文章编号: 1001-0920(2017)07-1313-06 DOI: 10.13195/j.kzyjc.2016.0697

基于共轭增强策略的差分进化算法

张贵军†, 王柳静, 周晓根, 丁 情

(浙江工业大学 信息工程学院，杭州 310023)

摘 要: 为了平衡差分进化算法的全局探测能力和局部搜索能力,提出基于共轭增强策略的差分进化算法. 首先,根据个体适应度信息设计基于轮盘赌的个体选择策略,选取适应值较差的个体组建子种群;然后,基于个体的时间和空间知识设计共轭增强方向,在不丧失全局探测能力的前提下实现子种群的局部增强,以提高算法的局部搜索能力;最后,18个标准测试函数的实验结果表明,所提算法在计算代价、可靠性及收敛速度方面均优于所介绍的主流改进差分进化算法和非差分进化算法.

关键词: 差分进化；全局优化；共轭增强；子种群

中图分类号: TP391 **文献标志码:** A

Differential evolution algorithm with conjugate enhancement strategy

ZHANG Gui-jun†, WANG Liu-jing, ZHOU Xiao-gen, DING Qing

(College of Information Engineering，Zhejiang University of Technology，Hangzhou 310023，China)

Abstract: To balance the global exploration and exploitation ability of differential evolution, a differential evolution algorithm based on the conjugate enhancement strategy is proposed. Firstly, a roulette-based individual selection strategy is designed according to the fitness of individual to select individuals with the poorer fitness value to form the subpopulation. Then, the temporal and spatial knowledge of the individual is used to design the conjugate enhancement direction, and the local enhancement of subpopulation is realized without losing the global exploration ability to improve the exploitation ability. Finally, experimental results of 18 benchmark functions show that the proposed algorithm is superior to the main-stream differential evolution variants and non-differential evolution algorithms mentioned in this paper in terms of computational cost, reliability and convergence speed.

Keywords: differential evolution；global optimization；conjugate enhancement；subpopulation

0 引 言

工程实践和科学研究等诸多领域中存在着大量的优化问题亟待解决,然而公认理论严谨的传统优化方法在这些领域的优化表现并不理想,原因在于实际的工程问题往往缺乏传统优化方法所需要的连续、可微等信息,且传统优化方法面对这类多峰值、强非线性、动态变化的实际问题时,并不具备全局优化能力,存在着难以克服的局限性[1]. 因此,智能优化方法这种通过模拟自然生态机制并能自适应调节的概率搜索算法为解决复杂优化问题提供了新的研究方向,现已广泛应用于金融、电力系统、生物信息等领域[2-4].

进化算法(EAs)本质上是一类建立在模拟生物进化基础上的概率随机搜索算法,其基本思想是通过变异等操作生成适应性更好的新种群,以此寻求全局最优解[5]. Storn等[6]所提出的差分进化算法(DE)根据种群个体间的差分信息进行变异,以增加种群多样性,并且能够记录当前的搜索情况,从而动态调整搜索策略,鲁棒搜索能力较强. DE算法具有自组织性、自适应性、自学习、并行性和全局探测能力较强的优势[7],然而还存在诸如局部收敛能力弱、后期算法效率下降等问题,收敛速度较慢尤其是制约算法搜索性能的关键因素[8].

为了提高DE算法的性能,Wang等[9]提出了基于

收稿日期: 2016-06-02；**修回日期:** 2016-09-23.

基金项目: 国家自然科学基金项目(61075062, 61573317, 61379020)；浙江省自然科学基金青年基金项目(LQ16E080012)；浙江省重中之重学科开放基金项目(20151008, 20151015)；浙江省大学生科技创新活动计划(新苗人才计划)项目(2016R403083).

作者简介: 张贵军(1974—), 男, 教授, 博士生导师, 从事智能信息处理、优化理论及算法设计、生物信息学等研究；王柳静(1993—), 女, 硕士生, 从事智能优化的研究.

†通讯作者. E-mail: zgj@zjut.edu.cn

混合个体及参数生成策略的差分进化算法(CoDE),通过对设置的策略池和参数池进行随机选择组合并以竞争方式生成新个体;Qin等[10]提出了自适应差分进化算法(SaDE),设置变异策略池及参数的均匀分布策略,并根据产生优质解的历史记录自适应调整进行变异策略和参数;Zhang等[11-13]提出了基于抽象凸下界估计的DE算法,通过对种群建立局部抽象凸下界松弛模型,并设计基于支撑面的下降方向作局部增强,以加快算法的收敛速度.

为了平衡差分进化算法的全局探测能力与局部搜索能力,本文提出一种基于共轭增强策略的差分进化算法(DECE). 首先,通过设计基于适应度信息的轮盘赌个体选择策略,从种群中挑选适应值较差的个体组建子种群;然后,利用个体的时间和空间知识设计共轭增强方向对所组建的子种群作局部增强,以加快算法向全局最优解收敛的速度,从而在全局探测能力与快速收敛能力之间取得平衡.

1 DECE算法

DECE算法的思想如图1所示. 一方面,DECE算法的进化过程以个体适应值为参照,依据传统进化算子生成新个体,并根据优胜劣汰机制更新种群,以保留算法的全局探测能力;另一方面,根据适应度排名信息从原种群中选择个体组建子种群,并根据个体的时间和空间知识设计共轭增强方向,以指导子种群的局部搜索进程,从而提高算法的局部搜索能力.

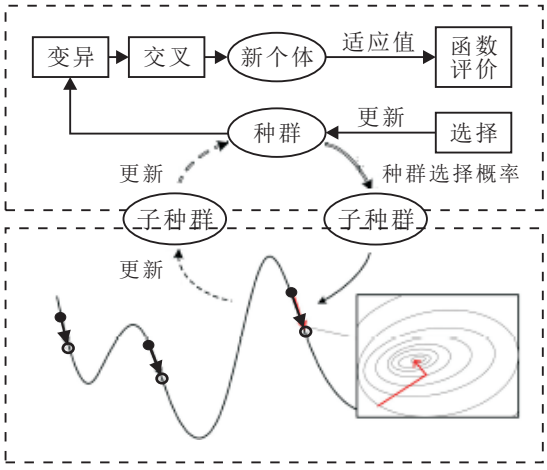


图1 算法思想示意图

1.1 个体选择策略

根据当前种群的个体适应度信息对所有个体进行升序排列,记录每个个体的排名,根据排名计算种群选择概率:

$$P^{i,g}=\frac{r^{i,g}}{0.5(\mathrm{NP}^2+\mathrm{NP})}. \tag{1}$$

其中:$r^{i,g}$表示当前第g代种群中第i个个体的适应值排名,NP表示种群规模,$P^{i,g}$表示当前第g代种群中第i个个体的种群选择概率.

式(1)体现了个体选择策略的本质是个体适应值越差则其种群选择概率越大,从而基于个体的种群选择概率利用轮盘赌方式从种群中有针对地挑选出适应值较差的个体进行局部增强. 最后,利用轮盘赌方式根据各个体的种群选择概率在每一代中选取原种群的部分个体(如NP/2个个体)进行子种群的组建.

1.2 共轭增强策略

传统DE算法的全局探测能力强,但局部搜索能力弱,后期收敛速度较慢. 本文基于种群进化的时间和空间知识设计子种群个体的局部下降方向,进而设计共轭增强方向指导子种群进行局部搜索,从而在保证全局探测能力的前提下,提高算法的局部搜索能力.

定义1 设$x^{c,g}$为当前子种群中的某个个体,若上一代种群中的个体$x^{Bnb,g-1}$与$x^{c,g}$的距离满足

$$d(x^{Bnb,g-1},x^{c,g})=\min\left\{d(x^{i,g-1},x^{c,g})\right\}. \tag{2}$$

其中:$i=1,2,\cdots,\mathrm{NP}$,g为代数. 则称$x^{Bnb,g-1}$为$x^{c,g}$的最近个体.

定义2 设$x^{Bnb,\,g-1}$为当前个体$x^{c,\,g}$的最近个体,如果$f(x^{Bnb,\,g-1})>f(x^{c,\,g})$,则称方向

$$d=x^{c,g}-x^{Bnb,g-1} \tag{3}$$

为当前个体$x^{c,\,g}$的局部下降方向;如果$f(x^{Bnb,\,g-1})<f(x^{c,\,g})$,则称方向

$$d=x^{Bnb,g-1}-x^{c,g} \tag{4}$$

为$x^{c,\,g}$的局部下降方向.

为了对子种群进行局部增强,首先根据定义1确定$x^{i,g}$的最近个体$x^{Bnb,g-1}$,从而根据定义2计算$x^{i,g}$的局部下降方向

$$d^{i,g}=-\left|x^{i,g}-x^{Bnb,g-1}\right|. \tag{5}$$

其中若选定的最近个体出现$f(x^{Bnb,g-1})=f(x^{c,g})$的情形,则取次最近个体按照式(5)进行局部下降方向的计算,本文个体间的距离采用欧氏距离. 进而利用时间和空间的进化知识以及局部下降方向设计共轭增强方向

$$d^{i,g+1}=\frac{f(x^{i,g})-f(x^{i,g-1})}{x^{i,g-1}-x^{i,g}}+\beta^{i,g}d^{i,g}. \tag{6}$$

其中$\beta^{i,\,g}$为组合搜索方向的系数因子,可以根据下式进行计算:

$$\beta^{i,g}=\left\|\frac{f(x^{i,g})-f(x^{i,g-1})}{(x^{i,g}-x^{i,g-1})(x^{Bnb,g-1}-x^{i,g})}\right\|^2. \tag{7}$$

最后根据式(6)所示的共轭增强方向$d^{i,\,g+1}$进行搜

索,以完成子种群个体的局部增强,方向$d^{i,g}$和$d^{i,g+1}$具有共轭特性,具体增强操作如下:

$$x^{i,g+1} = x^{i,g} + F \cdot d^{i,g+1}. \tag{8}$$

其中:$x^{i,g}$为第g代子种群中的第i个个体,F的取值与DE变异策略的步长因子相同.

1.3 复杂度分析

DECE算法与传统DE算法的差异在于引入了个体选择策略和共轭增强策略. 个体选择策略的时间复杂度为$\max\{O(\mathrm{NP}\cdot\log_2\mathrm{NP}),\ O(\mathrm{NP}),\ O(\mathrm{NP}/2)\}$,由于$\log_2\mathrm{NP} > 1$,整个个体选择策略的时间复杂度为$O(\mathrm{NP}\cdot\log_2\mathrm{NP})$;共轭增强策略的时间复杂度分别为$\max\{O(\mathrm{NP}),\ O(\mathrm{NP}\cdot\log_2\mathrm{NP})\}$、$O(\mathrm{NP}/2)$、$O(\mathrm{NP}/2)$、$O(\mathrm{NP}/2)$,所以整个共轭增强策略的时间复杂度为$O(\mathrm{NP}\cdot\log_2\mathrm{NP})$,因此,整个DECE算法的时间复杂度为$O(\mathrm{NP}\cdot\log_2\mathrm{NP})$. 可以看出,个体选择策略和共轭增强策略的引入并没有显著增加DE算法的时间复杂度.

2 数值实验与分析

为了评估本文所提DECE算法的优化性能,实验采用18个具有代表性的典型全局优化问题来保证实验结果的可靠性. 这18个问题中包含7个高维单模态问题($f_1 \sim f_7$)、7个高维多模态问题($f_8 \sim f_{14}$)及4个低维多模态问题($f_{15} \sim f_{18}$),表1列出了各测试函数的数学表达式及对应的参数.

表 1　标准测试函数

函数名	维数(N)	取值范围	全局最小值
f_1: Sphere	30	$(-100,100)^N$	0
f_2: SumSquares	30	$(-10,10)^N$	0
f_3: Schwefel2.22	30	$(-10,10)^N$	0
f_4: Exponential	30	$(-1,1)^N$	-1
f_5: Step	30	$(-100,100)^N$	0
f_6: Zakharov	30	$(-5,10)^N$	0
f_7: Rosenbrock	30	$(-30,30)^N$	0
f_8: Griewank	30	$(-600,600)^N$	0
f_9: Himmelblau	30	$(-5,5)^N$	$-78.332\,3$
f_{10}: Levy and Montalvo 1	30	$(-10,10)^N$	0
f_{11}: Levy and Montalvo 2	30	$(-5,5)^N$	0
f_{12}: Ackley	30	$(-30,30)^N$	0
f_{13}: Neumaier	30	$(-900,900)^N$	0
f_{14}: Alpine	30	$(-10,10)^N$	0
f_{15}: Cosine Mixture	4	$(-1,1)^N$	0.4
f_{16}: Kowalik	4	$(-5,5)^N$	0.000 307 48
f_{17}: Six-hump Camel-back	2	$(-5,5)^N$	$-1.031\,628\,5$
f_{18}: Branin	2	$(-5,10)^N$	0.397 89

实验选用了Qin等[10]提出的自适应差分进化算法(SaDE)、Zhang等[14]提出的带有外部存档功能的改进差分进化算法(JADE)、Wang等[9]提出的复合差分进化算法(CoDE)3种主流改进DE算法,以及Liang等[15]提出的综合学习粒子群算法(CLPSO)、Hansen等[16]提出的基于协方差矩阵的自适应进化策略算法(CMA-ES)、Garcia-Martinez等[17]提出的基于父代中心交叉操作的遗传算法(GL-25)3种先进的非DE算法进行比较.

DECE算法参数设置:步长因子$F = 0.5$,交叉概率$\mathrm{CR} = 0.5$,种群规模$\mathrm{NP} = 50$,子种群规模为25. SaDE、JADE、CoDE以及CLPSO、CMA-ES、GL-25这6种对比算法均采用原文献中的参数设置. 各算法对各测试函数独立运行30次. 实验基于Intel Core i5-4200 M 2.50 GHz, 4 GB, Windows 8.1, Visual Studio 2010环境,算法采用C++和Matlab语言编程实现.

2.1 与state-of-the-art算法比较

所提出的DECE算法与SaDE、JADE、CoDE这3个DE改进算法进行比较,这4种算法的终止条件均设置为函数评价次数$\mathrm{FES} = 2\,000N$,其中N为测试函数各自的维数. 通过函数评价次数(FES)和成功率(SR)这两个指标来衡量验证各算法在计算代价和可靠性方面的性能,规定所求得的最优值$f_{\min}(\boldsymbol{x})$达到一定精度要求时,即$|f_{\min}(\boldsymbol{x}) - \mathrm{optimum}| \leqslant 10^{-5}$,记录此时的FES和SR值,其中optimum为函数的全局最小值. 30次独立运行的平均函数评价次数和成功率结果见表2,其中最优结果加粗表示.

对比表2所列数据,在DECE算法对全部18个测试函数的优化中,有15个测试函数的FES值优于其他比较算法. CoDE对所有函数的结果均未优于其他算法,SaDE只对一个测试函数f_5的优化表现出最好性能,JADE对f_7和f_{16}的优化相对于其他算法取得了最优,DECE算法在大部分测试函数的实验中均可获得最优性能. 从表2的最后一行可以看出:由于DECE算法采用共轭增强策略,加快了算法的收敛速度,所以平均函数评价次数最低,为30 925,SaDE、JADE、CoDE的函数评价次数分别为56 139、37 265、70 211,即相对于SaDE、JADE、CoDE,本文所提DECE算法分别节省了44.9%、17.0%和55.9%的函数评价次数. 由此表明,DECE算法在计算代价方面显著优于其他算法.

在算法的成功率SR方面,SaDE对测试函数f_7和f_{13}在给定的目标函数评价次数里无法达到预设的精度,对于测试函数f_8的成功率为0.73;JADE算法对于函数f_6和f_8无法以100%的成功率求解,成功率均为0.97;CoDE对函数f_{14}在给定的目标函数评价次数里无法达到预设的精度,对f_{14}的成功率为0.93;本文所提DECE算法对所有函数均能以100%的成功率求解. 因此,从表2最后一行的平均成功率可以看出,本文所提DECE算法的成功率最高.

表 2 SaDE、JADE、CoDE和DECE算法的平均函数评价次数和成功率

Fun	N	SaDE		JADE		CoDE		DECE	
		FE	SR	FE	SR	FE	SR	FE	SR
f_1	30	20297	1.00	23557	1.00	40155	1.00	**1090**	1.00
f_2	30	18410	1.00	21600	1.00	36270	1.00	**4043**	1.00
f_3	30	24368	1.00	35787	1.00	56421	1.00	**4793**	1.00
f_4	30	10993	1.00	13377	1.00	22362	1.00	**1136**	1.00
f_5	30	**10657**	1.00	11790	1.00	20070	1.00	44577	1.00
f_6	30	137484	1.00	56107	0.97	77979	1.00	**36436**	1.00
f_7	30	NA	0.00	**97793**	1.00	236957	1.00	263707	1.00
f_8	30	21740	0.73	26893	0.97	43878	1.00	**1306**	1.00
f_9	30	22873	1.00	56503	1.00	35565	1.00	**7827**	1.00
f_{10}	30	12260	1.00	17340	1.00	26307	1.00	**11836**	1.00
f_{11}	30	12855	1.00	17373	1.00	26964	1.00	**12218**	1.00
f_{12}	30	30453	1.00	33080	1.00	56826	1.00	**13261**	1.00
f_{13}	30	NA	0.00	87030	1.00	268905	0.93	**52465**	1.00
f_{14}	30	72615	1.00	159810	1.00	NA	0.00	**40956**	1.00
f_{15}	4	3043	1.00	2597	1.00	3423	1.00	**1084**	1.00
f_{16}	4	7583	1.00	**5992**	1.00	6969	1.00	88002	1.00
f_{17}	2	2293	1.00	1998	1.00	2304	1.00	**831**	1.00
f_{18}	2	2573	1.00	2152	1.00	2445	1.00	**886**	1.00
AVE		56139	0.898	37265	0.996	70211	0.956	**30925**	**1.000**

表 3 SaDE、JADE、CoDE和DECE算法函数误差的平均值和标准偏差

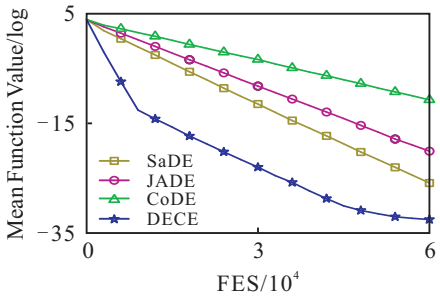
Fun	N	SaDE			JADE			CoDE			DECE	
		Mean	Std	Sig	Mean	Std	Sig	Mean	Std	Sig	Mean	Std
f_1	30	1.29e-23	3.07e-23	+	3.01e-20	8.74e-20	+	2.16e-10	1.73e-10	+	**5.04e-30**	**1.35e-29**
f_2	30	6.59e-25	8.39e-25	+	1.16e-21	1.59e-21	+	2.83e-11	2.61e-11	+	**2.93e-33**	**9.27e-33**
f_3	30	**8.70e-16**	**5.17e-16**	−	3.11e-10	3.36e-10	−	3.86e-06	1.32e-06	+	1.76e-09	1.10e-09
f_4	30	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**	≈	1.30e-14	1.13e-14	+	**0.00e+00**	**0.00e+00**
f_5	30	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**
f_6	30	9.27e-02	1.30e-01	+	6.16e-03	3.13e-02	+	5.69e-04	7.53e-04	+	**1.10e-07**	**1.91e-07**
f_7	30	5.07e+01	3.36e+01	+	**9.29e+00**	**1.06e+00**	−	1.97e+01	5.80e-01	−	2.22e+01	5.91e-01
f_8	30	2.22e-03	4.73e-03	+	9.70e-15	5.25e-14	+	2.47e-07	7.34e-07	+	**0.00e+00**	**0.00e+00**
f_9	30	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**
f_{10}	30	**2.47e-27**	**2.47e-27**	−	3.69e-21	1.55e-20	+	1.80e-13	3.36e-13	+	4.81e-16	7.33e-16
f_{11}	30	1.46e-03	3.80e-03	+	5.07e-22	2.22e-21	+	1.31e-13	1.51e-13	+	**4.81e-23**	**6.11e-23**
f_{12}	30	2.40e-01	4.45e-01	+	4.19e-11	4.49e-11	+	3.75e-06	1.88e-06	+	**3.99e-15**	**8.32e-31**
f_{13}	30	2.40e+03	8.58e+02	+	7.94e+00	1.99e+01	+	6.92e+02	8.00e+02	+	**5.24e-06**	**7.82e-06**
f_{14}	30	7.45e-05	4.86e-05	+	9.94e-03	3.50e-03	+	1.75e+00	1.47e+00	+	**6.12e-06**	**8.69e-06**
f_{15}	4	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**
f_{16}	4	9.38e-06	1.95e-05	+	1.37e-03	5.08e-03	+	9.60e-06	4.06e-05	+	**0.00e+00**	**0.00e+00**
f_{17}	2	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**
f_{18}	2	4.67e-07	1.17e-06	+	**0.00e+00**	**0.00e+00**	≈	1.33e-07	5.71e-07	+	**0.00e+00**	**0.00e+00**
+/≈/−			11/5/2			10/6/2			13/4/1			

通过计算各算法30次独立运行结果的平均值(Mean)和标准差(Std)来衡量优化结果的质量. 另外,为了验证DECE算法的优势,采用Wilcoxon Signed Rank Test[18]对优化结果进行非参数假设检验,显著性水平为0.05,结果见表3. 最优结果加粗表示,"−"表示本文算法明显差于其他算法,"+"表示本文算法相对于其他算法有显著优势,"≈"表示本文算法的优化效果与其他算法并无明显差异.

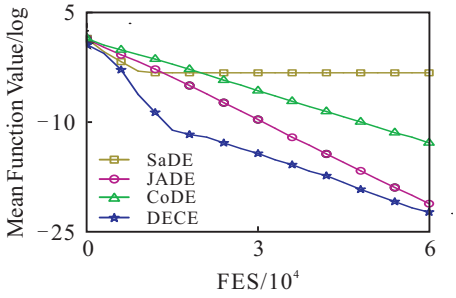
对比表3所列数据,在给定2000N的函数评价次数内,本文所提DECE算法在全部18个测试函数中有15个测试函数的结果优于其他算法,只对测试函数f_3、f_{10}的优化结果逊于SaDE,对测试函数f_7的优化结果逊于JADE,对其余测试函数的优化均好于对比算法. 从数值上看,DECE算法的大部分优化结果均优于其他算法多个数量级,尤其是f_2的Mean值分别优于SaDE、JADE、CoDE算法8、12、22个数量级,f_{13}的Mean值分别优于SaDE、JADE、CoDE算法9、6、8个数量级,Std值类似,可以看出DECE算法优化结果较优且较为稳定. 非参数假设检验的结果表明,DECE算法的优化性能较其他算法有显著优势,分别优于SaDE、JADE、CoDE算法11、10、13个测试函数. 另外,虽对8个测试函数(f_4、f_5、f_8、f_9、f_{15}、f_{16}、f_{17}、f_{18})的优化与对比算法相比并无显著差异,但同对比算法一样均能优化得到全局最优解.

表 4　CLPSO、CMA-ES、GL-25和DECE算法函数误差的平均值和标准偏差

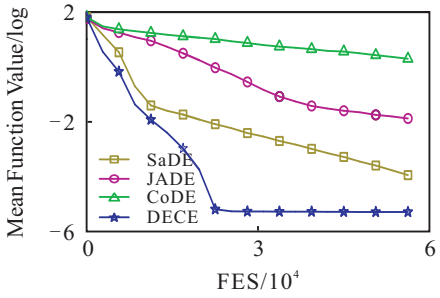
Fun	N	CLPSO			CMA-ES			GL-25			DECE	
		Mean	Std	Sig	Mean	Std	Sig	Mean	Std	Sig	Mean	Std
f_1	30	9.13e-14	3.33e-14	+	1.97e-29	2.07e-30	+	**4.78e-87**	**1.69e-86**	−	1.60e-36	3.73e-36
f_2	30	9.79e-15	5.44e-15	+	3.75e-28	4.61e-29	+	**2.66e-78**	**1.25e-77**	−	1.07e-45	2.41e-46
f_3	30	4.48e-09	1.48e-09	+	2.03e-14	9.76e-16	−	**2.98e-28**	**1.12e-27**	−	2.61e-10	1.82e-10
f_4	30	3.37e-16	7.98e-17	+	4.81e-17	5.60e-17	+	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**
f_5	30	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**
f_6	30	4.73e+00	1.20e+00	+	**2.95e-27**	**4.82e-28**	−	3.14e-02	8.77e-02	+	1.08e-13	6.57e-14
f_7	30	1.95e+01	2.62e+00	+	**2.66e-01**	**1.01e+00**	−	2.21e+01	6.19e-01	+	1.70e+01	8.38e-01
f_8	30	2.40e-09	3.67e-09	+	1.40e-03	3.70e-03	+	1.61e-15	4.55e-15	+	**0.00e+00**	**0.00e+00**
f_9	30	**0.00e+00**	**0.00e+00**	≈	1.30e+01	2.50e+00	+	3.14e-05	3.97e-14	+	**0.00e+00**	**0.00e+00**
f_{10}	30	1.13e-17	6.31e-18	+	9.13e-01	1.06e+00	+	8.82e-31	4.10e-30	+	**9.24e-31**	**2.55e-30**
f_{11}	30	5.94e-17	3.31e-17	+	1.10e-03	3.35e-03	+	**1.28e-30**	**5.05e-30**	−	5.17e-29	2.51e-29
f_{12}	30	9.76e-08	2.38e-08	+	1.93e+01	1.97e-01	+	1.09e-13	1.91e-13	+	**3.64e-15**	**1.12e-15**
f_{13}	30	6.26e+03	9.93e+02	+	5.59e-10	7.36e-11	+	2.49e+03	4.07e+02	+	**0.00e+00**	**0.00e+00**
f_{14}	30	2.33e-04	9.49e-05	+	8.96e-02	1.46e-01	+	3.55e-04	9.33e-04	+	**3.09e-06**	**1.51e-07**
f_{15}	4	**0.00e+00**	**0.00e+00**	≈	2.46e-02	5.60e-02	+	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**
f_{16}	4	3.15e-04	1.07e-04	+	3.59e-03	7.50e-03	+	2.19e-04	6.73e-05	+	**0.00e+00**	**0.00e+00**
f_{17}	2	**0.00e+00**	**0.00e+00**	≈	5.44e-02	2.07e-01	+	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**
f_{18}	2	**0.00e+00**	**0.00e+00**	≈	3.99e-01	9.07e-01	+	**0.00e+00**	**0.00e+00**	≈	**0.00e+00**	**0.00e+00**
+/≈/−		13/5/0			14/1/3			9/5/4				



(a) f_2



(b) f_{11}



(c) f_{14}

图 2　SaDE、JADE、CoDE和DECE算法对部分函数的平均收敛速度曲线

图2给出了测试函数 f_2、f_{11} 和 f_{14} 的收敛曲线,图中的数据为基于算法30次独立优化的平均结果,以算法函数评价次数为横坐标,平均函数值为纵坐标,为方便绘图,横坐标均取函数评价次数,纵坐标均取函数值的对数. 对图2收敛曲线进行分析可知,由于本文所提DECE算法引入了共轭增强策略,显著加快了收敛速度. 如图2所示,DECE算法在前期表现出优越的优化性能,并一直以稳定的速度较快地向全局最优解收敛. 其次,虽然SaDE的收敛速度仅次于本文算法,但在函数 f_{11} 的优化中陷入了局部最优而未能得到满意解.

2.2　与非DE算法比较

选用CLPSO、CMA-ES、GL-25三种非DE算法与所提DECE算法进行比较,上述4种算法的终止条件均设置为函数评价次数FES = 5 000N,其中 N 为测试函数各自的维数.

表4给出了CLPSO、CMA-ES、GL-25和DECE算法在30次独立运行所获得的结果的平均值和标准差. 对比表4所列数据,在给定5 000N 的函数评价次数内,与CLPSO、CMA-ES和GL-25这3种非DE算法进行比较,DECE算法对测试函数 f_6 和 f_7 的优化结果逊于CMA-ES,对测试函数 f_1、f_2、f_3、f_{11} 的优化结果逊于GL-25,而对其余测试函数的优化结果均优于对比算法. 与state-of-the-art算法比较的结论类似,无论数值还是显著性,本文所提DECE算法在解的质量和

稳定性方面都有显著优势,分别优于CLPSO、CMA-ES和GL-25这些非DE算法13、14、9个测试函数. 同样对6个测试函数(f_4、f_5、f_9、f_{15}、f_{17}、f_{18})的优化结果,所有算法均能找到全局最优解,因此与对比算法相比无显著差异.

3 结 论

本文提出了一种基于共轭增强策略的差分进化算法,通过设计共轭增强策略实现对DE算法的改进. 由于DE算法后期收敛速度变慢,降低了整个算法的搜索效率,因此DECE算法通过个体选择策略组建子种群,进而利用个体的时间和空间知识构建共轭增强方向实现子种群的局部增强,能够有效提高算法的优化性能. 通过18个标准测试函数验证了DECE算法在计算代价、可靠性、解的质量及收敛速度方面的优越性能,实验结果表明,DECE算法能够平衡全局探测能力与局部搜索能力,是一种有效的全局优化算法. 下一步的主要工作将集中在对复杂实际问题的应用研究.

参考文献(References)

[1] Floudas C A, Gounaris C E. A review of recent advances in global optimization[J]. J of Global Optimization, 2009, 45(1): 3-38.

[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] 刘宏志, 高立群, 孔祥勇, 等. 改进差分进化算法在可靠性冗余分配问题中的应用[J]. 控制与决策, 2015, 30(5): 917-922.
(Liu H Z, Gao L Q, Kong X Y, et al. Improved differential evolution algorithm for solving reliability redundancy allocation problem[J]. Control and Decision, 2015, 30(5): 917-922.)

[4] Zhang G J, Zhou X G, Yu X F, et al. Enhancing protein conformational space sampling using distance profile-guided differential evolution[J]. IEEE/ACM Trans on Computational Biology and Bioinformatics, DOI:10.1109/TCBB.2016.2566617.

[5] Lozano M, Garcia-Martinez C. Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report[J]. Computers & Operations Research, 2010, 37(3): 481-497.

[6] Storn R, Price K V. Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces[J]. J of Global Optimization, 1997, 11(4): 341-359.

[7] Price K, Storn R, Lampinen J A. Differential evolution —— A practical approach to global optimization[J]. Springer Optimization & Its Applications, 2005, 141(2): 1-24.

[8] Zhou X G, Zhang G J, Hao X H, et al. A novel differential evolution algorithm using local abstract convex underestimate strategy for global optimization[J]. Computers & Operation Research, 2016, 75(11): 132-149.

[9] Wang Y, Cai Z, Zhang Q. Differential evolution with composite trial vector generation strategies and control parameters[J]. IEEE Trans on Evolutionary Computation, 2011, 15(1): 55-66.

[10] Qin A K, Huang V L, Suganthan P N. Differen- tial evolution algorithm with strategy adaptation for global numerical optimization[J]. IEEE Trans on Evolutionary Computation, 2009, 13(2): 398-417.

[11] 张贵军, 周晓根. 基于抽象凸下界估计的群体全局优化算法[J]. 控制与决策, 2015, 30(6): 1116-1120.
(Zhang G J, Zhou X G. Population-based global optimization algorithm using abstract convex underestimate[J]. Control and Decision, 2015, 30(6): 1116-1120.)

[12] 周晓根, 张贵军, 郝小虎. 局部抽象凸区域剖分差分进化算法[J]. 自动化学报, 2015, 41(7): 1315-1327.
(Zhou X G, Zhang G J, Hao X H. Differential evolution algorithm with local abstract convex region partition[J]. Acta Automatica Sinica, 2015, 41(7): 1315-1327.)

[13] 周晓根, 张贵军, 梅珊, 等. 基于抽象凸估计选择策略的差分进化算法[J]. 控制理论与应用, 2015, 32(3): 388-397.
(Zhou X G, Zhang G J, Mei S, et al. Differential evolution algorithm based on abstract convex underestimate selection strategy[J]. Control Theory & Applications, 2015, 32(3): 388-397.)

[14] Zhang J, Sanderson A C. JADE: Adaptive differential evolution with optional external archive[J]. IEEE Trans on Evolutionary Computation, 2009, 13(5): 945-958.

[15] Liang J J, Qin A K, Suganthan P N, et al. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions[J]. IEEE Trans on Evolutionary Computation, 2006, 10(3): 281-295.

[16] Hansen N, Ostermeie A. Completely Derandomized Self-Adaptation in Evolution Strategies[J]. Evolutionary Computation, 2001, 9(2): 159-195.

[17] Garcia-Martinez C, Lozano M, Herrera F, et al. Global and local real-coded genetic algorithms based on parent-centric crossover operators[J]. European J of Operational Research, 2008, 185(3): 1088-1113.

[18] Corder G W, Foreman D I. Nonparametric statistics for non-statisticians: A Step-by-step approach[M]. Hoboken: John Wiley & Sons, 2009: 38-55.

（责任编辑：齐 霁）