International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 5 (2015) pp. 13579-13594

http://www.ripublication.com

Differential Evolution with Added Components for Early Detection and Avoidance of Premature Convergence in Solving Unconstrained Global Optimization Problems

Rachamalla Rahul Reddy[1], G.Jeyakumar[2]

[1]Final Year B.Tech CSE [2]Associate Professor
Department of Computer Science and Engineering
Amrita School of Engineering,
Amrita Vishwa VidyaPeetham, Coimbatore – 641 112. India.
[1]sunnyracha14@gmail.com [2]g_jeyakumar@cb.amrita.edu

Abstract

Differential Evolution (DE) is a recent addition to the repository of Evolutionary Algorithms (EAs) under Evolutionary Computing Techniques. As similar to other EAs, DE also used for optimization based on population of members. During population optimization the classical DE faces major problem of premature convergence, which causes the sample members to converge early to a local optimum though there is a global optimum in the search space. This paper presents methods to reduce the effect premature convergence and achieve better optimal solutions. There are few works in the literature for same reason by altering the control parameters of DE viz., scaling factor (F) and crossover rate (CR). However, we propose methods to make suitable amendments in the population level directly after detecting premature convergence during the search of DE. These methods can be added as an additional component to DE algorithm. The proposed components are to replace the population members with distinct highest objective function values with random members, to replace the population members with distinct lowest objective function values with random members, to replace members in random fashion and to increase the population size dynamically to counter the early convergence of the population. The above mentioned techniques are implemented and added with classical DE. The performance efficacy of DE with added components is verified on implementing it over a set of Benchmarking Function Suite with functions of different characteristics. The experimental results proved that DE with above components added is able to achieve better optimum values than the classical DE.

Keywords: Differential Evolution, premature convergence, replacing population members, objective function values, suboptimal solutions

1. Introduction

Differential Evolution (*DE*) is one of the recent additions in Evolutionary Algorithm (*EA*) employed to optimize real parameter, real valued functions using stochastic, population-based optimization algorithm, proposed by Storn and Price in 1995 (Stron and Price, 1995). The underlying idea behind *DE* is to generate mutated parameter vectors by adding the weighted difference vector to another sample vector in the population. It is followed by a recombination operation between the mutant vector and the target vector, which produces the trail vector. Through selection, the fitness value of the resultant trial vector is compared with the fitness value of the parent vector. The member with the good objective function value is transferred to the next generation. *DE* in-turn consists of many variants based on multiple ways of mutation and recombination. Some of the variants result in suboptimal solutions caused by premature convergence. The premature convergence can occur due to loss of population diversity and it prevents the algorithm from reaching global optimum. This paper presents a method to detect the premature convergence early than it happen and also propose four methods to achieve better optimal solutions when premature convergence happens. The paper is organized as follows. Section 2 describes related papers briefly followed by the details of design of experiment in Section 3. Section 4 mentions the experimental results and inferences and finally Section 5 concludes the work.

2. Related Works

Different authors developed various techniques to overcome the problem of suboptimal solutions. Following is a brief mention about some methods proposed by different authors.

Jouni Lampinen and Ivan Zelinka presented (Lampienen and Zelinka, 2000) the possible reasons for the premature convergence and stagnation problems of Differential Evolution. One of the reasons for the premature convergence to occur is losing the diversity in the population (search space). The techniques of adaptive evolutionary algorithms and hybrid evolutionary algorithms (Grosan and Abraham, 2007) have been implemented for *DE* (Vanaret et al, 2013) to avoid the effect of premature convergence. As an attempt, hybridizing *DE* with the popular branch-and-bound technique (as a local search guide) was experimented in (Vanaret et al, 2013), and proved the efficiency of the hybrid *DE* algorithm. Lu et al., proposed adaptive hybrid differential evolution algorithm (*AHDE*) integrated with a local random search operator (*LRS*) (Lu et al, 2010) to achieve near global solution. This combination of *DE* and local random search could balance the global exploration and local exploitation.

Understanding the search behavior of *DE*, in maintaining population diversity, during its evolution is a possible scope to find some strategies to avoid the problem of premature convergence and stagnation. Various theoretical attempts have been made in the literature to understand the search behavior of the *DE* algorithm by measuring the population diversity during the evolution. A theoretical expression to measure the population diversity from generation to generation for a *DE* variant *DE/rand/1/bin* was derived by Zaharie in (Zaharie, 2001). Later, this expression was extended to few more variants of *DE* in (Jeyakumar and Shunmuga Velayutham, 2010) and (Thangavelu et al, 2015). However, all these theoretical studies did not comment on the solutions to premature convergence problem.

There are numerous works found in the literature to improve the efficiency of *DE* algorithm by the virtue of avoiding premature convergence and stagnation problem. Few interesting works are presented below.

Sá et al., proposed a modification to the standard mechanisms in *DE* algorithm by introducing a probability selection of a new member (Sa et al, 2008). Though the probability of success shown by the modified *DE* is higher it relaxes the greedy selection scheme of the *DE* algorithm. Little selective pressure prohibits the *DE* to converge to an optimum in a reasonable time.

Josef Tvrdík adopted competitive variants of *DE* such as *DEBR18* which performed apparently better than standard *DER* (Tvrdik, 2006). This competition revolves around 18 fixed settings of *F* and *C* contributing to no further change. Moreover, variant *DERADP3* using *F* adaptive search had less reliability.

Rahnamayan et al., proposed Opposition-Based Learning (*OBL*) (Rahnamayan et al, 2008) to improve the convergence rate of *DE*. It depends on a new control parameter, the jumping rate to force to jump on a new solution candidate. This method performs better over high-dimensional problems with large population size but still high jumping rates are not recommended to achieve optimum solution.

Mallipeddi et al., suggested an ensemble of mutation strategies and parameter values for *DE* (*EPSDE*) (Mallipeddi et al, 2011). It consists of a pool of diverse characterized mutation strategies along with a pool of values for each of the associated control parameters. *EPSDE* focuses on employing different mutation strategies with different parameter settings during different strategies of evolution.

Zheng et al., developed behavioral metrics to investigate *DE's* search behavior as a function of parameter values (Zheng et al, 2014). The measured statistics serve as the guidance for parameter-tuning and appropriate take-up of *DE* to improve performance.

Instead of controlling and adjusting the parameters either dynamically or statically, this paper in contrast to other works presents the techniques presented in Table 1 to improve the solutions at the situation of premature convergence. Rather than altering the parameters we propose methods to directly control the population members of the search space and improve the suboptimal solutions.

When premature convergence happens the suboptimal solution produced remains constant in all the subsequent generations. Hence, a threshold value *t* is fixed such that whenever the objective function value remains the same for at least *t* (taken as 50 after considering various samples) generations then one of the methods described in Table 1 is performed to recover the population from premature convergence or to produce better optimal solutions.

Method 1 suggests that, on the detection of premature convergence, we identify three population members in the current generation with good objective function values on condition that they are distinct from each other. We then replace these members with random population members and are carried forward to the next generation along with the existing members. We chose members with distinct objective function values to maintain the diversity in the population. If the diversity is lost at an early stage then the algorithm converges quickly to a suboptimal solution Thus the Method 1 makes an alternation in the population with the aim of altering the search direction of *DE* to escape from the local optimum.

Table 1. List of methods to achieve better optimal solutions during premature convergence

Method No.	Method
1	To replace three population members of distinct highest objective function values
2	To replace three population members of distinct lowest objective function values
3	To replace three population members randomly in the sample
4	To increase the population size once, during a run

Similarly to Method 1, Method 2 also updates the population. However, by replacing three population members with distinct worst objective function values by three new random members. The Method 3 replaces any three random members in the population with any other random vectors in the current generation.

On contrary to Method 1, 2 and 3 which are altering the population by replacing selected candidates by new random candidates, the Method 4 increases the population size dynamically from 100 to 300 by adding random members to the existing population at once during the run on identifying the premature convergence.

3. Design of Experiments

The scaling factors (*F*), population size (*NP*) and crossover rate (*CR*) are the essential control parameters of *DE* algorithm. The population size (*NP*) is fixed as 100 for all proposed methods except for method 4 where the population has been increased to 300 after the identification of premature convergence. The scaling factor (*F*) is confined to the interval [0.3, 0.9) by following (Jeyakumar and Shunmuga Velayutham, 2010b) and a new *F* value is generated for each generation. To experiment the nature of the proposed methods we have chosen a small benchmarking function suite of 5 functions with different optimization characteristics. The details of the benchmarking functions are presented in Table 2. Table 3 shows the crossover rate (*CR*) for all the 5 test functions, by following (Jeyakumar and Shunmuga Velayutham, 2010b). As *DE* is stochastic in nature 10 independent runs were performed with random population initialization for every run against each method listed in Table 1. A run will be stopped before the maximum number of generations is reached if the tolerance error of 1x 10^{-12} is obtained with respect to the global optimum. The maximum number of generations is kept as 1000 in all the runs.

Table 2. Description of Benchmarking Functions

***f₁* – Sphere model** $f_{sp}(x) = \sum_{i=1}^{n} x_i^2$ $-100 \le x_i \le 100; x^* = (0,0,\dots,0)$ $f_1(x^*) = 0$	***f₂* – Schwefel's Problem 1.2** $f_{sch2}(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$ $-100 \le x_i \le 100; x^* = (0,0,\dots,0);$ $f_2(x^*) = 0;$
***f₃* - Schwefel's Problem 2.21** $f_{sch3}(x) = max_i\{\lvert x_i\rvert, 1 \le i \le n\}$ $-100 \le x_i \le 100; x^* = (0,0,\dots,0); f_3(x^*) = 0;$	***f₄* - Generalized Rosenbrock's Function** $f_{gr}(x) = \sum_{i=1}^{n}\lvert 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\rvert$ $-30 \le x_i \le 30; x^* = (0,0,\dots,0);$ $f_4(x^*) = 0;$
***f₅* – Generalized Restrigin's Function** $f_{grf}(x) = \sum_{i=1}^{n}[x_i^2 - 10cos(2\pi x_i) + 10]$ $-5.12 \le x_i \le 5.12; x^* = (0,0,\dots,0); \quad f_5(x^*) = 0;$	

The functions f_1 to f_5 are grouped by the feature - unimodal separable, unimodal nonseparable, multimodal separable and multimodal nonseparable. The experiments are carried out by considering all the four proposed methods on a function from each of the features. Function f_1 and f_3 are selected from unimodal separable, f_2 is selected from unimodal nonseparable, f_5 is selected from multimodal separable and f_4 from multimodal nonseparable.

Table 3. CR values assigned to each test function

Function	CR
f_1	0.9
f_2	0.9
f_3	0.5
f_4	0.9
f_5	0.1

4. Results and Discussion

As an initial attempt, the superiority of the proposed methods is validated with two different variants of classical *DE* algorithms viz. *DE/best/1/exp* and *DE/best/2/exp*. The experimental results of implementing the identified variants with the proposed methods as added components over the chosen benchmarking problems are presented in Table 4 to 13 and Figures 1 to 4. The variants *DE/best/1/exp* is chosen for our experiment, since it was found to be often falling in premature convergence in our empirical analyis. The tables are using the notations OV_h, OV_l, OV_r, OV_p and OV_n and the Figures are using the notations *OV-h*, *OV-l*, *OV-r*, *OV-p* and *OV-n*. The meanings of the notations are described in Table 4.

Table 4. The notations used in the Tables and Graphs.

Notation in table	Notation in figure	Meaning
OV_h	*OV-h*	Objective function value when Method 1 is used
OV_l	*OV-l*	Objective function value when Method 2 is used
OV_r	*OV-r*	Objective function value when Method 3 is used
OV_p	*OV-p*	Objective function value when Method 4 is used
OV_n	*OV-n*	Objective function value when no recovery method is used

Generation wise results of the *DE* variant *DE*/best/1/exp on f_2 presented in Table 5 shows that objective function values obtained by dynamic increase in population size gives the best possible result of 1.69E-10 (near to the tolerance value of 1 x 10^{-12}) . There was a rapid jump in values from 400 to 600 generations. The next better result was given by the method of replacing population member with distinct good objective function values by random method. Method of replacing members with worst objective function values did not give promising results, comparatively. The results in Table 5 show an interesting anomaly in *DE* with OV_r method that from generations 900 to 2000 there are spikes in the convergence of objective function values. This is due to the random nature of the method as we are replacing existing members randomly with random members.

All the four methods are tested on f_2 for 10 independent runs. Table 6 shows the average objective function value for 10 runs and it is noticed from the results that method of dynamic increase in population size gives better result.

Table 5. Generation wise objective function values on *DE/best/1/exp* on function *f_2*

Generation	OV_h	OV_l	OV_r	OV_p	OV_n
1	135560	75499.58	85545.23	101639.70	115000
100	3526.31	3804.85	3199.15	3189.83	3290
200	1298.75	1878.48	1242.57	1655.87	599
300	820.73	1633.84	418.87	1163.99	534
400	372.77	1054.73	211.88	1118.40	455
500	39.19	270.65	136.99	51.4068	451
600	12.81	145.65	134.72	0.09	450
700	11.06	141.23	0.46	0.01	445
800	10.90	140.52	0.06	0.00	435
900	0.01	140.48	0.07	1.37E-05	434
1000	0.00	140.47	21.79	3.04E-7	416
1100	0.00	140.45	57.46	2.11E-8	416
1200	0.00	140.45	32.28	1.03E-09	416
1300	0.00	140.45	2.286	4.06E-10	416
1400	0.00	140.45	0.86	3.53E-10	416
1500	6.85E-5	140.45	0.48	3.52E-10	416
1600	6.51E-05	140.45	0.49	3.51E-10	416
1700	6.50E-5	140.45	14.61	3.35E-10	416
1800	6.50E-5	140.45	1.45	1.76E-10	416
1900	6.50E-5	140.45	0.47	1.69E-10	416
2000	6.50E-5	140.45	0.38	1.69E-10	416

Table 6. Experimental Results for *DE/best/1/exp* on function *f_2*

Run	Generation	OV_h	OV_l	OV_r	OV_p	OV_n
1	2000	0.00	140	0.38	1.69E-10	416
2	2000	160	5.72	16.6	3.73E-09	826
3	2000	38	3050	6.35	0.00	1.18
4	2000	0.07	205	3.18	6.5E-011	98
5	2000	7.69	1270	3.14	-	476
6	2000	0.00	143	1.01	-	1.60
7	2000	128	4.2	225	-	795
8	2000	28.3	189	0.90	-	3.36
9	2000	5.89	156	23.80	-	324
10	2000	20.2	23.3	61.20	-	19.1
Average		**38.81**	**519**	**34.16**	**0.00**	**296**

Figure 1 shows the convergence of objective function values of the variant *DE/best/1/exp* plotted for f_2 based on the results in Table 5. It compares the objective values achieved through 4 methods with the objective values attained when no recovery method is used. The graph shows that all the 4 proposed methods outperformed the solution obtained when no recovery method is used.

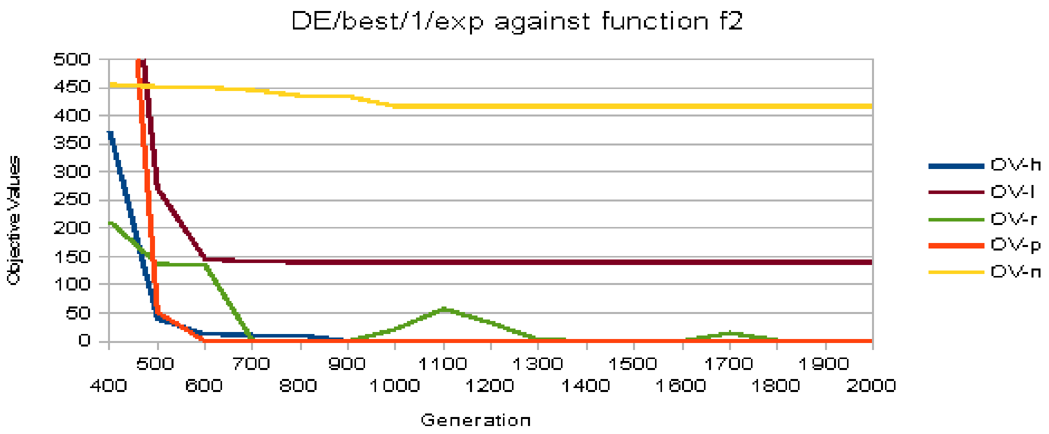

Figure 1. Convergence of Objective Function Values for *DE/best/1/exp* for function f_2

Table 7 shows the experimental results of implementing *DE/best/1/exp* on function f_4. The results show that the Method 5 has shown competitive results compared to other methods.

Table 7. Experimental Results for *DE/best/1/exp* on function f_4

Run	Generation	OV_h	OV_l	OV_r	OV_p	OV_n
1	2000	68.81	47300	47.10	0.01	633000
2	2000	27.82	3080	111	67.10	91100
3	2000	4570	505	160	3.99	49700
4	2000	170	285000	82.70	3.99	206000
5	2000	288	722000	74.9	5.73	213000
6	2000	71.63	538000	96.70	67	23800
7	2000	1770	25900	101	3.99	510000
8	2000	8.94	139000	14.40	28.10	449000
9	2000	438	247000	47.60	0.08	246000
10	2000	1420	81200	3460	-	208
Average		**883**	**208899**	**420**	**20**	**242181**

The objective function values measured generation wise is presented in Table 8. The results reiterate that the method of increasing population size gives better result than other methods. In the search of optimum value with dynamic population increase method, a faster jump in objective functions value between the generations 600 and 700 can be noticed.

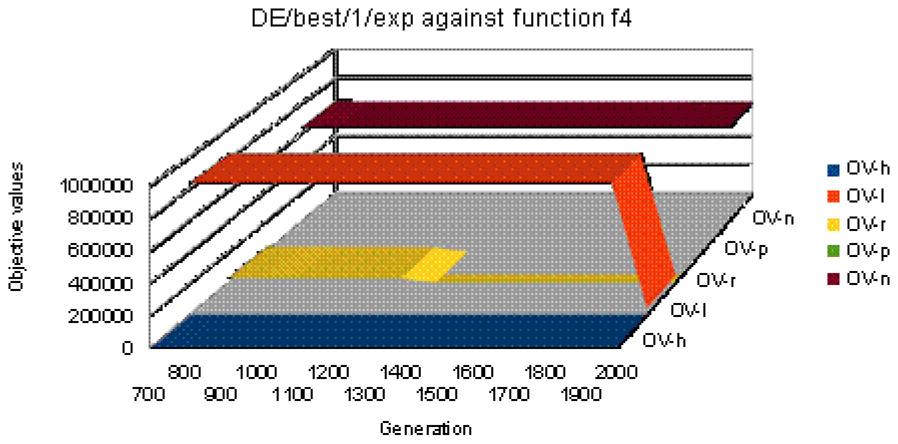

Figure 2. Convergence of Objective Function Values for *DE/best/1/exp* for function f_4

Table 8. Generation wise objective function values on *DE/best/1/exp* against function f_4

Generation	OV_h	OV_l	OV_r	OV_p	OV_n
1	2.2E+08	1.8E+08	1.1E+08	2.2E+08	2.7E+08
100	601577	2061901	323997	124981	1410000
200	392794	844576	51478.61	74640.15	975000
300	390124	831121	33515.48	65694.79	954000
400	364446	830778	33176.26	65126.99	640000
500	364379	830679	32965.68	64811.26	634000
600	2210.83	830655	32947.26	64728.89	634000
700	1875.14	830653	32930.37	13.92	640000
800	104.57	830652	32926.60	9.97	634000
900	98.23	830652	30441.23	5.58	634000
1000	93.88	830640	30439.20	2.62	640000
1100	90.79	830576	30436.99	0.47	634000
1200	88.35	830575	30273.37	0.05	634000
1300	85.84	830575	365.20	0.02	640000
1400	84.27	830575	285.48	0.01	634000
1500	70.74	830575	58.39	0.01	634000
1600	69.34	830575	53.16	0.01	640000
1700	68.94	830575	48.86	0.01	634000
1800	68.83	830575	48.82	0.01	634000
1900	68.80	830571	49.24	0.01	633000
2000	68.78	47258.6	47.13	0.01	633000

Figure 2 plots the objective function value convergence for the variant *DE/best/1/exp* on f_4. It is observed from Table 8 and Figure 2 that, with Method 2, the objective function values from generation 1200 stood constant. But after the generation 1900 there was a sudden decrease in the solution. It is also worth noticing that, while all other methods achieved better results, the one with no methods to recover is locked at a suboptimal solution of 633000.

The Table 9 presents the results recorded for *DE/best/1/exp* on f_5. Even though the performance differences between the methods are not significant in this case, *DE* with Method 1 shows better result comparatively. Table 10 shows the objective function value recorded generation wise. The results show negligible differences among the methods. Only the Method 1 and Method 4 are showing better results than the classical *DE*.

Table 9. Experimental Results for *DE/best/1/exp* on function f_5

Run	Generation	OV_h	OV_l	OV_r	OV_p	OV_n
1	2000	329	394	387	322	375
2	2000	328	364	402	357	345
3	2000	329	372	378	340	340
4	2000	313	380	366	324	344
5	2000	318	404	390	365	366
6	2000	308	403	350	371	350
7	2000	276	361	406	300	382
8	2000	344	368	395	348	387
9	2000	334	341	383	358	383
10	2000	323	360	385	375	295
Average		**320**	**375**	**384**	**346**	**357**

The result in Table 10 is plotted as convergence graph in Figure 3. The graph shows the variant *DE/best/1/exp* on f_5 shows nearly same results on all the methods. However, Method 1 and Method 4 performed better right from the early generations. An interesting anomaly observed in Figure 3 is that the method of random replacement started fluctuating from the initial generations and reached it's best value of 336.47 at generation 800 and continued to show irregular behavior.

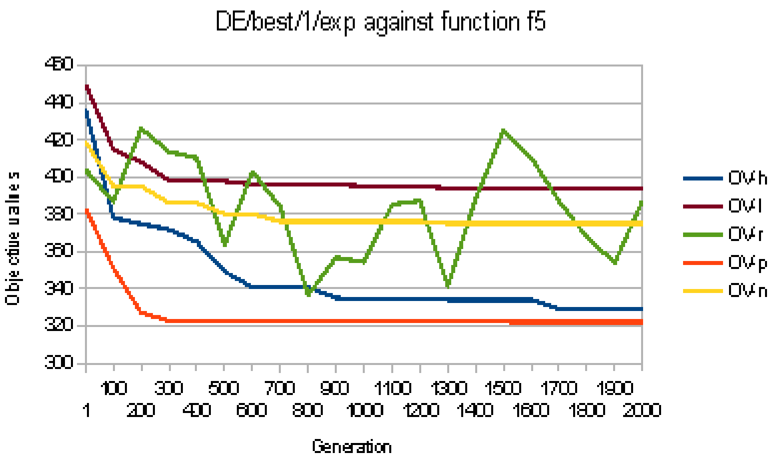

Figure 3. Convergence of Objective Function Values for *DE/best/1/exp* for function f_5

Table 10. Generation wise objective function values on *DE/best/1/exp* against function f_5

Generation	OV_h	OV_l	OV_r	OV_p	OV_n
1	436.37	450.04	403.87	383.34	419
100	378.06	414.80	386.64	351.16	395
200	374.66	407.91	426.02	327.10	395
300	371.66	398.20	413.49	322.82	386
400	365.41	397.96	410.19	322.69	386
500	349.29	397.69	363.68	322.69	380
600	340.80	396.05	402.88	322.69	380
700	340.80	395.96	383.77	322.69	376
800	340.80	395.95	336.47	322.69	376
900	335.16	395.95	356.90	322.54	376
1000	334.17	395.05	354.46	322.54	376
1100	334.17	395.05	385.29	322.54	376
1200	334.17	395.05	387.03	322.54	376
1300	334.17	393.90	341.42	322.54	375
1400	334.13	393.90	388.09	322.37	375
1500	334.13	393.90	425.17	322.37	375
1600	334.13	393.62	410.28	322.04	375
1700	329.18	393.62	386.90	322.04	375
1800	329.18	393.62	367.48	322.04	375
1900	329.18	393.54	354.25	322.04	375
2000	329.07	393.54	387.20	322.04	375

The experimental result achieved by *DE/best/1/bin* variant on the f_1 is presented in Table 11. It is noticed from the results that the Method 2 produced significantly better results, with average objective function value of 0.00. Along with Method 2, Method 4 also could achieve better result than the classical *DE*. Considering the independent runs of Method 1 in Table 11, the runs 3, 4, 5 and 6 showed significant performance differences than other runs.

A close generation wise look up at run 1 can be observed in Table 12 where all the methods performed better than the one with no recovery solution. Spikes are observed between generations 1500 to 1700 under Method 3. However, Method 2 performed better than the rest under the variant *DE/best/1/bin* against f_1.

When *DE/best/1/exp* against function f_1 is considered, Method 4 performed significantly better than classical *DE*. This is followed by the Method 1 with average objective function value of 4.31 and best objective function value of 5.88E-008 at run 3 as shown Table 13.*DE/best/1/exp* against function f_3 also produced a similar result (Table 14) as the variant *DE/best/1/exp* against f_5 (Table 9).

Table 11. Experimental Results for *DE/best/1/bin* on function f_1

Run	Generation	OV_h	OV_l	OV_r	OV_p	OV_n
1	2000	200	0.00	263	230	1460
2	2000	826	0.00	80.4	199	1170
3	2000	120000	0.00	46600	148	1940
4	2000	1690	0.00	465	282	1510
5	2000	117000	0.00	130	94.41	1070
6	2000	100000	0.00	182	98.72	2860
7	2000	527	0.00	446	534	1070
8	2000	504	0.00	58100	376	3290
9	2000	644	0.00	75.2	795	997
10	2000	47	0.00	28.5	25.53	963
Average		**34143.8**	**0.00**	**10637**	**278**	**1633**

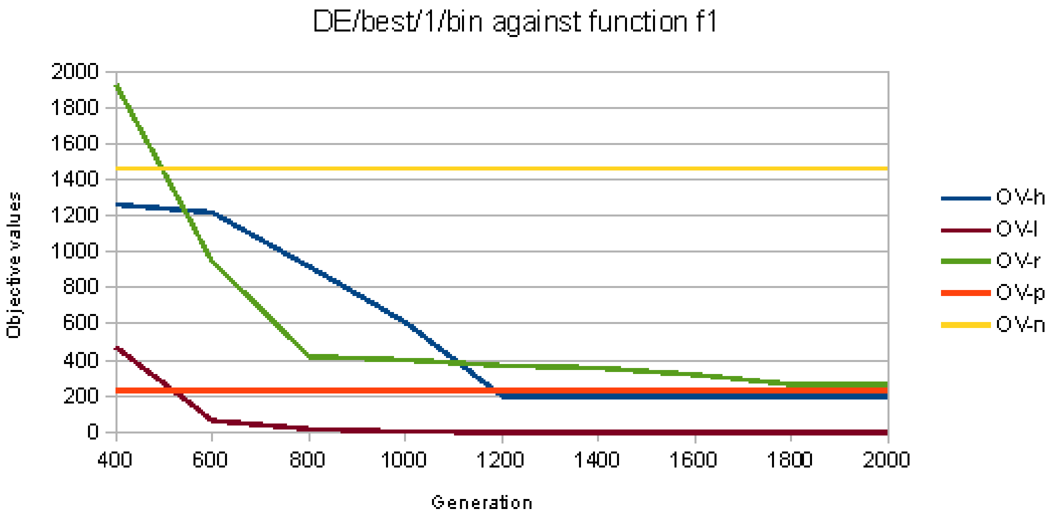

Figure 4. Convergence of Objective Function Values for *DE/best/1/bin* for function f_1

Table 12. Generation wise objective function values on *DE/best/1/bin* against function f_1

Generation	OV_h	OV_l	OV_r	OV_p	OV_n
1	44618.23	57154.15	62933.82	50657.66	51400
100	1618.72	15098.86	2518.10	1354.31	1490
200	1558.29	5829.11	2301.51	1334.94	1480
300	1507.07	1585.43	2016.66	1328.71	1470
400	1262.32	472.57	1926.38	231.67	1460
500	1256.21	249.21	1154.78	230.65	1460
600	1217.59	63.71	942.78	230.63	1460
700	1151.37	25.93	517.02	230.34	1460
800	918.37	17.58	420.70	230.33	1460
900	626.83	6.10	409.61	230.33	1460
1000	608.49	2.98	401.73	230.17	1460
1100	247.75	0.83	396.81	230.17	1460
1200	200.24	0.33	370.25	230.17	1460
1300	200.24	0.14	357.18	230.17	1460
1400	200.23	0.04	357.18	230.17	1460
1500	200.23	0.02	2555.08	230.17	1460
1600	200.23	0.01	319.69	230.17	1460
1700	200.23	0.00	84484.45	230.17	1460
1800	200.23	0.00	264.27	230.17	1460
1900	200.23	0.00	263.39	230.17	1460
2000	200.23	0.00	263.18	230.17	1460

Figure 4 plots the convergence of objective function values of the variant *DE/best/1/bin* (using the results in Table 12) on function f_1. The graph shows that the all the four proposed methods when compared produce the better results than *DE* with no recovery method. We can observe that all proposed methods achieved lower objective function values while the *DE* with no recovery mechanism held in local optimum value at an objective function value of 1460.

Table 13. Experimental Results for *DE/best/1/exp* on function f_1

Run	Generation	OV_h	OV_l	OV_r	OV_p	OV_n
1	2000	0.00	259	22	0.00	162
2	2000	0.00	129	48200	0.00	20.1
3	2000	5.88E-008	1220	0.16	0.00	49.9
4	2000	0.00	251	120	0.00	2.01
5	2000	0.00	1000	0.09	0.00	543
6	2000	0.00	801	0.78	0.00	694
7	2000	43.11	78	11.53	0.00	147
8	2000	0.00	1.99	24.23	-	11
9	2000	0.00	477	166	-	56.1
10	2000	0.00	394	0.08	-	206
Average		**4.31**	**461.1**	**4854.48**	**0.00**	**189**

Table 14. Experimental Results for *DE/best/1/exp* on function f_3

Run	Generation	OV_h	OV_l	OV_r	OV_p	OV_n
1	2000	70.5	65.7	80.9	71.7	79.1
2	2000	60.7	65.7	70.1	71.3	78.8
3	2000	66	65.7	82.5	67.7	79.1
4	2000	60.8	65.7	83.5	70.3	79.1
5	2000	60.5	65.7	80.6	75.9	79.1
6	2000	67.3	65.7	84.4	72.6	79
7	2000	65	65.7	83.6	67.9	79.1
8	2000	61.4	73.6	79.4	66.2	79.1
9	2000	69.4	71.7	83.9	66.6	78.3
10	2000	73	73.8	82.6	72.5	67.8
Average		65.46	67.9	81.15	70.27	77.85

5. Conclusion

This paper proposed four population alteration methods. These methods were aimed to improve the suboptimal solutions caused by premature convergence against the conventional methods like changing the parameters either statically or dynamically. All these methods were tested on benchmarking function suite. It is observed that for a benchmark function and variant pair at least one of the methods gave a far better optimal value than the suboptimal solution produced when no recovery method is used.

Care should be taken in Method 3 to avoid spikes such that only new sample members with good fitness values replace the existing ones. In future, we propose to find a procedure that transfers the results after achieving the best possible objective value produced by Method 3 to other methods appropriately because random nature of Method 3 has good chances of keeping diversity alive.

We also admit that the samples presented in the paper were for few benchmarking functions on two variants. However, extending these methods to other functions and variants will provide a better insight. This will form a subset of our future work along with the possibility of categorizing the functions and variant pair in an appropriate way to analyze the results and common patterns when the four proposed methods are applied.

6. References

1. Ângela A. R. Sá, Adriano O. Andrade, Alcimar B. Soares and Slawomir J. Nasuto,"Exploration vs. Exploitation in Differential Evolution",Proc. AISB 2008 Symposium on Swarm Intelligence Algorithms and Applications,Vol. 11,pp. 57-63,2008. (Sa et al, 2008)
2. Charlie Vanaret, Jean-Baptiste Gotteland, Nicolas Durand, Jean-Marc Alliot."Preventing pre-mature convergence and proving the optimality in evolutionary algorithms", Proceeding -, 11th International Conference on Artificial Evolution,, France. pp 84-94., 2013. (Vanaret et al, 2013)
3. D.Zaharie, "On the Explorative Power of Differential Evolution Algorithms", *Proceeding - 3rd International Workshop "Symbolic and Numeric Algorithms of Scientific Computing*, Romania, 2001. (Zaharie, 2001)
4. Feifei Zheng, Aaron C. Zecchin, Angus R. Simpson,"Investigating the run-time searching behavior of the differential evolution algorithm applied to water distribution system optimization", Environmental Modelling & Software,Oct 2014. (Zheng et al 2014).
5. Grosan C., and Abraham, A, "Hybrid Evolutionary Algorithms: Methodologies, Architectures, and Reviews", *Studies in Computational Intelligence* (*SCI*), Vol., 75, pp. 1 – 17., 2007. (Grosan and Abraham, 2007)
6. Jeyakumar, G. and ShunmugaVelayutham, C. "A Comparative Study on Theoretical and Empirical Evolution of the Population Variance of the Differential Evolution Variants", *Lecture Notes in Computer Science* (*LNCS-6457*), Springer-Verlag, pp. 75-79, 2010. (Jeyakumar and Shunmuga Velayutham, 2010a)
7. Jeyakumar,G. and Shunmuga Velayutham, C, "An Empirical Performance Analysis of Differential Evolution Variants on Unconstrained Global Optimization Problems", International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM), Vol. 2, pp. 077-086, 2010. (Jeyakumar and Shunmuga Velayutham, 2010b)

8. Josef Tvrdík,"Differential Evolution: Competitive Setting of Control Parameters",Proc. International Multiconference on Computer Science and Information Technology,pp. 207-213,2006. (Tvrdik, 2006)
9. Jouni Lampinen and Ivan Zelinka, "On Stagnation of the Differential Evolution Algorithm", *Proceedings- Mendel's 6th International Conference on Soft Computing,* (Ed.) P. Osmera., pp. 76 -83, 2000. (Lampienen and Zelinka, 2000)
10. R. Mallipeddi, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren,"Differential evolution algorithm with ensemble of parameters and mutation strategies",Applied Soft Computing, Vol. 11, Issue 2, pp. 1679-1696, March 2011. (Mallipeddi et al, 2011)
11. R.Storn and K.Price , " Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, Technical Report TR-95-012, ICSI March 1995. (Stron and Price, 1995)
12. Shahryar Rahnamayan, Hamid R. Tizhoosh, Magdy M. A. Salama,"Opposition-Based Differential Evolution",IEEE Transactions on Evolutionary Computation, Vol. 12, No. 1, Feb 2008. (Rahnamayan et al)
13. Thangavelu. S, Jeyakumar. G and Shunmuga Velayutham. C, "Theoretical Analysis of Expected Population Variance Evolution for a Differential Evolution Variant", *Computational Intelligence in Data Mining, Springer-Volume 2,* 403-416, January- 2015. (Thangavelu et al, 2015)
14. Youlin Lu, Jianzhong Zhou, Hui Qin, Yinghai Li, Yongchuan Zhang," An adaptive hybrid differential evolution algorithm for dynamic economic dispatch with valve-point effects",Expert Systems with Applications, Volume 37, Issue 7, pp. 4842–4849, July 2010. (Lu et al, 2010)