雙清學術预印本
PrePrints

雙清學術出版社
Contents lists available at www.qingpress.com
Journal homepage: qingpress.com/zh-cn/journals/26

一种自适应差分进化算法求解连续函数优化问题*

项紫怡 [1*]，张颖聃 [2]

1. 项紫怡，湖北文理学院，湖北襄阳(441053)，fourierea@163.com
2. 张颖聃，中山市外国语学院，广东中山(528400)
*. 通讯作者：项紫怡，湖北文理学院，fourierea@163.com

摘 要：中原始差分进化(*DE*)算法的控制参数采用固定值，不能进行动态调整，从而影响了 *DE* 算法的整体优化性能。针对这一问题，在详细分析了变异和交叉策略后，提出了一种自适应差分进化(*ADE*)算法。该算法采用自适应机制动态调整变异因子和交叉概率，能够有效的平衡算法的全局和局部优化能力。基于 6 个基准测试函数的实验数据和仿真结果表明，*ADE* 算法既能提高算法的搜索精度，又能提高算法的收敛速度。

关键词：差分进化算法, 自适应机制, 变异, 交叉, 动态调整

引言

差分进化算法（Differential Evolution， *DE* ）是 Rainer Storn 和 Kenneth Price 教授在 1995 年提出的一种群体智能优化算法[1]，该算法的结构简单、控制参数少并且容易实现，广泛的应用于图像处理[2]、工程设计[3]、物流选址[4]等领域。*DE* 算法采用变异和交叉策略，引导算法进行全局和局部搜索，寻找优化问题的最优值，但是原始 *DE* 算法采用固定的变异因子和交叉概率[5-6]，容易陷入局部最优，影响了算法的收敛速度和求解精度。针对这一缺陷，分析了 *DE* 算法的变异因子和交叉概率对算法优化性能的影响规律，提出了一种自适应差分进化算法（ *ADE* ）。 *ADE* 算法在寻优过程中，按照自适应机制动态调整变异因子和交叉概率的大小，有效引导算法的全局和局部寻优过程。将 *ADE* 算法用于求解 6 个连续基准函数优化问题，相关实验数据和仿真结果表明，自适应差异进化（ *ADE* ）算法的求解精度和收敛速度得到了明显的提高，能够高效的求解连续函数优化问题。

1　原始 DE 算法

*基金项目：国家级大学生创业实践训练计划资助项目（批号：202010404011s）

Article history: Available online July 14, 2022
To cite this paper: 项紫怡, 张颖聃(2022). 一种自适应差分进化算法求解连续函数优化问题. 雙清學術预印本，卷 1，第 1 期, 1-8.
Doi: https://doi.org/10.55375/preprints.2022.1.1

DE 算法是一种基于种群差分信息的进化搜索方法,采用变异、交叉和选择策略产生下一代种群，引导种群向全局最优解移动，直到满足算法的终止条件，停止进化并输出最优解[1]。假设种群规模是 N ，搜索空间是 D 维， *DE* 算法随机生成初始种群，则种群中的个体可以表示为 $\vec{x}_i=(x_{i1},x_{i2},\mathrm{K},x_{iD})^T,i=1,2,\mathrm{K},N$ 。 *DE* 算法的执行的步骤如下所示：

(1) 差分变异操作：

在 *DE* 算法中，父代个体形成的差分向量是主要的变异成分，采用多个父代个体的线性组合产生变异个体。假设当前进化个体为 $\vec{x}_i(k)$ ，按照公式（1）生成变异个体 $\vec{w}_i(k+1)$ 。

$$\vec{w}_i(k+1)=\vec{x}_{r_1}(k)+F[\vec{x}_{r_2}(k)-\vec{x}_{r_3}(k)] \tag{1}$$

其中， $i=1,2,\mathrm{K},N$ ， k 为当前的进化代数， $\vec{x}_{r_1}(k),\vec{x}_{r_2}(k),\vec{x}_{r_3}(k)$ 是种群中随机选择的三个独立个体，而且 $r_1\neq r_2\neq r_3\neq i$ ， F 是变异因子,并且 $F\in[0,2]$ 。取向量 $\vec{x}_{r_2}(k)$ 和向量 $\vec{x}_{r_3}(k)$ 的向量差，经 F 缩放之后，再加到向量 $\vec{x}_{r_1}(k)$ 上，最终生成变异个体 $\vec{w}_i(k+1)$ 。

(2) 交叉操作：

在将当前进化个体 $\vec{x}_i(k)$ 与变异个体 $\vec{w}_i(k+1)$ 进行混合交叉，按照公式（2）生成实验个体 $\vec{u}_i(k+1)$ 。

$$u_{ij}(k+1)=\begin{cases}w_{ij}(k+1), & if\ \ rand_j\le Cr\ or\ \ j=jrand\\ x_{ij}(k), & otherwise\end{cases} \tag{2}$$

其中， $i=1,2,\mathrm{K},N,\ j=1,2,\mathrm{K},D$ ， $rand_j$ 是[0,1]中的一个随机数。 $CR\in[0,1]$ 是交叉概率，它的取值越大，表示变异个体 $\vec{w}_i(k+1)$ 对实验个体 $\vec{u}_i(k+1)$ 的贡献越大。 $jrand$ 是 $[1,D]$ 之间的随机整数，它能够保证 $\vec{u}_i(k+1)$ 中至少有一维的值与变异个体 $\vec{w}_i(k+1)$ 的某一维相同，避免与 $\vec{x}_i(k)$ 完全一样，进一步避免种群陷入进化停滞状态。

(3) 选择操作：

利用贪婪选择策略按照公式（3）在当前进化个体 $\vec{x}_i(k)$ 和实验个体 $\vec{u}_i(k+1)$ 之间执行选择操作。

$$\vec{x}_i(k+1)=\begin{cases}\vec{u}_i(k+1), & if\ \ f(\vec{u}_i(k+1))\le f(\vec{x}_i(k))\\ \vec{x}_i(k), & otherwise\end{cases} \tag{3}$$

以连续函数优化问题作为评价目标，若实验个体 $\vec{u}_i(k+1)$ 的适应度函数值小于当前进化个体 $\vec{x}_i(k)$ 的适应度函数值时，实验个体 $\vec{u}_i(k+1)$ 被保存到下一代，否则，当前进化个体 $\vec{x}_i(k)$ 仍将保留至下一代。

2 改进的 DE 算法

在算法进化过程中，如果全局搜索能力较强，就能够快速的到达最优值附近，搜索过程中也不容易陷入局部最优，当算法搜索到最优值附近时，就需要加强局部搜索能力，从而提高求解精度。变异策略和交叉策略引导着算法的全局和局部搜索过程，并通过变异因子 F 和交叉概率 CR 来平衡算法的全局和局部搜索能力，因此，在进化过程中，固定的控制参数不能很好的起到引导作用，需要采用自适应机制动态的调整 F 和 CR 的大小。

2.1 自适应变异因子

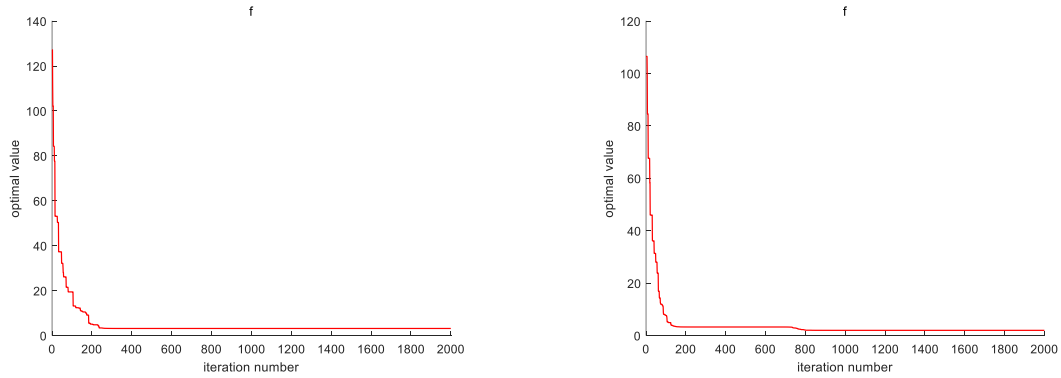
变异策略引导着算法进行全局搜索，能够快速的在解空间中寻找最优值，但是，由于 F 是固定值，随着迭代次数的增加，限制了算法的寻优精度，因此，需要对变异因子 F 做适当修正，从而平衡算法的全局和局部搜索能力。近年来，许多学者也对变异因子 F 做了适当的修正，例如，文献 4 中的 *IDE* 算法对 F 采用公式（4）进行线性变化修正，并取得了一定的效果[7]。

$$F = F_{\max} - \frac{0.4 * k}{k_{\max}} \tag{4}$$

其中，k 是当前迭代次数。$k_{\max}$ 是最大迭代次数，$F \in [0.1, 0.5]$。在迭代初期，较大的 F 优化效果明显，收敛曲线平滑稳定，在迭代后期,较小的 F 对算法收敛精度有所提高，对避免陷入局部最优和收敛速度有一定的改善作用。但是，变异操作扩大了种群多样性，具有一定的随机性，而变异因子 F 的线性变化从某些方面来说也限制了种群的繁衍，减少了种群的多样性，同时在公式（1）中，F 的变化仅仅取决于迭代的次数，没有根据上一代的 F 动态调整，因此，对公式（4）做了相应的修改，使 F 能够根据上一代的值进行幂指数的变，如公式（5）所示：

$$F_{k+1} = F_{\max} - (F_{\max} - F_{\min}) * \exp(\frac{2\pi k}{k_{\max}} * \frac{F_k}{F_{\max}}), F \in [0.1, 0.5] \tag{5}$$

其中，F_k 是第 k 代的 F 数值；F_{k+1} 是第 $k+1$ 代 F 数值。为了验证改进的效果，假设公式（5）的 DE 算法命名为 DE_F，分别将 IDE 和 DE_F 代入多模基准测试函数 Rastrigin，得到的收敛曲线如图 1 和图 2 所示。



(a) IDE 收敛曲线　　(b) DE_F 收敛曲线

图 1　IDE 和 DE_F 的收敛曲线图

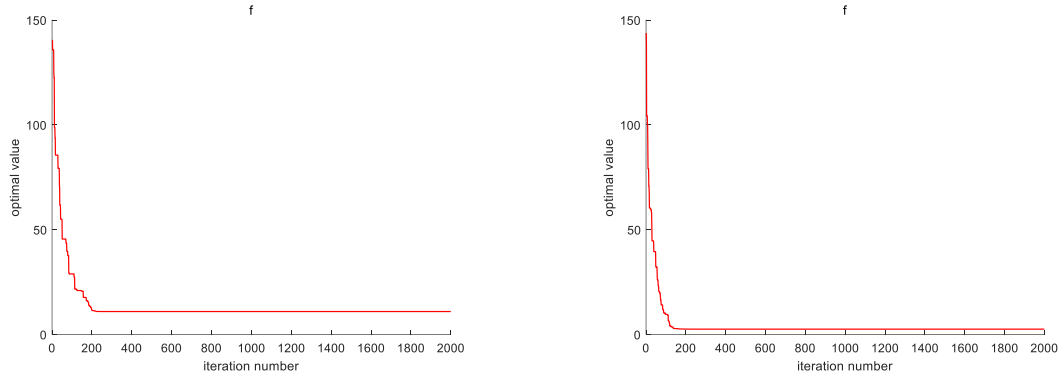
从图 1 可知，在 200 代左右时，DE_F 要比 IDE 略快到达最优值附近，说明 DE_F 的全局搜索能力要好，但是 IDE 在接下来的代数中，收敛精度并没有进一步的提高，而 IDE 在 750 代时，收敛精度得到了进一步的提高。

2.2　自适应交叉概率

交叉策略可以根据当前代的个体扩大种群多样性，既有利于选择，也能很好的模拟生物的特性。大量研究表明当 CR 为常数时，在一定平衡范围内，收敛速度与取值成正比。因此，在兼顾全局搜索和局部搜索情况下，可以改变 CR 的值来加快收敛速度。针对交叉概率取值对收敛和优化种群的影响，CR 的自适应策略如公式（6）所示，并命名为 DE_CR。

$$CR_{k+1} = CR_{\max} - (CR_{\max} - CR_{\min}) * \exp(\frac{2\pi k}{k_{\max}} * \frac{CR_k}{CR_{\max}}), \quad CR \in (0, 1] \tag{6}$$

其中，CR_k 是第 k 代的 CR 数值；CR_{k+1} 是第 $k+1$ 代 CR 数值。假设公式（6）的 DE 算法命名为 DE_CR，分别将 DE 和 DE_CR 代入多模基准测试函数 Rastrigin，得到的收敛曲线如图 2 所示。



(a) *DE* 收敛曲线　　(b) *DE _CR* 收敛曲线

图 2　*DE* 和 *DE _CR* 的收敛曲线图

从图 2 的收敛曲线上可以看出，采用这种策略的收敛性非常好，运行速度快，曲线平滑，没有出现坍塌式选择，即兼顾了全局搜索和局部搜索，又对种群的多样性保持的很好，不会陷入局部最优。

2.3　自适应交叉概率

通过上述分析可以知道，*DE _F* 主要在优化算法的全局搜索能力，*DE _CR* 主要在于提高算法的收敛精度，因此，将式（5）和式（6）结合起来，即为自适应差分进化算法（*ADE*），如式（7）所示。

$$\begin{cases} F_{k+1} = F_{\max} - (F_{\max} - F_{\min}) * \exp(\dfrac{2\pi k}{k_{\max}} * \dfrac{F_k}{F_{\max}}) \\ CR_{k+1} = CR_{\max} - (CR_{\max} - CR_{\min}) * \exp(\dfrac{2\pi k}{k_{\max}} * \dfrac{CR_k}{CR_{\max}}) \end{cases} \tag{7}$$

将式（7）代入 Rastrigin 基准函数仿真，得到如图 3 所示的收敛曲线。

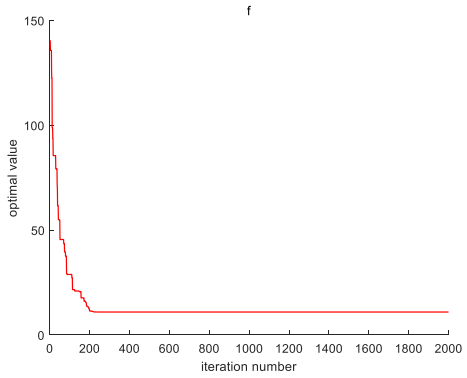


图 3　*ADE* 的收敛曲线

从图 1 至图 3 可知，*DE* 、*IDE* 、*DE _F* 和 *DE _CR* 在 200 代左右就可以到达最优值附近，收敛精度有所差异；*ADE* 在 100 代之前就能够到达最优值附近，并且收敛精度也高于前面四种算法，表现出了较强的全局搜索能力和较高的搜索精度，由此可见，通过自适应修正策略修正变异因子和交叉概率可以有效的平衡算法的全局和局部搜索能力，明显的提高算法的优化性能。

3　基准函数测试

3.1　基准函数

为了进一步验证改进后算法的有效性，将对 *ADE* 算法进行基准函数测试[8-11]，并与 *DE* 和 *IDE* 算法进行比较。需要用到的基准函数集如表 1 所示。该基准测试函数集中的函数各有特色，既有单模函数又有多模函数，能够比较全面的检测各算法的性能。

表 1 基准函数表

序号	函数	取值范围	X*	最优值
1	Ackley	[-32.768,32.768]	(0,...,0)	0
2	Sum Squares	[-5.12,5.12]	(0,...,0)	0
3	Schaffer N.2	[-100,100]	(0,0)	0

续表 1 基准函数表

序号	函数	取值范围	X*	最优值
4	Booth	[-10,10]	(1,3)	0
5	Matyas	[-10,10]	(0,0)	0
6	Zakharov	[-5,10]	(0,...,0)	0

3.2 参数设置

算法中的种群规模 *NP* 设置为 10；最大迭代次数 $k_{\max}$ =2000；交叉概率 *CR* =0.2；变异因子 *F* =0.6 [12]。

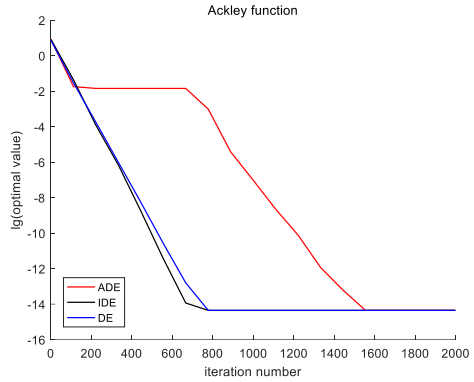
3.3 数值仿真结果与分析

统计 *DE* 、 *IDE* 和 *ADE* 这三种算法对每个基准函数独立运行 20 次求得的最优值结果，并把最好值（Best）、最差值（Worst）、平均值（Mean）和标准差（Standard Deviation，SD）作为算法性能优劣的评价指标，用来分析每个算法对函数优化问题的性能。三种算法对这 6 个基准测试函数的优化结果分别如表 2-7 所示。同时，为了能够直观的观察算法的动态寻优过程，图 4 绘制了三种算法对 6 个基准测试函数的收敛曲线。曲线图的纵轴表示算法每次迭代获得的最优值（为了便于比较，对获得的最优函数值取其以 10 为底的对数），曲线图的横轴表示算法进化代数。

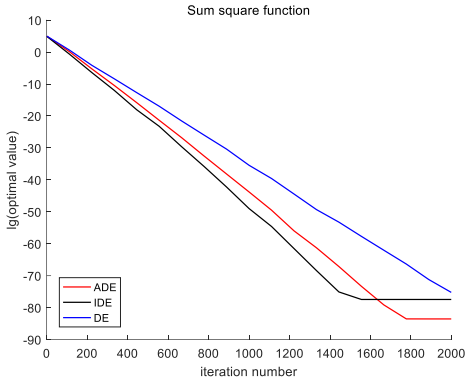
表 2 三种算法对基准测试函数的优化结果比较

函数	算法	Best	Worst	Mean	SD
Ackley	DE	4.4409e-15	4.4409e-15	4.4409e-15	0
	IDE	4.4409e-15	4.4409e-15	4.4409e-15	0
	ADE	**8.8818e-16**	**4.4409e-15**	**4.0856e-15**	**1.1235e-15**
Sum Squares	DE	6.0841e-76	5.0298e-73	1.2034e-73	1.8575e-73
	IDE	1.5760e-78	4.1330e-76	4.1330e-77	1.3070e-76
	ADE	**2.5016e-91**	**6.3652e-88**	**6.3799e-89**	**2.0123e-88**
Schaffer N.2	**DE**	**0**	**0**	**0**	**0**
	IDE	**0**	**0**	**0**	**0**
	ADE	**0**	**0**	**0**	**0**
	DE	**0**	**0**	**0**	**0**

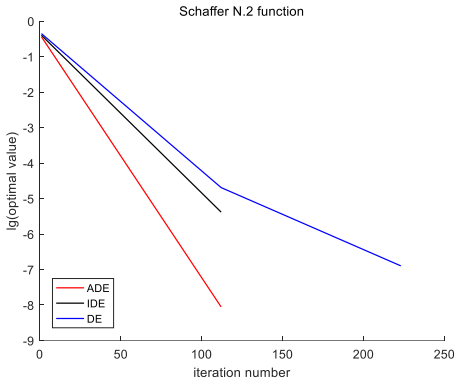
Booth	**IDE**	**0**	**0**	**0**	**0**
	ADE	**0**	**0**	**0**	**0**
Matyas	DE	3.8232e-27	1.3556e-21	1.5448e-22	4.2332e-22
	IDE	1.0499e-67	5.1950e-64	1.4284e-64	1.8467e-64
	ADE	**4.0529e-98**	**4.7506e-80**	**4.7752e-81**	**1.5014e-80**
Zakharov	DE	1.2068e-04	2.19e-02	5.0000e-03	6.8000e-03
	IDE	3.0848e-06	1.8e-03	7.4422e-04	5.6724e-04
	ADE	**1.8908e-10**	**6.3056e-08**	**9.9040e-09**	**1.9322e-08**



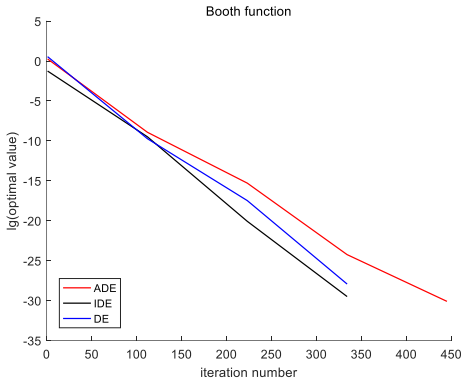
（a） Ackley 函数



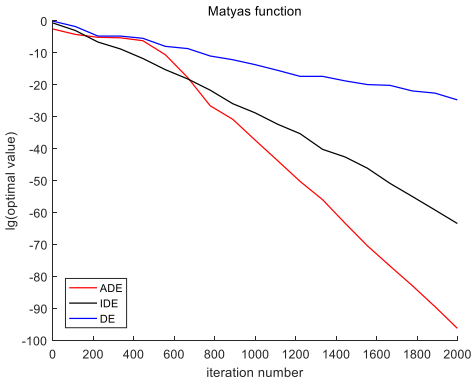
（b） Sum Squares 函数



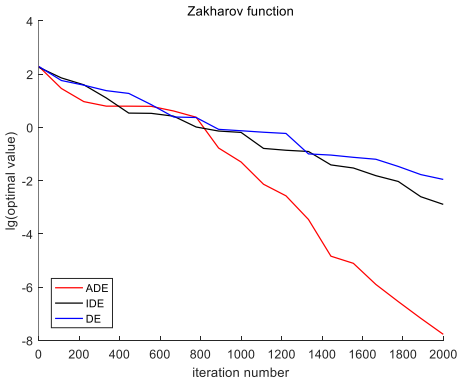
（c） Schaffer N.2 函数



（d） Booth 函数



（e） Matyas 函数



（f） Zakharov 函数

图 4 基准测试函数收敛曲线

从表 2 可知，对于 Ackley 函数，在 20 次的独立运行中， *DE* 和 *IDE* 每次都能找到局部最优值 4.4409e-15， *ADE* 大部分情况下也是找到局部最优值 4.4409e-15，在少数情况下， *ADE* 找到的最好值要略高于前两种算法一个数量级；图 4 中的（a）显示了三种算法的动态收敛过程，该图中 *DE* 和 *IDE* 在相同的迭代次数

时搜索到局部最优值， *ADE* 在寻优过程早期陷入局部最优，在第 700 代时跳出该局部最优，动态收敛过程稍微欠缺，总体而言， *ADE* 的优化性能略好于 *DE* 和 *IDE* 。对于 Sum Squares、Matyas 和 Zakharov 函数， *ADE* 的最好值、最差值、平均值和标准差都要好于 *DE* 和 *IDE* 2 至 4 倍，因此， *ADE* 的寻优精度和稳定性都要远好于 *DE* 和 *IDE* ；图 4 中的（b）显示 *ADE* 的收敛速度介于 *IDE* 和 *DE* 之间，收敛精度要远高于 *DE* 和 *IDE* ；图 4 中的（e）和（f）表明 *ADE* 的寻优速度也要比 *DE* 和 *IDE* 快的多，收敛精度也要更好。对于 Schaffer N.2 和 Booth 函数，独立运行时， *DE* 、 *IDE* 和 *ADE* 都找到了最优值 0，收敛精度和稳定性都较好，图 4 中的（c）显示， *ADE* 的动态收敛表现最好，图 4 中的（d）显示， *ADE* 的动态收敛要逊色于 *DE* 和 *IDE* 。

基于上述的数值实验与统计分析，变异因子和交叉概率的动态自适应调整机制显著提高了原始 *DE* 算法的优化性能。与 *DE* 和 *IDE* 相比，对于求解绝大部分连续函数优化问题来说， *ADE* 在收敛速度、寻优精度、算法稳定性和动态收敛过程等方面的表现更优秀。

4 结语

针对原始 *DE* 算法的变异因子和交叉概率采用固定值，难以满足算法进化过程中的动态要求，影响算法优化性能的缺陷，提出一种自适应差分进化（ *ADE* ）算法。该算法显著提升了原始 *DE* 算法的寻优性能。将提出的 *ADE* 算法与 *DE* 和 *IDE* 相比较，采用 6 个经典的连续函数优化问题进行仿真实验，仿真结果与统计分析表明所提出的 *ADE* 算法对于绝大多数函数优化问题而言，可以取得更优的求解精度和收敛速度。

参考文献

[1] R. Storn, K. Pricel. “Differential evolution: a simple and efficient heuristic for global optimization over continuous space.” Journal of Global Optimization. **11** (1997): 341-359

[2] A. Datta, S. Ghosh, and A. Ghosh. “Self-adaptive differential evolution for feature selection in hyperspectral image data.” Applied Soft Computing. **13** (2013): 1969–1977

[3] Y. Li, L. Rao and R. He. “ A novel combination method of electrical impedance tomography inverse problem for brain imaging.” IEEE Transactions on Magnetics. **41** (2005): 1848 - 1851

[4] T. Thongdee, R.Pitakaso. “ Differential evolution algorithms solving a multi-objective, source and stage location-allocation problem.” Industrial Engineeering and Management Systems. **14**(2015): 11-21

[5] S. Sakr Walaa, R.A. EL-Sehiemy, and A.M. Azmy. “ Adaptive differential evolution algorithm for efficient reactive power management.” Applied Soft Computing. **53**(2017): 336-351

[6] D. Gao, G.G. Wang and W. Pedrycz. “ Solving Fuzzy Job-Shop Scheduling Problem Using DE Algorithm Improved by a Selection Mechanism.” IEEE Transactions on Fuzzy Systems. **2**(2020): 3265-3275

[7] A.A. Abou El Ela, M.A. Abido and S.R. Spea. “ Differential evolution algorithm for optimal reactive power dispatch.” Electric Power Systems Research. **81**(2011): 458-464

[8] D. Karaboga, B. Akay. “A comparative study of artificial bee colony algorithm.” Applied mathematics and computation. **214** (2009): 108-132

[9] A.R. Hedar, M. Fukushima. “Tabu search directed by direct search methods for nonlinear global optimization.” European Journal of Operational Research. **170** (2006): 329-349

[10] L. Wang, F. Zou, X.H. Hei, D.D. Yang, D.B. Chen, Q.Y. Jiang and Z.J. Cao. " A hybridization of teaching-learning-based optimization and differential evolution for chaotic time series prediction." Neural Computing and Application. **25**(2014): 1407-1422

[11] P.N. Suganthan, N. Hansen et al., Problem definitions and evaluation criteria for the CEC 2005 special session on real parameter optimization. Singapore: Nanyang, 2005

[12] S.S. Dhillon, S. Agarwal, and G.G. Wang, J.S. " Automatic generation control of interconnected power systems using elephant herding optimization." Lecture Notes in Electrical Engineering. **607**(2020): 9-18