1}^g+F\cdot(\boldsymbol{X}_{r2}^g-\boldsymbol{X}_{r3}^g)$,(2) best/1: $\boldsymbol{V}_i^g=\boldsymbol{X}_{\text{best}}^g+F\cdot(\boldsymbol{X}_{r1}^g-\boldsymbol{X}_{r2}^g)$,(3) current-to-best/1: $\boldsymbol{V}_i^g=\boldsymbol{X}_i^g+F\cdot(\boldsymbol{X}_{\text{best}}^g-\boldsymbol{X}_i^g)+F\cdot(\boldsymbol{X}_{r1}^g-\boldsymbol{X}_{r2}^g)$,其中 $\boldsymbol{X}_r$ 是完全随机选择的父辈个体, $r_1\neq r_2\neq r_3\neq i\in\{1,2,\cdots,N\}$, $\boldsymbol{X}_{\text{best}}$是最优个体,参数 $F\in(0,1]$为缩放因子.上述变异算子中的变异向量都是由缩放后的差分向量加载在一个基向量而成.

交叉:将目标向量 $\boldsymbol{X}_i^g$ 和变异向量中的维值交叉,用于产生试验向量 $\boldsymbol{T}_i^g=(t_{i1}^g,\cdots,t_{id}^g,\cdots,t_{iD}^g)$,常用的二项式交叉法如下:

$$t_{id}^g=\begin{cases}v_{id}^g, & \text{if rand}()<C_r\text{or } d=k\\ x_{id}^g, & \text{otherwise}\end{cases}\tag{1}$$

其中 $C_r\in[0,1]$是交叉率, $k\in\{1,2,\cdots,D\}$是随机选择的维.

选择:采用一对一贪婪选择算子如下:

$$\boldsymbol{X}_i^{g+1}=\begin{cases}\boldsymbol{T}_i^g, & \text{if } T_i^g \text{ is better than } \boldsymbol{X}_i^g\\ \boldsymbol{X}_i^g, & \text{otherwise}\end{cases}\tag{2}$$

3 RTDE 算法框架

DE 算法中变异算子的父辈个体选择使用均匀随机选择策略(某些变异算子直接使用最优解),没有考虑父辈种群解之间的距离信息,是一种完全随机的搜索策略,不利于算法局部开采能力和全局勘探能力的平衡.提出的 RTDE 算法框架分为偏重局部开采和偏重全局勘探二阶段,此二阶段分别在目标向量的近邻或者远邻中选择参与变异的父辈个体,目标向量的近邻或者远邻主要依据解向量距离矩阵确定.本节首先介绍解向量距离矩阵的计算,然后对二阶段循环优化策略和算法流程进行介绍和分析.

3.1 解向量距离矩阵的计算

设第 g 代种群 $\boldsymbol{P}^g=\{\boldsymbol{X}_0^g,\boldsymbol{X}_1^g,\boldsymbol{X}_2^g,\cdots,\boldsymbol{X}_{N-1}^g\}$,计算任意两解之间的距离,这些距离形成一个对角线元素为零的对称距离矩阵 $\boldsymbol{R}^g$: $\boldsymbol{R}^g=\begin{bmatrix}0 & r_{0,1}^g & r_{0,2}^g & \cdots & r_{0,N-1}^g\\ r_{1,0}^g & 0 & r_{1,2}^g & \cdots & r_{1,N-1}^g\\ r_{2,0}^g & r_{2,1}^g & 0 & \cdots & r_{2,N-1}^g\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ r_{N-1,0}^g & r_{N-1,1}^g & r_{N-1,2}^g & \cdots & 0\end{bmatrix}$,其中, $r_{i,j}^g=\|\boldsymbol{X}_i^g,\boldsymbol{X}_j^g\|$表示第 g 代种群中解 $\boldsymbol{X}_i^g,X_j^g$ 之间的距离.因为解的不同维取值范围可能不同,提出的 RTDE 使用马氏距离方法计算任意两解之间的距离[14],以避免出现偏重某些维的问题.第 g 代两解 $\boldsymbol{X}_i^g,\boldsymbol{X}_j^g$ 的马氏距离表示如下:

$$r_{i,j}^g=\sqrt{(\boldsymbol{X}_i^g-\boldsymbol{X}_j^g)^{\mathrm{T}}\boldsymbol{S}^{-1}(\boldsymbol{X}_i^g-\boldsymbol{X}_j^g)}\tag{3}$$

其中, $\boldsymbol{S}$ 为解的协方差矩阵.因为解向量的各维相互独立,所以其协方差矩阵为对角阵,则马氏距离计算公式(3)变为:

$$r_{i,j}^g=\sqrt{\sum_{d=0}^{D-1}\frac{(x_{id}^g-x_{jd}^g)^2}{s_d^2}}\tag{4}$$

其中, s_d^2 为每一维标准差的平方,即每维的方差.因为解的每一维在其取值范围上是均匀分布的随机变量,所以其方差为:

$$s_d^2=\frac{x_d^{\max}-x_d^{\min}}{2}\tag{5}$$

最终得出 $\boldsymbol{X}_i^g,\boldsymbol{X}_j^g$ 的马氏距离为:

$$r_{i,j}^g=\sqrt{2\cdot\sum_{d=0}^{D-1}\frac{(x_{id}^g-x_{jd}^g)^2}{x_d^{\max}-x_d^{\min}}}=\|\boldsymbol{X}_i^g,\boldsymbol{X}_j^g\|\tag{6}$$

3.2 二阶段循环优化策略

提出的 RTDE 算法框架分为偏重局部开采和偏重全局勘探二阶段.当变异算子中选择的父辈与目标向量距离较近时,通过变异与交叉会生成靠近目标向量的新解,即在目标向量邻域进行搜索,偏重局部开采.当选取的父辈与目标向量距离较远时,会生成离目标向量距离较远且也不会落在父辈邻域内(因为交叉算子的存在)的新解,即在目标向量较远区域进行搜索,偏重全局勘探.图 1 是假设 DE 的变异算子为 rand/1,在每一维取值范围都为 1 的二维解空间中进行搜索的示意图.图 1(a)所示是偏重局部开采阶段,变异算子选择与目标向量 $\boldsymbol{X}_i^g$ 相距较近的 $\boldsymbol{X}_{r1}^g,\boldsymbol{X}_{r2}^g,\boldsymbol{X}_{r3}^g$为父辈,生成的下一代解落于两矩形之差的范围内.图 1(b)所示是偏重全局勘探阶段,选择与目标向量 $\boldsymbol{X}_i^g$ 相距较远的解作为父辈,从图中可见下一代解产生的区域比图 1(a)面积大,而且大部分与父辈有较远的距离.

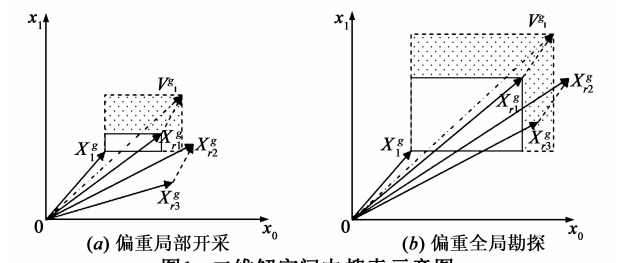

图1　二维解空间中搜索示意图

在确定父辈时,如果采用最简单的方式,即直接选定离目标向量最近或最远的数个解作为父辈,会使得在偏重局部开采阶段算法变得过于贪婪,而在偏重全局勘探阶段,易导致解跳至边界,如生成的解经过越界截断处理之后,出现多个维值同时处于边界.因此,提出的算法先根据马氏距离计算出每个父辈被选中参与变异操作的概率,然后按照轮盘赌选择方式来最后确定父辈.

在偏重局部开采阶段,对目标向量 $\boldsymbol{X}_i^g$ 进行更新时,计算父辈的选择概率公式如下:

$$p_{i,j}^g = 1 - \frac{(r_{i,j}^g)^2}{\sum_{k=0}^{N-1}(r_{i,k}^g)^2} \tag{7}$$

在偏重全局勘探阶段,对目标向量 $\boldsymbol{X}_i^g$ 进行更新时,计算父辈的选择概率公式如下:

$$p_{i,j}^g = \frac{(r_{i,j}^g)^2}{\sum_{k=0}^{N-1}(r_{i,k}^g)^2} \tag{8}$$

两个概率公式中都使用马氏距离的平方,是为了更好的放大或体现距离差异,在二阶段分别让距离更近或更远的解有更多的被选择机会.计算出选择概率后,再使用经典的轮盘赌策略来选择参与变异操作的父辈个体,以取代原始 DE 使用完全随机方式选择父辈个体的方式.对于确定性的父辈选择,如变异算子 best/1中作为基向量的最优父辈个体 $\boldsymbol{X}_{\text{best}}^g$,仍然按照原 DE 的方式,只对 $\boldsymbol{X}_{r1}^g$和 $\boldsymbol{X}_{r2}^g$依据概率采用轮盘赌选择.

提出的算法框架在搜索过程中对偏重局部开采和偏重全局勘探二阶段采用多次循环迭代的方式进行,图 2 是二阶段循环优化示意图.二阶段循环优化策略的具体实现方法是,设定两个参数 G_l 与 G_g 分别为执行偏重局部开采代数与偏重全局勘探代数,以这两个参数来控制每次执行不同阶段的长度(代数).注意在 RTDE 具体的实现中,DE 的选择过程采用同步更新策略,即所有的试验向量产生后,再对实验向量和目标向量执行一对一选择操作,整个种群同步更新.种群全部更新后,再对距离矩阵进行更新.

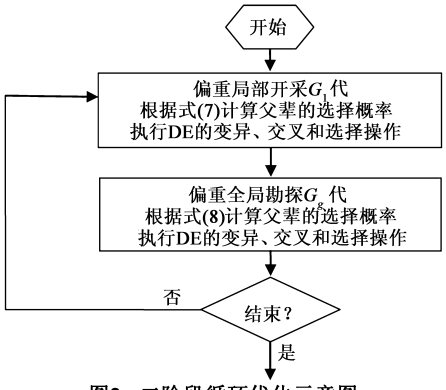

图2 二阶段循环优化示意图

如前所述,演化算法要取得比较好的性能,关键是要平衡全局勘探和局部开采能力[11~13].怎样平衡两者,文献提出了多种方法[11],经典的思想和方法是类似模拟退火的方法,即搜索初期着重全局勘探,随着搜索的进行(或者温度的降低),渐进地过渡到局部开采,此类全局勘探和局部开采之间的平衡是渐进的此消彼长,没有明显的分阶段,全局搜索能力随着搜索的进行单调下降.其缺点是搜索后期算法一旦陷入局部极值,就难以脱出局部极值,导致搜索停滞.提出的 RTDE 框架明确的分为偏重局部开采与偏重全局勘探二阶段,搜索过程中二阶段循环迭代优化,即全局-局部-全局-局部……,这样,全局勘探和局部开采的平衡是分阶段且反复循环,算法的全局搜索能力不是单调下降,而是循环震荡,使得局部开采能力和全局勘探能力在反复循环中得到震荡平衡.这样,即使算法在某一个局部搜索阶段陷入局部极小,后续的全局搜索阶段仍然有可能使得算法脱离局部极值,继续进行搜索.因此,避免了演化算法中难以同时兼顾两者的问题[11,12],有效地达到了局部开采能力和全局勘探能力的平衡[11,12].其它研究成果也表明这种平衡方式的有效性,具体可参考最近的综述[11]和二阶段演化规划方法[15].本文主要工作是用这种思想提升 DE 的性能,具体实现是用马氏距离矩阵为依据分别在目标向量的近邻和远邻中选择父辈个体参与 DE 变异操作.

演化算法解决问题的时间开销主要由算法基本操作和适应度评估两部分组成.假设适应度值的计算时间复杂度为 $O(P)$,则总的种群的适应度评估复杂度为 $O(G_{\max}\cdot N\cdot P)$,其中 $G_{\max}$为算法运行的最大代数.原始 DE 的计算复杂度为算法三个基本操作(变异,交叉和选择)的时间复杂度加上总的适应度评估复杂度,即 $O(G_{\max}\cdot N\cdot D)+O(G_{\max}\cdot N\cdot P)$.相对于原始 DE,RTDE 的额外开销在于需计算距离矩阵,即需计算两两个体距离 $N\cdot(N-2)/2$次,复杂度为 $O(D\cdot N\cdot(N-2)/2)$.所以 RTDE 的复杂度为 $O(G_{\max}\cdot N\cdot D+G_{\max}\cdot D\cdot N\cdot(N-2)/2)+O(G_{\max}\cdot N\cdot P)$.当计算适应度值的复杂度低于 $O(N\cdot D)$时,RTDE 的时间复杂度要比 DE 高;当计算适应度值的复杂度高于 $O(N\cdot D)$时,RTDE 与 DE 的时间复杂度处于同一数量级.即如果适应度值的计算开销很大的话,整个算法的开销主要取决于适应度的评估,此时新算法引入的距离矩阵计算开销,相对于占统治地位的适应度值评估开销来说,非常轻微.

4 仿真实验

4.1 实验设置

为了检验算法的性能,CEC2005 函数集的 25 个函

数作为测试函数,分 4 种类型[16]:(1)f_1-f_5:单模函数;(2)f_6-f_{12}:基本多模函数;(3)$f_{13}-f_{14}$:扩展多模函数;(4)$f_{15}-f_{25}$:复杂混合函数.其维数取 $D=30$[16].算法测试采用 C++在 PC(CPU Intel Dual-Core E5400,2.70GHz,内存 2GByte)上运行.根据 CEC2005 的推荐设置[16],用适应度函数评估的最大次数(Maximum Function Evaluations:M_{FE})作为算法结束条件,所有函数 $M_{FE}=10^4\cdot D$,即本实验中 $M_{FE}=300000$.每种算法对每个测试函数进行 100 次重复实验,每次重复实验都采用相同的初始化种群,以减少初始化的影响.DE 的三个参数采用常用设置[7]:种群数目 $N=100$,缩放因子 $F=0.5$,交叉率 $C_r=0.9$.经过一些初始实验后,设定 RTDE 中二阶段搜索代数 $G_l=G_g=M_{FE}/N\cdot 100$.

4.2 与原始 DE 比较

原始 DE 和提出的 RTDE 分别采用三种典型的变异算子 rand/1、best/1 和 current-to-best/1 进行实验比较,其中算子 rand/1 侧重全局勘探,best/1 侧重局部开采,current-to-best/1 两者兼顾.表 1 的 2~4 列是原始 DE 算法和 RTDE 算法分别采用三种变异算子对 25 个测试函数运行 100 次后,在 198 自由度下双边分布的 t-检验结果(以 $p=0.05$ 为标准,+代表明显优于原始 DE,=代表与原始 DE 结果无明显差别,-代表明显差于原始 DE),表中最后一行(Total)是统计引入新框架后每种算法的+/-/=的函数个数(限于篇幅,表中省略了具体的函数值,如需可联系作者).从表 1 可以看出,对于侧重全局勘探的 rand/1 变异算子,在 25 个函数中,提出的 RTDE 在 13 个函数上取得了明显优于原始 DE 的效果.对于 f_1-f_5 这 5 个单模函数,由于 RTDE 既有加强全局勘探阶段,也有加强局部开采阶段,所以取得了较好的结果;对于 f_6-f_{12}这 7 个多模函数以及 $f_{13}-f_{14}$拓展多模函数,RTDE 的改进并不明显;但是对于 $f_{15}-f_{25}$这 10 个混合的复杂函数,RTDE 取得了明显的改进结果.这表明,即使在偏重全局勘探的 rand/1 变异算子上,提出的框架依然能加强其全局勘探能力,获得较好的结果.使用 best/1 变异策略时,RTDE 算法所影响的只是变异算子的差分向量,其基向量依然是当前最优解.从表 1 中可以看出,RTDE 算法在 19 个函数上明显优于原始 DE,且仅在 1 个函数 f_{13}上表现较差.由于原始 DE/best/1 算法利用当前最优解作为基向量,使得算法集中搜索当前最优值的邻域,全局勘探能力较弱,所以对于多模和复杂混合函数,极易陷入局部极值.提出的 RTDE 引入了全局勘探阶段,有效的增强了算法的全局勘探能力,所以在 $f_{14}-f_{25}$上取得了明显的改进效果.在以当前解为基向量的 current-to-best/1 变异算子上,RTDE 的作用效果也较明显.与 best/1 相同,新算法框架的改进主要体现在复杂混合函数上,在 16 个函数上取得了明显的改进效果.此外,在这个变异算子上,引入新框架后的结果全部不差于原始 DE 算法.这进一步表明引入新算法框架后增强了 DE 的全局勘探能力.

表 1　三种变异算子下原始 DE 引入新框架前后以及 SaDE 和 ODE 引入新框架前后的实验结果

函数	rand/1	best/1	current-to-best/1	RTSaDE	RTODE
	RTDE vs. DE	RTDE vs. DE	RTDE vs. DE	vs. SaDE	vs. ODE
f_1	=	=	=	=	=
f_2	+	+	+	−	−
f_3	−	+	+	=	−
f_4	+	+	=	−	−
f_5	+	+	+	=	=
f_6	+	+	=	+	+
f_7	=	=	=	+	=
f_8	=	+	+	+	+
f_9	+	=	+	=	=
f_{10}	−	=	=	=	+
f_{11}	+	+	=	−	=
f_{12}	=	=	+	+	=
f_{13}	+	−	=	+	+
f_{14}	=	+	+	+	+
f_{15}	+	+	+	+	=
f_{16}	+	+	=	=	+
f_{17}	=	+	=	=	+
f_{18}	+	+	+	+	=
f_{19}	=	+	+	+	=
f_{20}	+	+	+	+	+
f_{21}	=	+	+	=	=
f_{22}	+	+	+	+	+
f_{23}	+	+	+	+	+
f_{24}	=	+	+	=	=
f_{25}	=	+	+	=	=
Total	13/10/2	19/5/1	16/9/0	12/10/3	10/12/3

为了更好的观察新框架 RTDE 对于 DE 算法的影响,选取以 rand/1 为变异算子时 4 个不同类型函数(f_4,f_9,f_{13},f_{23})的收敛过程作为代表来进行分析(限于篇幅省略了该图,如需请联系作者).对于单模函数 f_4,两种算法的收敛过程比较相似,只是 RTDE 收敛得更快而且结果更优.对于基础多模函数 f_9,在搜索起始阶段,两种算法的收敛速度都十分快,呈迅速下降趋势,之后,原始 DE 算法的收敛速度变慢,而 RTDE 仍然保持着较

快的收敛速度,取得了更好的结果.从扩展多模函数 f_{13} 的收敛过程可以看出,DE 算法在开始阶段出现了急速的收敛,而 RTDE 开始收敛相对较慢,此后一段时间收敛比 DE 算法快,在最后阶段,两种算法都处于一个几乎停滞的状态,最终 RTDE 的结果比 DE 稍好.混合复杂函数 f_{23} 极易使算法陷入局部最优,所以 DE 前期收敛速度较快,但很快就进入了停滞状态,在后期的搜索过程中解几乎没有提升,而 RTDE 虽然收敛速度相对较慢但是能够持续进化提升解的质量,最终获得比原始 DE 更优的解.

以上结果表明 RTDE 框架对原 DE 算法的主要影响是,增强了其在优化复杂目标函数时的全局勘探能力,而在此同时也保持了原 DE 算法的局部开采能力.所以在优化越复杂的函数时,体现出来的差异越大.

在实验中,RTDE 与原始 DE 在 25 个 CEC2005 函数上的平均计算时间的比值等于 1.13,说明新算法 RTDE 在测试集上没有比原始 DE 增加太多的计算开销.

4.3 与改进 DE 比较

为了进一步验证所提框架的性能,本文选取另外两个典型的改进 DE 算法—SaDE 算法[7]和 ODE 算法[8]作为代表进行实验比较.新算法框架可以直接应用于这些改进算法而不需要做太多修改.表 1 的 5 – 6 列是引入了新框架的 SaDE 和 ODE 与原算法对 25 个测试函数运行 100 次后得到的 t-检验结果.从表中可以看出在 SaDE 和 ODE 中引入 RTDE 算法框架后,对于多模和复杂混合函数的优化结果普遍都有所改进.而对于单模函数,加入新框架后的算法改进效果不明显.这是因为,对于简单的单模函数,SaDE 和 ODE 作为改进的 DE 算法本来就能够取得较好的优化效果,继续提升的空间有限,RTDE 框架难以取得更优结果.而较复杂的多模和复杂混合函数,极易使算法陷入局部极值,由于 RTDE 框架具有更好的平衡和兼顾局部和全局搜索能力的机制,使得在这些复杂函数上取得了更优的效果.总体来看,引入新框架后的算法比原算法优秀,在 25 个测试函数上分别取得了 12 优,3 差(其余 10 相同),和 10 优,3 差(其余 12 相同)的优化结果.

5 结论

原始 DE 中变异算子的父辈选择没有利用解之间的距离信息,不能很好的保持算法局部开采和全局勘探能力的平衡,本文提出了 RTDE 算法框架.该框架分为偏重局部开采和偏重全局勘探二阶段优化,二阶段分别在目标向量的近邻和远邻选择变异算子的父辈,解的近邻和远邻以马氏距离矩阵为判断依据,此二阶段循环迭代,使得全局勘探和局部开采能力得到震荡平衡.将提出的算法框架引入到原始 DE 和两个典型的改进 DE 中,在 CEC2005 测试函数上的实验结果表明提出的算法框架能够很好地提升它们的性能.

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] DAS S, SUGANTHAN P N. Differential evolution: A survey of the state-of-the-art [J]. IEEE Transactions on Evolutionary Computation, 2011, 15(1): 4 – 31.

[3] NERI F, TITTONEN V. Recent advances in differential evolution: A survey and experimental analysis [J]. Artificial Intelligence Review, 2011, 33(1 – 2): 61 – 106.

[4] ANGIRA R, SANTOSH A. A modified trigonometric differential evolution algorithm for optimization of dynamic systems [A]. IEEE World Congress on Computation Intelligence [C]. Piscataway: IEEE Press, 2008. 1463 – 1468.

[5] NOMAN N, IBA H. Accelerating differential evolution using an adaptive local search [J]. IEEE Transactions on Evolutionary Computation, 2008, 12(1): 107 – 125.

[6] 张晓伟,刘三阳.免比例因子 F 的差分进化算法[J].电子学报,2009,39(6):1318 – 1323.
ZHANG Xiao-wei, LIU San-yang. Differential evolution without the scale factor F [J]. Acta Electronica Sinica, 2009, 39(6): 1318 – 1323. (in Chinese)

[7] BREST J, GREINER S, BOŠKOVIČB, MERNIK M, ŽUMER V. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems [J]. IEEE Transactions on Evolutionary Computation, 2006, 10(6): 646 – 657.

[8] RAHNAMAYAN S, TIZHOOSH H R, SALAMA M M A. Opposition-based differential evolution [J]. IEEE Transactions on Evolutionary Computation, 2008, 12(1): 64 – 79.

[9] DAS S, ABRAHAM A, CHAKRABORTY U K, KONAR A. Differential evolution with a neighborhood-based mutation operator [J]. IEEE Transactions on Evolutionary Computation, 2009, 13(3): 526 – 553.

[10] 贺毅朝,王熙照,刘坤起,王彦祺.差分演化的收敛性分析与算法改进[J].软件学报,2010,21(5):875 – 885.
HE Yi-chao, WANG Xi-zhao, LIU Kun-qi, WANG Yan-qi. Convergent analysis and algorithmic improvement of differential evolution [J]. Journal of Software, 2010, 21(5): 875 – 885. (in Chinese)

[11] CREPINSEK M, LIU S H, MERNIK M. Exploration and exploitation in evolutionary algorithms: a survey [J]. ACM Computing Surveys, 2013, 1(1): Article A: 1 – 33.

[12] XIN B, CHEN J, ZHANG J, FANG H, PENG Z. Hybridizing differential evolution and particle swarm optimization to design

powerful optimizers:A review and taxonomy[J]. IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews, 2012, 42(5): 744 – 767.

[13] EPITROPAKIS M G, TASOULIS D K, PAVLIDIS N G, PLAGIANAKOS V P, VRAHATIS M N. Enhancing differential evolution utilizing proximity-based mutation operators[J]. IEEE Transactions on Evolutionary Computation, 2011, 15(1): 99 – 119.

[14] MAESSCHALCK R De, JOUAN-RIMBAUD D, MASSART D L. The mahalanobis distance[J]. Chemometrics and Intelligent Laboratory Systems, 2000, 50(1): 1 – 18.

[15] ALAM M S, ISLAM M M, YAO X, MURASE K. Recurring two-stage evolutionary programming: a novel approach for numeric optimization[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2011, 41(5): 1352 – 1365.

[16] SUGANTHAN P, HANSEN N, LIANG J, DEB K, CHEN Y P, AUGER A, TIWARI S. Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization[EB/OL]. http://bschw.googlecode.com/svn-history/r623/trunk/EvalCompu/Tech-Report-May-30-05.pdf, 2005-05-01/2012-06-01.

作者简介

周雅兰　女,1979 年 3 月出生于湖南省常德市.现为广东商学院信息学院副教授.主要研究方向为人工智能与数据挖掘.
E-mail: zhouylan@163.com

王甲海　男,1977 年 6 月出生于江西省赣州市.现为中山大学计算机系副教授,从事人工智能及其应用的研究工作.
E-mail: wangjiah@mail.sysu.edu.cn

林　琛　男,1986 年出生于广东汕头.研究生.主要研究方向为智能优化算法及其应用.