Palestine Journal of Mathematics

Vol. 11(Special Issue I)(2022) , 65–74

MRL-JAYA: A FUSION OF MRLDE AND JAYA ALGORITHM

Pravesh Kumar and Amit Sharma

Communicated by Shimpi Singh Jadon

Abstract Jaya algorithm is a newly developed metahueristics algorithm to solve optimization problems. In this paper we have proposed a new variant named MRL-Jaya which is a fusion of Jaya algorithms with modified random localization based DE (MRLDE) algorithm. MRL-Jaya connects two algorithms by a systematic approach to utilize the advantage of both in a single variant. MRL-Jaya has tested on 13 traditional and 6 shifted benchmark problems taken from literature. In last the result and comparison shows the efficiency of proposed variant.

Keyword: Optimization, Differential evolution Algorithm, Jaya Algorithm.

1 Introduction

A lot of problems occur in various sciences and engineering filed can be modelled as global optimization problems. Use of traditional nonlinear programming techniques may show to be ineffective for solving such problems because of the occurrence of non linearity, non continuity, non differentiability and multiple local/global optima. In recent years, many non-traditional and nature inspired based methods have been developed in the area of optimization. Some of the well-known non-traditional techniques are GA, ACO, DE, PSO, ABC, $TLBA$, Jaya Algorithm and so on. Differential Evolution (DE) algorithm is first proposed by Storn and Price in [1]. The advantage of DE is including ease of implement, reliable, robust and efficient optimization algorithm. The compact design and the use of fewer control parameters make it more popular. However it does not guarantees to converge to optimum always. So many researches have been carried out on its improved versions and applications in different fields during some last years. Some popular enhanced variants of DE are, JDE [2], ODE [3], $SaDE$ [4], $jADE$[5], $LeDE$ [6], and so on. Some other DE variants which have developed recently can be found in literature given in [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]. A well organized literature review of DE algorithm with full details can also be found in [22, 23, 24]. $MRLDE$ is an enhanced variant of DE algorithm proposed by Kumar and Pant [25] in 2012. It is based on modified mutation strategy in which base vector is selected in a strategic mode to pick up the performance of algorithm. The significance of $MRLDE$ in optimization problems of various engineering and science can be found in [26, 27, 28, 29]. Jaya algorithm is newly developed metaheuristics algorithm in the field of optimization. It is proposed by Rao [30] in 2015. Like DE, It is also a gradient-free and population based search technique which repeatedly modified the solutions. The main advantage of the algorithm is its compact design and ease to use. Due to this reason it has become quite popular rapidly in solving many real life optimization problems. Some of recently developed Jaya algorithms variants and its applications are given in [31, 32, 33, 34, 35, 36, 37, 38, 39, 40] In this paper, a new variant named MRL-Jaya is proposed. MRL-Jaya is a hybridization of Jaya algorithms with modified random localization based$DE(MRLDE)$ algorithm MRL-Jaya connects two algorithms by a systematic approach to utilize the advantage of both in a single variant. The importance of planned algorithm is discussed later in the paper. Organization of the paper as follow; Introduction of $MRLDE$ and Jaya Algorithm is given in section 2. Proposed MRL-Jaya is explained in section-3. In section-4, explanation of Benchmark functions, various parameter settings, performance criteria, results and comparison of MRL-Jaya with other variants are discussed. In last the entire study has concluded in section 5:

2 Related Work

In this section basic structure and working of Jaya and $MRLDE$ is explained as follows:

2.1 MRLDE:

$MRLDE$ is an enhanced DE variant which is developed by taking base vector in an organized way to do mutation operation. This algorithm starts by a uniform distributed and randomly generated population of size N, say $POP = \{P_i(g) : i = 1 : N\}$. Now the search space is divided in three sub-regions say $POP_{best}(g)$, $POP_{medium}(g)$, and $POP_{worst}(g)$ according to fitness value of vectors After that mutation, crossover and selection operation are performed as defined in $MRLDE$ algorithm as below;

i **Mutation:** Select three mutually different vectors $P_{b1}(g)$, $P_{b2}(g)$ and $P_{b3}(g)$ from sub-regions $OP_{best}(g)$, $POP_{medium}(g)$ and $POP_{worst}(g)$ respectively for any target vector $P_i(g)$ and produce mutant vector by equation-1;

$$Y_i(g) = P_{b1}(g) + SC * [P_{b2}(g) - P_{b2}(g)] \tag{2.1}$$

Here SC is control factor contain value between [0, 2].

ii **Crossover:** By this operation, a trial vector $T_i(g) = \{t_{1,\,i},\, t_{2,\,i},\, ...t_{D,\,i}\}$ for $P_i(g)$ is generated by equation-2;

$$t_{j,\,i}(g) = \begin{cases} y_{j,\,i}, & \text{if}\, rand_1(0,\,1) < CR \;||\; j == k :\; k \in 1,\,2,\,...D) \\ p_{j,\,i,}, & \text{otherwise} \end{cases} \tag{2.2}$$

Here CR is crossover rate and $rand_1(0,\,1)$ is any uniform random numbers between 0 and 1.

iii **Selection:** Finally the supreme vector among $P_i(g)$ and $T_i(g)$ is selected for the next generation by equation -3;

$$P_i(g+1) = \begin{cases} T_i(g), & if\, fun\,[T_i(g)] < fun\,[P_i(g)] \\ P_i(g), & otherwise \end{cases} \tag{2.3}$$

A detailed explanation of $MRLDE$ and its working can be found in [25] – [29].

2.2 Jaya Algorithm:

Like $MRLDE$, Jaya algorithm also starts with a uniform distributed and randomly generated population of size N, say $POP = \{P_i(g) : i = 1 : N\}$ and then move towards the optimum solution by in the search space. Let $P_{best}(g)$ and $P_{worst}(g)$ are the global best and global worst vector in the search space in any generation g, then the new position for any vector $P_i(g)$ for next generation is created by equation-4 as below:

$$P_i(g+1) = P_i(g) + rand_2[0,1] * (P_{best}(g) - |P_i(g)|) - rand_3[0,1] * (P_{worst}(g) - |P_i(g)|) \tag{2.4}$$

where $rand_2[0,1]$ and $rand_3[0,1]$ are any uniform random numbers between 0 and 1. Now old vector $P_i(g)$ is replaced by new generated vector $P_i(g+1)$ if it has greater fitness value than $P_i(g+1)$ otherwise $P_i(g)$ will be retained for the next generation as given in equation -5.

$$P_i(g+1) = \begin{cases} P_i(g+1), & if\, fun\{P_i(g+1)\} < fun\{P_i(g)\} \\ P_i(g)\; else \end{cases} \tag{2.5}$$

The process will be repeated for each vector and generated a new population for next generation.

Working of MRL-Jaya

- Start with uniform distributed random generated population $POP = \{X_i(G), i=1, 2...N\}$.
  $P_i(g) = RAND\,[0,1] * (P_{UB} - P_{LB}) + P_{LB}$.
- Evaluate $fun\{P_i(G)\}$ for each i

WHILE (Max iteration not reached)
  FOR: i=1:N
  - Divide population in three sub-regions $POP_{best}(g)$, $POP_{medium}(g)$ and $POP_{worst}(g)$ of size N_1 , N_2 and N_3 respectively.
  - Select vectors $P_{b1}(g)$, $P_{b2}(g)$ and $P_{b3}(g)$ from $POP_{best}(g)$, $POP_{medium}(g)$ and $POP_{worst}(g)$ respectively
  - Generate $Y_i(g)$ and $T_i(g)$ using equation-1 and equation-2 respectively
    IF $fun\{T(g)\} < fun\{P_i(g)\}$
      $P_i(g+1) = T_i(g)$
    ELSE
      Create $P_i(g+1)$ by equation-4
  - Apply selection operation using equation-5
    END IF-ELSE
  END FOR
  - Update population $POP = \{P_i(g+1) : i = 1 : N\}$

END WHILE
Terminate

3 Proposed MRL-Jaya:

$MRLDE$ and Jaya are both prominent algorithms for global optimization. However they both do not gives assurance to grant optimum solution for all time. It has been checked that Jaya algorithm perform with a slow convergence rate however $MRLDE$ having fast convergence but can be stuck in local optima due to its greedy nature. MRL-Jaya is a simple and systematic hybridization of Jaya and $MRLDE$ which overcome the disadvantages of both algorithms. First it starts with $MRLDE$ algorithm and produced a trial vector $T_i(g)$ corresponding to target vector $P_i(g)$ as defined in section 2. Now if it is rejected for next generation then a new vector $P_i(g+1)$ being created by Jaya algorithm with respect to same $P_i(g)$ and applied the selection procedure based on fitness value as defined for Jaya algorithm is section 2 by equation-5. . By doing this we have an additional choice to find a new better vector corresponding to $P_i(g)$ and consequently we can improve the solution quality in each generation.

4 Result and Discussion

Various experiments have been conducted to check the significance of the proposed MRL-Jaya algorithm. A detail explanation of benchmark functions, parameter settings, comparison criteria and statistical result analysis is discussed in this section.

4.1 Benchmark Problems:

The experiments have been carried out on 13 non-shifted traditional and 6 shifted benchmark functions. All benchmark problems have been taken from various literatures [5],[6],[41] given and given in Table-1with their important properties.

4.2 Performance Measures:

The performance of algorithms has checked by following performance measures:

Table 1. Benchmark Functions

F	Name	Property	Boundary	Global value
$F1$	Sphere Function	Unimodal, Seperable, Scalable	$[-100, 100]^{-D}$	0
$F2$	Schwefel's Problem 2.22	Unimodal, Seperable, Scalable	$[-10, 10]^D$	0
$F3$	Schwefel's Problem 1.2	Unimodal, Non-seperable, Scalable	$[-100, 100]^D$	0
$F4$	Schwefel's Problem 2.21	Unimodal, Non-seperable, Scalable	$[-100, 100]^D$	0
$F5$	Generalized Rosenbrock's Function	Multimodal, Non-seperable, Scalable, narrow velly from local to global optimum	$[-30, 30]^D$	0
F6	Step Function	Unimodal, Seperable, Scalable	$[-100, 100]^p$	0
$F7$	Noise Function	Unimodal, Seperable, Scalable	$[-1.28, 1.28]^D$	0
$F8$	Quartic Noise Function	Multimodal, Seperable, Scalable, many local minima	$[-500, 500]^D$	0
$F9$	Generalized Rastrigin's Function	Multimodal, Seperable, Scalable, many local minima	$[-5.12, 5.12]^D$	0
$F10$	Multimodal, Seperable, Scalable, Ackley's Function	$[-32, 32]^D$	0	
$F11$	Generalized Griewank Function	Multimodal, Seperable, Scalable, many local minima	$[-600, 600]^D$	0
$F12$	Generalized Penalized Functions-I	Multimodal, Seperable, Scalable, many local		
$F13$	Generalized Penalized Functions-II	Multimodal, Seperable, Scalable, many local minima	$[-50, 50]^D$	0
SF_I	Shifted Sphere Function	Shifted, Unimodal, Seperable, Scalable	$[-100, 100]^D$	−450
SF_2	Shifted Schwefel 2.21 Function	Shifted, Unimodal, Non-seperable, Scalable	$[-100, 100]^D$	−450
SF_3	Shifted Rosenbrock Function	Shifted, Multimodal, Nonseperable, Scalable, Nerrow velly from local to global optimum	$[-100, 100]^D$	390
SF_4	Shifted Retrigin Function	Shifted, Multimodal, Seperable, Scalable, Several local optimum	$[-5.12, 5.12]^D$	−330
SF_5	Shifted Griewank Function	Shifted, Multimodal, Nonseperable, Scalable	$[-600, 600]^D$	−180
SF_6	Shifted Ackley Function	Shifted, Multimodal, Seperable, Scalable,	$[-32, 32]^D$	−140

Table 2. Parameter Settings

S.No	Parameter Name	Parameter Setting
1	Population Size (N)	100
2	Dimension (D)	30
3	Scale Factor Sc for MRLDE and MRL-Jaya	0.5
4	Crossover Factor Cr for MRLDE and MRL-Jaya	0.9
5	Size of $\text{N}_1,\ \text{N}_2,\ \text{N}_3$ for MRLDE and MRL-Jaya	$20\%, 40\%$ and 40% respectively
6	Total Run	100
7	Software Used	Matlab, Dev C++, Sigma Plot,

i **Average Error:** The minimum error $x|f(X) - f(X*)|$ where X* is the global optimum, is verified when the fixed Max iteration is attained in each run. After that the average error and standard deviation is taken of all run.

ii **Average NFEs:**The number of function evaluations (NFEs) is calculated when fixed error (VTR) is achieved i.e counted total NFE for $|f(X) - f(X*)| \leq VTR$ in each run.

iii **Acceleration Rate:** $AR = (1 - \frac{NFE_A}{NFE_B})$

iv **Convergence graphs:** The convergence graphs show the average fitness presentation in an experiment.

4.3 Parameter Setting:

Parameter settings for the study are given in Table-2. All parameter settings have been taken same to all so that a fair comparison and analysis can be carried out. All experiments are executed on a laptop with 8GB memory, 2.6$GHZCPU$, intel core™ i3 processor, 64-bit, Windows 10 and using software like Matlab R2012b and $DEVC++$.

4.4 Results and Comparison of MRL-Jaya with Jaya and $MRLDE$ Algorithm:

In this section comparison of MRL-Jaya with its parent algorithm Jaya and $MRLDE$ algorithm is discussed. The numerical results have been taken in term of error by fixing the maximum iteration for all functions as shown in Table-3. Here it can be easily observed that proposed MRL-Jaya accomplished the desire results for all function compare to others. A non-parametric Wilcoxon statistical test is also performed which can be shown in last two columns of Table. We can simply see that MRL-Jaya gives better performance than all other algorithm except function F9 where Java gives superior performance among all algorithms. For function $F6$ and $F8$, all algorithms perform similar while there is no significant difference between the performance of $MRLDE$ and MRL-Jaya performs for the$F7$ and $F11$.

Table 3. Results and Comparisons in term of Mean Error, standard deviation and Wilcoxon test at $a = 0.05$. Here '1','2','3' denote 'Jaya', 'MRLDE' and 'MRL-Jaya' respectively

Fun	Max-Iteration	Mean Error (Standard Deviation)			Wilcoxon Test	
		Jaya	MRLDE	MRL-Jaya	3/1	3/2
F1	1500	2.06E-16 -7.87E-16	1.37E-42 -1.56E-42	1.33E-47 -6.81E-47	+	+
F2	1500	5.23E-08 -5.37E-08	4.66E-21 -3.20E-21	7.48E-23 -2.15E-23	+	+
F3	3000	2.86E-05 -1.44E-05	1.72E-21 -1.03E-21	1.11E-22 -2.36E-22	+	+
F4	3000	5.74E-05 -1.11E-05	1.96E-15 -1.37E-15	7.71E-22 -7.37E-22	+	+
F5	2000	6.05E+00 -1.16E-01	8.90E-18 -3.87E-18	2.71E-23 -2.36E-23	+	+
F6	1500	0.00E+00 0.00E+00	0.00E+00 0.00E+00	0.00E+00 0.00E+00	=	=
F7	3000	5.51E-03 -4.91E-03	2.03E-03 -7.45E-05	1.50E-03 -9.93E-05	+	=
F8	3000	1.01E-03 -8.21E-03	1.01E-03 -2.40E-03	1.01E-03 -1.42E-03	=	=
F9	3000	8.20E-17 -2.31E-17	9.01E-01 -2.30E-01	3.10E-07 -5.60E-07	-	+
F10	1500	4.67E-09 -4.77E-09	3.40E-15 0.00E+00	4.15E-16 0.00E+00	+	+
F11	1000	5.96E-09 -2.07E-09	0.00E+00 0.00E+00	0.00E+00 0.00E+00	+	=
F12	1000	1.02E-10 -6.69E-10	1.35E-19 0.00E+00	0.00E+00 0.00E+00	+	+
F13	1000	1.19E-09 -9.96E-10	1.29E-19 0.00E+00	0.00E+00 0.00E+00	+	+
SF1	500	3.08E-02 -3.56E-02	5.68E-11 -1.21E-11	1.70E-13 -2.32E-13	+	+
SF2	1500	4.22E-01 -1.06E-01	4.45E-07 -8.27E-07	9.76E-08 -8.78E-08	+	+
SF3	1500	1.23E+00 -3.21E+00	4.34E-06 -6.64E-06	1.81E-11 -6.34E-11	+	+
SF4	1500	3.73E+02 -3.12E+01	1.38E+02 -2.80E+01	1.85E+01 -4.45E+00	+	+
SF5	500	5.33E-03 (6.28E-03	1.17E-09 -1.45E-09	1.70E-13 -1.67E-14	+	+
SF6	500	4.04E-02 (2.17E-02	2.85E-06 -2.24E-06	2.85E-08 -2.24E-08	+	+

'+','=' and '-' indicate best, equal and worst performance respectively.

Table 4. Results and Comparisons in term of average NFE and Acceleration Rate

Fun	VTR	Mean NFE			AR	
		Jaya	MRLDE	MRLJaya	MRL-Jaya vs/Jaya	MRL-Jaya vs/MRLDE
F1	10^{-08}	982100	42000	**38300**	96.10	8.81
F2	10^{-08}	182000	68000	**64800**	64.40	4.71
F3	10^{-05}	265000	118000	**104000**	60.75	11.86
F4	10^{-05}	188000	114000	**94000**	50.00	17.54
F5	10^{-08}	NA	148000	**127100**	NA	14.12
F6	10^{-08}	22000	14000	**12400**	43.64	11.43
F7	10^{-02}	152000	68000	**46500**	69.41	31.62
F8	10^{-03}	106000	112000	**96000**	9.43	14.29
F9	10^{-08}	**264000**	NA	322000	$\mathbf{-21.97}$	NA
F10	10^{-08}	144000	62000	**59900**	58.40	3.39
F11	10^{-08}	100000	44000	**40900**	59.10	7.05
F12	10^{-08}	84500	36000	**34700**	58.93	3.61
F13	10^{-08}	96000	41000	**36700**	61.77	10.49
SF1	10^{-08}	81000	43000	**38000**	53.09	11.63
SF2	10^{-06}	NA	122000	**105000**	NA	13.93
SF3	10^{-06}	NA	140000	**135000**	NA	3.57
SF4	10^{-08}	NA	NA	NA	NA	Na
SF5	10^{-08}	83000	47000	**38000**	54.22	19.15
SF6	10^{-06}	128000	65000	**55000**	57.03	15.38
				Average AR	**51.62**	**11.92**

Table-4 demonstrates the numerical results in term of average NFE and acceleration rate (AR) of 100 runs. Here it can be observed that proposed MRL-Jaya takes fewer NFE to attain the set error for each function except for function $F9$ except $F9$ where Jaya takes less NFE than all others. All algorithms are unable to reach the fixed error in case of function $SF4$. By acceleration rate (AR), we can also verify the fast convergence speed of proposed algorithm compare to both Jaya and $MRLDE$ for each function (except $F9$ and $SF4$). It is also observed that MRL-Jaya improves the average convergence speed by 51.62% and 11.92% compare to Jaya and $MRLDE$ respectively.

4.5 Results and Comparison of MRL-Jaya with DE, $ODEjDE$, $LeDE$:

In this section comparison of proposed MRL-Jaya with DE and its other modified variants ODE [2], jDE [3] and LeDE [6] has been carried out. The numerical results have taken in term of average-NFE of 100 runs by fixing error as 10 – 08 for all function except function $F7$ for which error is taken as 10 – 02. Parameter settings and numerical results for ODE, jDE and LeDE are taken from [6]. From the Table-5, it can be easily observed that proposed MRL-Jaya attain desire results rapidly compare to all other algorithm for all function except$F3$, $F8$ and $F9$. LeDE gives best performance for F3 while jDE gives perform superior for F8 and F9. The rank-wise performance of all algorithms is also given in Table-5 for each function. The average rank of LeDE, jDE, ODE and DE are 1.92, 2.92, 3.85 and 4.77 whereas the average rank of MRL-Jaya is 1.54 which proved the superiority and robustness in performance of proposed MRL-Jaya on others.

Table 5. Comparison of MRL-Jaya with DE, ODE, LeDE and SaDE in term of average NFE

Fun	Average NFE					Rank				
	DE	ODE	jDE	LeDE	MRL-Jaya	DE	OD **E**	jDE	LeDE	MRL -Jaya
F1	104000	67,524	60000	49,494	**38300**	5	4	3	2	1
F2	174000	140,170	83000	77,464	**64800**	5	4	3	2	1
F3	422000	489,210	340000	**140, 176**	146000	4	5	3	1	2
F4	NA	145,880	300000	157,499	**132100**	5	3	4	2	**1**
F5	424000	NA	580000	282,972	**127100**	3	5	4	2	**1**
F6	36000	25,008	23000	17,123	**12400**	5	4	3	2	**1**
F7	90500	60,230	100000	33,302	**46500**	5	3	4	2	**1**
F8	NA	147,472	**89000**	111,013	488000	5	3	**1**	2	4
F9	NA	190,604	**120000**	187,813	322000	5	3	1	2	4
F10	165000	10,6694	91000	76,111	**59900**	5	4	3	2	**1**
F11	112000	79,888	63000	50,579	**40900**	5	4	3	2	**1**
F12	96000	63,710	55000	41,384	**34700**	5	4	3	2	**1**
F13	106000	63,202	60000	46,529	**36700**	5	4	3	2	**1**

4.6 Convergence Graph:

In this section, the convergence graphs of Jaya, MRLDE and proposed MRL-Jaya is presented for function $F1$, $F4$, $F5$ and $F10$. The convergence speed of MRL-Jaya can be easily analyzed by these graphs over Jaya and MRLDE.

5 Conclusion:

In this paper, a new algorithm 'MRJ-Jaya' by fusion of MRLDE and Jaya algorithm is proposed for solving global optimization problems. The combination of two algorithms is taken in a systematic way so that the advantage of both algorithms can be utilized to improve the search ability with a high convergence speed. The proposed MRL-Jaya has been compared with its parent algorithms i.e Jaya and MRLDE in term of average error and average NFE on 13 non-shifted and 6 shifted unconstrained benchmark functions. A non-parametric Wilcoxon statistical test is also performed to analyze the comparison. Furthermore MRL-Jaya is also compared with DE and other DE variants: ODE, jDE and LeDE on 13 traditional unconstrained benchmark functions in term of average NFE and acceleration rate. The results and comparisons have revealed that the MRL-Jaya perform superior compared to other algorithms in terms of search process efficiency, solution quality and the convergence speed.

References

[1] R. Storn, K. Price, Differential evolution–a simple and efficient adaptive scheme for global optimization over continuous, Spaces. Berkeley, CA, Tech. Rep. TR-95-012

[2] J Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, *IEEE Transaction of Evolutionary Computing* **10**, 646–657 (2006).

[3] S. Rahnamayan, H. Tizhoosh, M. Salama, Opposition based differential evolution *IEEE Transaction of Evolutionary Computing* **12**, 64–79 (2008).

[4] A.K Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization *IEEE Transaction of Evolutionary Computing* **13**, 398–417 (2009).

[5] J. Zhang, A. Sanderson, JADE: adaptive differential evolution with optional external archive, *IEEE Transaction of Evolutionary Computing* **13**, 945–958 (2009).

[6] Y. Cai, J. Wang, J. Yin, Learning enhanced differential evolution for numerical optimization, *Springer-Verlag, Soft Computing,* doi:10.1007/s00500-011-0744-x (2011).

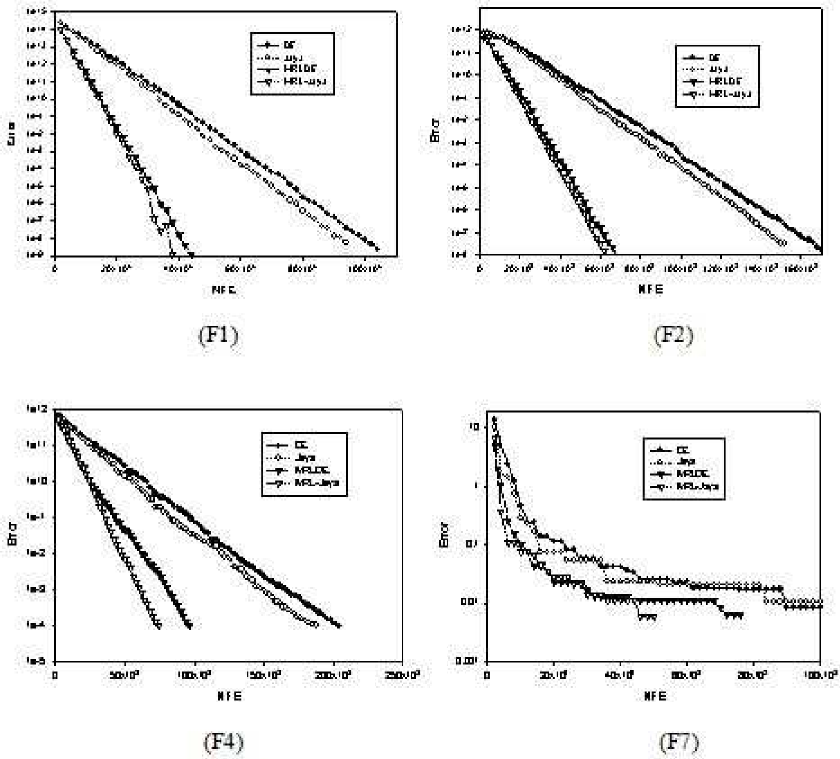

Figure-1: Convergence Graphs of function F1, F2, F3 and F7

[7] W. Zhu, Y. Tang, J.-A. Fang, and W. Zhang, Adaptive population tuning scheme for differential evolution, *Information Sciences* **223**, 164–191 (2013).

[8] M. Asafuddoula, T. Ray, and R. Sarker, An adaptive hybrid differential evolution algorithm for single objective optimization, *Applied Mathematics and Computation* **231**, 601–618 (2014).

[9] W. L. Xiang, X. L. Meng, M. Q. An, Y. Z. Li, and M. X. Gao, An enhanced differential evolution algorithm based on multiple mutation strategies, *Computational Intelligence and Neuroscience* **2015**, Article ID 285730, 15 pages (2015).

[10] L. Cui, G. Li, Q. Lin, J. Chen, and N. Lu, Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations, *Computers & Operations Research* **67**, 155–173 (2016).

[11] G. Wu, R. Mallipeddi, P. N. Suganthan, R. Wang, and H. Chen, Differential evolution with multi-population based ensemble of mutation strategies, *Information Sciences* **329**, 329–345 (2016).

[12] L. M. Zheng, S. X. Zhang, K. S. Tang, and S. Y. Zheng, Differential evolution powered by collective information, *Information Sciences* **399**, 13–29 (2017).

[13] Z. Meng, J.-S. Pan, and L. Kong, Parameters with adaptive learning mechanism (palm) for the enhancement of differential evolution, *Knowledge-Based Systems* **141**, 92–112 (2018).

[14] P. Singh, P. Chaturvedi, P. Kumar, A novel differential evolution approach for constraint optimization, *International Journal of Bio-Inspired Computation* **12**, 254–265 (2018).

[15] Z. Meng, J.-S. Pan, and K.-K. Tseng, PaDE: an enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization, *Knowledge-Based Systems* **168**, 80–99 (2019).

[16] Z. Wei, X. Xie, T. Bao, and Y. Yu, A random perturbation modified differential evolution algorithm for unconstrained optimization problems, *Soft Computing* **23**, 6307–6321 (2019).

[17] M. Duan, H. Yang, H. Liu, and J. Chen, A differential evolution algorithm with dual preferred learning mutation, *Applied Intelligence* **10**, 605–627 (2019).

[18] M. Tian and X. Gao, Differential evolution with neighborhood-based adaptive evolution mechanism for numerical optimization, *Information Science* **478**, 422–448 (2019).

[19] S. H. Wang, Y. Z. Li, and H. Y. Yang, Self-adaptive mutation differential evolution algorithm based on particle swarm optimization, *Applied Soft Computing* **81**, (2006).

[20] J. S. Pan, C. Yang, F. J. Meng, Y. X. Chen, and Z. Y. Meng, A parameter adaptive DE algorithm on real-parameter optimization, *Journal of Intelligent and Fuzzy Systems* **38**, 1–12 (2020).

[21] M. Di Carlo, M. Vasile, and E. Minisci, Adaptive multipopulation inflationary differential evolution, *Soft Computing* **24**, 3861–3891 (2020).

[22] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, *Artif Intell Rev* **33**, 61–106 (2010).

[23] S. Das., P.N. Suganthan, "Differential evolution: a survey of the state-of-the-art *IEEE Transaction of Evolutionary Computing* **15**, 4–13 (2011).

[24] P. M. Bilal, M. Pant, H. Zaheer, L. Garcia-Hernandez, and A. Abraham, "Differential evolution: a review of more than two decades of research, *Engineering Applications of Artificial Intelligence* **90**, 103479 (2020).

[25] P. Kumar, M. Pant, Enhanced mutation strategy for differential evolution, *Proceeding of IEEE Congress on Evolutionary Computation (CEC-12)* 1–6 (2012).

[26] S. Kumar, P. Kumar, T.K. Sharma, M. Pant, M., Bi-level thresholding using PSO, Artificial Bee Colony and MRLDE embedded with Otsu method, *Memetic Computing* **5**, 323–334 (2013).

[27] P. Kumar, M. Pant, V.P.Singh, Modified random localization based de for static economic power dispatch with generator constraints, *International Journal of Bio-Inspired Computation* **6**, 250–261 (2014).

[28] P. Kumar, D. Singh, S. Kumar, MRLDE for solving engineering optimization problems, *In proc of IEEE conference ICCCA-2015* DOI: 10.1109/CCAA.2015.7148512 (2015).

[29] P. Kumar, M. Pant, Recognition of noise source in multi sounds field by modified random localized based DE algorithm, *International Journal of System Assurance Engineering and Management, Springer* **9**, 245–261 (2016).

[30] R. V. Rao, Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems, *Int. J. Ind. Eng. Comput* **7**, 19–34 (2016).

[31] S Mishra and PK Ray, Power quality improvement using photovoltaic fed DSTATCOM based on Jaya optimization, *IEEE Trans. Sust. Energy* **99**, 1–9 (2016). [31] , Vol. 99, 2016, pp. 1-9.

[32] C. Gong, An Enhanced Jaya Algorithm with a Two Group Adaption, *International Journal of Computational Intelligence Systems* **10**, 1102–1115 (2017).

[33] K. Yu, J. Liang, B. Qu, X. Chen and H. Wang, Parameters identification of photovoltaic models using an improved JAYA optimization algorithm, *Energy Conversion and Management* **150**, 742–753 (2017).

[34] K. Gao, Y. Zhang, A. Sadollah, A. Lentzakis and R. Su, Jaya harmony search and water cycle algorithms for solving large-scale real-life urban traffic light scheduling problem, *Swarm and Evolutionary Computation* **37**, 58–72 (2017).

[35] R. V. Rao and K. C. More, Design optimization and analysis of selected thermal devices using self-adaptive Jaya algorithm, *Energy Conversion and Management* **140**, 24–35 (2017).

[36] S. P. Singh, T. Prakash, V. Singh, and M. G. Babu, Analytic hierarchy process based automatic generation control of multi-area interconnected power system using jaya algorithm, *Engineering Applications of Artificial Intelligence* **60**, 35–44 (2017).

[37] R. V. Rao and A. Saroj, Economic optimization of shell-and-tube heat exchanger using jaya algorithm with maintenance consideration, *Swarm and Evolutionary Computation* **116**, 473–487 (2017).

[38] R. Venkata Rao and A. Saroj, A self-adaptive multi-population based jaya algorithm for engineering optimization, *Swarm and Evolutionary Computation* **37**, 1–37 (2017).

[39] R. V. Rao and A. Saroj, Multi-objective design optimization of heat exchangers using elitist-jaya algorithm, *Energy Systems* **9**, 305–341 (2018).

[40] J.-T. Yu, C.-H. Kim, A. Wadood, T. Khurshaid, and S.-B. Rhee, Jaya algorithm with self-adaptive multi-population and lévy flights for solving economic load dispatch problems, *IEEE Access* **7**, 21372–21384 (2019).

[41] K. Tang, X. Yao, P.N. Suganthan, C. MacNish, Y.P. Chen, C.M. Chen, Z. Yang Z,Benchmark functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization, *Technical Report Nature Inspired Computation and Applications Laboratory* **37**, 58–72 (2007).

Author information

Pravesh Kumar, Department of Mathematics, Rajkiya Engineering College Bijnor, India.
E-mail: praveshtomariitr@gmail.com,

Amit Sharma, Department of Mathematics, Amity University Haryana, India.
E-mail: dba.amitsharma@gmail.com