International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-7, Issue-6, March 2019

Crossover-Free Differential Evolution Algorithm to study the impact of Mutation Scale Factor Parameter

Dhanya M Dhanalakshmy, G. Jeyakumar, C. Shunmuga Velayutham

Abstract : The Differential Evolution (DE) algorithm, which is one of the popular optimization algorithms in the category of Evolutionary Algorithms (EAs), is known for its simplicity and wide applicability. Analysing and understanding the working nature of DE algorithm, for its further improvement, is an active research area in Evolutionary Computing (EC) field. In particular studying the role of its control parameters and their effects in its performance needs more attention. As an attempt in this direction, this paper presents evidences to showcase the role of the Scale Factor (F) parameter of DE algorithm through the plots generated based on the studies made from experimental results obtained through a well formulated experimental setup. The experimental set up includes five different benchmarking functions and a crossover-free DE algorithm, in which the crossover component is removed, for capturing better insights about the impact of F. The empirical evidences for the observed inferences are plotted as graphs.

Index Terms*: *Differential Evolution, Parameter Study, Mutation Scale Factor, Nature of Convergence, Premature Convergence, Successful Convergence and Stagnation

I. INTRODUCTION

The Evolutionary Computing (*EC*) field of Computer Science has many potential optimization algorithms in its pool for solving wide variety of real world optimization problems. All those algorithms follow a unified algorithmic structure designed by following Darwin's theory of evolution. This pool of algorithms is termed as Evolutionary Algorithms (*EAs*). The algorithmic structure of *EAs* includes candidate representation, population initialization, parent selection, mutation, crossover, survivor selection and termination criteria. The popularly known instances of *EAs* are Genetic Algorithm (*GA*), Evolutionary Strategy (*ES*), Evolutionary Programming (*EP*), Genetic Programming (*GP*) and Differential Evolution (*ES*). Each of these instances has proven their skill in different set of applications. As stated by 'No Free Lunch Theorem' [1] none of the above algorithm is good at solving all the type of optimization problem. Hence, finding suitable *EA* to solve the given optimization problem is yet another problem for the practitioner and researchers in *EC* community. In order to alleviate this problem proposing ideas for automatic selection of algorithm for the given problem, tuning/controlling the parameters of the algorithm automatically based on the nature of the problem and hybridising different algorithms are the active research areas in the field of *EC*. Among these the research works focusing on design and development of tuning-free *EAs* is attracting more researchers as there are many open research problems to solve. The design and development of tuning-free *EAs* includes theoretically analyzing, empirically validating, suitably depicting and comprehensively documenting the role of each of the inherent parameters of an *EA*. There are many research works devoted by researchers to cover the above said aspects of *EAs*.

In the pool of *EAs* the latest addition is *DE* [2, 3], which is most popular among researchers and industrialist for its simplicity in implementation. The *DE* algorithm has only three parameters viz Mutation Scale Factor (*F*), Crossover Rate (*CR*) and Population Size (*NP*). The performance of *DE* is largely depending on suitable setting of values for these parameters, either before start the run or at the run of the algorithm. The former one is known as parameter tuning and the later one is known as parameter control (or tuning-free parameters). Proposing different tuning-free *DE* is an active as well as challenging research area, as it proposes to eliminate parameter tuning by incorporating different parameter adaptation strategies at the execution of the algorithm. The tuning-free *DE* is supposed to self-adapt all its parameters without user choice and intervention, based on the given optimization problem. Many research works have been proposed to self-adapt one or more of *DE* parameters, but not all the parameters together. Self-adapting all the parameters need in-depth analysis on the role of each of the parameter and the dependency between them. The objective of this paper is to present a simple but comprehensive description on the impact of the Mutation Scale Factor (*F*) of *DE*.

The role of *F* is analyzed on a simple experimental set up considering different values of *F* and different benchmarking functions. This paper presents visual evidences to depict different situations that arise while solving different benchmarking functions using *DE* with different *F* values, with no crossover happening.

Revised Manuscript Received on 30 March 2019.
***** Correspondence Author
Dhanya M Dhanalakshmy*, Department of Computer Science and Engineering, Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, India.
G. Jeyakumar, Department of Computer Science and Engineering, Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, India.
C.Shunmuga Velayutham3, Department of Computer Science and Engineering, Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, India..

The details rendered in this paper would be a useful resource for the researchers working in design and development of different tuning-free *DE* strategies. The rest of the paper is organized as follows. The section 2 describes the algorithmic structure and nature of convergence of *Differential Evolution* algorithm. In Section 3, the works related to our work is summarized. The Section 4 presents the design of experiments which includes the benchmarking functions and the parameter set up. The results and discussions are given in Section 5 where the results are presented in both tabular and graph forms and finally Section 6 concludes the paper highlighting the inferences about role of *F* on the nature of *DE*.

II. CLASSICAL DIFFERENTIAL EVOLUTION ALGORITHM

The algorithmic structure of *DE* is shown in Figure 1. As shown in the Figure 1, the *DE* algorithm includes population initialization, mutation and crossover as its major components. The corresponding control parameters in these components are *NP*, *F* and *CR*, respectively. The other components are evaluation of the candidates, selection of survivor and checking for termination. The population size, *NP*, is the number of candidates in the population. It is generally a constant value in all the generations. The mutation scale factor, *F*, is used as the step size for mutation. The crossover rate, *CR*, is the probability of crossover used at crossover to decide the components for the offspring. Though, structurally, *DE* is similar to other *EA* it differs from them by its mutation process which is termed as *Differential Mutation*. Also, in the order of components the mutation is done before crossover while *DE* solves a problem.

The *Differential Mutation* generates one mutant vector (*mv*) for each of the current vector (*cv*) in the population. The *mv* is generated by adding the scaled (by *F*) difference(s) of one or two pair(s) of random vectors of the population with a base vector ($b_a v$). The $b_a v$ can be selected in different ways from the population. It can be a random vector (*rv*), best vector ($b_e v$), vector from the *cv* to a random vector, a vector from the *cv* to the best vector or a vector from a random vector to the best vector. The number of pairs chosen for the difference can be one or two pairs. The *rv* is a candidate selected randomly from the population at the current generation. The $b_e v$ is the candidate in the current population who has highest fitness according to the objective of the problem. Based on different $b_a v$ selection schemes and the number of pairs of vectors selected, the different mutation strategies available for *DE* are *rand/1*, *rand/2*, *best/1*, *best/2*, *current-to-rand/1*, *current-to-best/1* and *rand-to-best/1*.

The crossover component is to mix each *mv* with the corresponding *cv* to generate a trial vector (*tv*). The values to the components of the tv_s are inherited either from *mv* or *cv*, based on a probability. The probabilistic decision is taken by comparing the *CR* value with a random number. The most commonly used crossover schemes in *DE* are binomial crossover (*bin*) and exponential crossover (*exp*). Then a selection process which conducts a one-to-one competition between the *tv* and *cv* decides the survivor for next generation. The combination of seven different mutation schemes with two crossover schemes produces the popular *DE* variants [14] viz *DE/rand/1/bin*, *DE/rand/1/exp*, *DE/best/1/bin*, *DE/best/1/exp*, *DE/rand/2/bin*, *DE/rand/2/exp*, *DE/best/2/bin*, *DE/best/2/exp*, *DE/current-to-rand/1/bin*, *DE/current-to-rand/1/exp*, *DE/current-to-best/1/bin*, *DE/current-to-best/1/exp*, *DE/rand-to-best/1/bin* and *DE/rand-to-best/1/exp*.

1. Population Initialization
2. $X(0) \leftarrow \{x_1(0),.....x_{NP}(0)\}$
3. $g \leftarrow 0$
4. Compute $\{f(x_1(g)),...f(x_{NP}(g))\}$
5. while the stopping condition is false do
6. for $i = 1$ To NP do
   $y_i \leftarrow$ generatemutant($X(g)$)
   $z_i \leftarrow$ crossover($x_i(g)$, y_i)
   if $f(z_i) < f(x_i(g))$ then
   $x_i(g+1) \leftarrow z_i$
   else
   $x_i(g+1) \leftarrow x_i(g)$
   end if
   end for
7. $g \leftarrow g+1$
8. Compute $\{f(x_1(g)),\ldots,f(x_{NP}(g))\}$
9. End while

Figure 1. General Structure of *DE* Algorithm

Since the objective of this study is to focus on understanding the role of *F* which is the control parameter in *Differential Mutation* component of *DE*, the experiments conducted in this study use a unique variant of *DE* (crossover-free *DE*) named as *DE/rand/1/~*. The *DE/rand/1/~* does only the mutation for generating the offspring and switches off the crossover component. The offspring are directly compared with the corresponding cv_s by their fitness and the better offspring are replacing the cv_s for next generation.

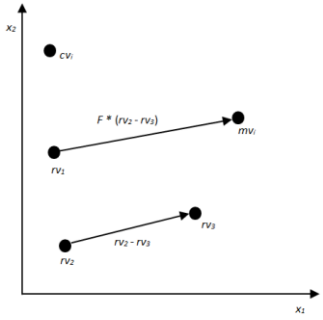
The *DE/rand/1/~* enables the experiment to analyze the changes in the population due to mutation process. The *DE/rand/1/~* uses a random vector as base vector and one pair of other random vectors. The *rand/1* mutation scheme is depicted in Figure 2. As shown in Figure 2, three random vectors (rv_1, rv_2 and rv_3) are selected from the population. The rv_1 is used as the base vector. The difference of rv_2 and rv_3 generates a difference vector. Then, the difference vector is scaled by the mutation scale factor (*F*) and added to the base vector to generate the mutation vector. The equation for *DE/rand/1* mutation, to generate the mutation vector (*mv*) for the i^{th} vector in the population, is shown in Equ (1).

$$mv_i = rv_1 + F * (rv_2 - rv_3) \quad (1)$$

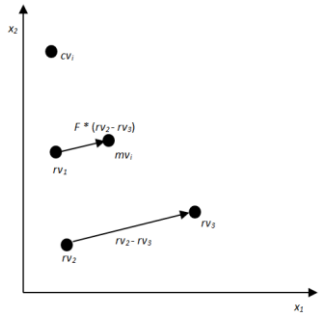
A. Role of *F* and Nature of Convergence of *DE*

As it is seen in Equation (1), *F* scales the difference vector add it to the base vector to produce the mutant vector. It is found in the literature that *F* takes different values. In general, the domain of *F* includes {0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}. These values are considered for the experimental studies in this paper.

The Figure 2 (a) and Figure 2(b) shows the difference between *F* with larger and smaller values, respectively. Since the distance ($rv_2 - rv_3$) is scaled by *F*, the value of *F* decides how far the mv_i would be from the rv_1 (base vector). For smaller values of *F*, mv_i is closer to the base vector and for larger *F* value it is away from the base vector. This means that for smaller values of *F* the offspring are similar to the base vectors, as they are closer to the base vector in the solution space. For larger values of *F*, the offspring are diversified from the base vector, as they are at far distance from the base vector in the solution space. In the former case the exploration of the solution space is less and in the later one it is high. Thus the values of *F* largely affect the population and the performance of *DE*. These facts can be analyzed further and validated by relating them to the nature of convergence of *DE* algorithm for varying values of *F*.



(a)



(b)

Figure 2. Differential Mutation (a) with larger *F* value and (b) with smaller *F* value

DE starts its search for the optimum solution(s) of the given problem with a set of initial solutions (known as population). Though there are many approaches possible to initialize the population, the most common approach followed is random initialization. This is to support the stochastic nature of *DE* algorithm. Starting with the initial population the search for global optimal solution may end either successful or unsuccessful. The successful cases are named as Successful Convergence (*SuC*), where the search is converging to an optimal solution. The unsuccessful cases, where the search is not converging to a solution, are name as Unsuccessful Convergence (*USuC*). The two most common reasons for *USuC* are Premature Convergence (*PrC*) and Stagnation (*Stag*).

The two active processes at *DE* searching for a solution are *exploration* and *exploitation*. The *exploration* process generates new candidate solutions by using the variation operators (mutation and crossover) and the exploitation process selects suitable candidates among the existing candidate (using selection) for the subsequent generations in search of the solution. At certain stage of evolutionary search when *exploitation* becomes dominant the search does not explore new solutions hence settle down in a local optimal solution. This is termed as *PrC*. On the other hand, if the search explores too much it produces more diversified solutions in the space and not converges to any of the region. This makes stagnation to occur and *DE* is not ending with any solution (either local or global optimal solution) before the stopping criteria is met. It is understood that if the exploration is less and exploitation is more the search ends with *PrC* and for the cases where exploration is dominant than exploitation the search ends with *Stag*.

The exploratory and exploitative nature of *DE* algorithm varies for each of its instance of run and hence it shows different nature of convergence for a same problem for varying values of control parameters. These possible cases of convergence are represented in Table 1, for various granularities of exploration and exploitation processes. It is worth understanding that the *SuC* is possible only if the exploration and exploitation processes are balanced. Any strategy used for designing tuning-free *DE* should consider all the above facts, for suitable adaptation of the control parameters. Designing a suitable algorithmic component to monitor the explorative and exploitative nature of *DE* algorithm at the time of solving a problem and reporting meaningful inferences to adapt the control parameters is yet another challenging and open research problem for the researchers working for tuning-free *DE* algorithms.

Table 1. Possible cases of *DE* search

Possible Cases	Exploration			Exploitation		
	Low	Medium	High	Low	Medium	High
Premature Convergence	Yes	-	-	-	-	Yes
Stagnation	-	-	Yes	Yes	-	-
Successful Convergence	-	Yes	-	-	Yes	-

Since the *DE* variant considered in this experiment (*DE/rand/1/~*) excludes crossover, the mutation takes over the charge of exploration. Variations to the population diversity are brought only through this *differential mutation* process. The only parameter available in mutation is *F*. Hence the value of *F* largely affects the explorative process of *DE/rand/1/~* and hence its nature of convergence. This setup enables the study to analyze the role of *F* in more valid manner. The experimental setup for this work reports the population level changes at various stages of search due to the mutation component and present the results with the support of visualization (as suggested in [20]).

III. RELATED WORKS

Since this paper focuses only on showing the role of *F* on *DE's* nature of convergence, this section highlights recent research works on parameter adaptations of *DE* algorithms and studies on analyzing impact of *F*.

A comprehensive summary of all the parameter adaptation techniques available in the literature for the *F* and *CR* is presented in [4, 5]. The authors have categorized the existing techniques in to different groups based on the method followed for the adaptation. Though numerous papers exist for parameter adaptation, the research works to discuss the role of each of the parameter is very less. There are only very few research works stating the impact of *F* on *DE*'s performance. They are described below.

Storn, in 1996, has suggested [6] that *F* can be chosen from [0.5, 1.0] and most of the researchers have followed this suggestion while developing adaptation strategies for *F*. Roger et al., in 2002, studied [7] the effect of *DE* parameters and gave some insights on setting the values for these parameters by considering four benchmark functions and four *DE* variants. The *Competitive DE* proposed by Josef Trvdik [8, 9] experimented with four variants of competitive *DE* for various *F* and *CR* combinations on six benchmark functions to find out which setup works better.

Ali and Torn, in 2004, have done [10] an empirical analysis to set minimum value for *F* in *DEPD* (a modified *DE* algorithm) and proposed that minimum value for *F* to be in [0.4, 0.5]. Brest et al. analyzed [11] the effect of *F* as well as *CR* for developing a self-adaptive *DE* and stated that best parameter setting is problem dependent in case of original *DE* and there are situations where *CR* influences *F* or vice versa. Lampinen and Liu had conducted experiments [12] to find how different *F* values in the range [0, 2] and *CR* values in the range [0, 1] affect the convergence to global optimum as well as maximum function evaluations.

Jeyakumar and Shunmuga Velayutham have used a bootstrap test [13, 14] proposed by Mezura et al. [15] for selecting value of appropriate *CR* for classical *DE* as well as Distributed *DE*. For each test problem considered in this study the *CR* values are tuned. The different values considered for *CR* are {0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}. 50 independent runs were performed for Function-*CR* value. From the results a bootstrap test was conducted to determine the confidence interval for the mean objective function value. The *CR* values corresponding to the best confidence interval were used for further experiments.

Akhila et al. have studied the impact of *F* [16] value on Population Diversity on four functions for 50 generations. Authors have implemented different methods of population diversity measurement and found from the results that the theoretical variances are increasing, decreasing or remains constant for varying values of *F*.

There are also many modified mutation strategies proposed in literature. A trigonometric mutation operator proposed by Vaishali et al [17] is an example. A detailed survey of different mutation strategies is presented by Karol Opara and Jarosaw Arabasb in [18], in a probabilistic perspective.

It is observed from the existing works that, though there exist different adaptation strategies for *F*, providing suitable evidences to highlight the role of *F* on *DE*'s nature of convergence will provide more insights for the researchers working for tuning-free *DE*. This paper is framed based on the results and graphs obtained through the experimental setup designed for the above said objective.

IV. DESIGN OF EXPERIMENTS

The experimental setup includes five benchmarking functions f_1 – Sphere function (Unimodal Separable), f_2 – Schwefel's 2.22 function (Unimodal Separable), f_3 - Schwefel 1.2 function (Unimodal Nonseparble), f_4 – Schwefel's 2.21 function (Unimodal Separble) and f_5 – Generalized Rosenbrock's function (Multimodal Nonseparable), chosen from the list proposed by Yao et al. [19] . All these functions are minimization function whose global optimal value is zero. The *F* values used in the experiment are set as {0.0, 0.3, 0.6, 0.9}, to include low, medium and high values of *F* in the range of [0.0, 1.0]. The population size (*NP*) is kept as constant 60. The dimension (*D*) for the problem is taken as 2, for easier visualization. Since the *DE* variant *DE/rand/1/~* does not have crossover process, the *CR* parameter is not used. The results are recorded for each Function-*F* value combination for one run, which consists of 100 generations. The detailed description of the test functions is presented in Table 2, which shows the function description along with the domain of the variable.

Table 2. The description of the benchmarking functions used in the experiment.

f_1 – Sphere model $Min\ f_{sp}(x) = \sum_{i=1}^{D} x_i^2$ $-100 \le x_i \le 100$	**f_2 – Schwefel's Problem 2.22** $Min\ f_{sch}(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{30} \lvert x_i \rvert$ $-10 \le x_i \le 10$
f_3 – Schwefel's Problem 1.2 Min $f_{schDS}(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$ $-100 \le x_i \le 100$	**f_4 - Schwefel's Problem 2.21** $Min\ f_{sch3}(x) = \max_i \{ \lvert x_i \rvert, 1 \le i \le D \}$ $-100 \le x_i \le 100$
f_5 – Generalized Rosenbrock's Function $Min\ f_{rof}(x) = \sum_{i=1}^{D} \lvert 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \rvert$ $\quad -30 \le x_i \le 30$	

V. EXPERIMENTAL RESULTS AND DISCUSSIONS

The crossover-free *DE* (*DE/rand/1/~*) mentioned in the experimental setup is implemented to solve the five benchmarking functions listed in Table 2. For each function *DE/rand/1/~* is executed for one run which has 100 generations. At solving each function, the initial population is initialized randomly with 60 candidates for each run. For easy visualization and analysis, this experiment uses the benchmarking problems as 2-dimensional problems (*D*=2). In the population, each candidate solution has two parameters x_1 and x_2 (since $D = 2$). The values for the parameters x_1 and x_2 of the candidates are randomly chosen within the lower and upper bounds mentioned in the function description (Table 2). The fitness of each candidate in the population is measured as the objective function value of the corresponding benchmarking function.

The evolutionary search for global optimum solution by *DE/rand/1/~* starts with this initial population. At each generation a new population is generated from the current population by applying the variation (only mutation) and selection operations.

In *DE/rand/1/~*, the variation operation is done by the *differential mutation* which does *rand/1* mutation to generate a mutant vector for each vector in the current population. Now a one-to-one knock-out competition is done between the mutant vector and the current vector to decide the survivor vector for next generation. This process is repeated for 100 generations. The objective function value of the fittest candidate in the 100th generation is regarded as the optimal solution and is named as *Best Objective Function Value* (BO_FV). This value is used as measure to report the accuracy of the solution obtained and to indicate the performance of *DE/rand/1/~* in solving a function.

Since the aim of this study is to analyze the effect of F on population diversity and the values of F were grouped into three (*Low*, *Medium*, *High*), in our earlier study, only four values of F from its domain of values are chosen for this experiment (0.0, 0.3, 0.6 and 0.9). Each function is solved using *DE/rand/1/~* for all the values of F. The BO_FV obtained for each *function-F* combination is presented in Table 3.

Table 3. The BO_FV obtained by *DE/rand/1/~*

Function	*F=0.0*	*F=0.3*	*F=0.6*	*F=0.9*
f_1	805.9008	**9.5134e-23**	1.4378e-17	7.9677e-13
f_2	11.2555	**1.5731e-13**	6.4236e-10	1.9912e-07
f_3	0.002004	**6.5063e-48**	9.4358e-38	1.7281e-33
f_4	8.2669	**9.0542e-25**	2.538e-19	5.8301e-14
f_5	24086.7681	0.20491	0.20557	**8.678e-06**

Also to analyse the impact of varying values of F in the population level, the population is visualized as a 2-dimensional graph. The x and y axes of the graph are the parameters x_1 (Dimension 1) and x_2 (Dimension 2), respectively. All the candidates in the population are plotted in the 2D space as (x_1, x_2) points. This graph is generated at each generation when *DE/rand/1/~* is solving a benchmarking function with a particular F value. Thus a run of *DE/rand/1/~* generates 100 graphs of population. These graphs are continuously monitored to identify the population level changes from the 1st generation to 100th generation. The graphs are depicted in Table 4 to 7. Due its large numbers, for each run only a set of graphs which are showing significant changes are shown in this paper.

Premature Convergence (*PrC*) was observed for F=0.0, as there were no population variation operators available (No cross over and No mutation, but only selection of a random candidate). Stagnation (*Stag*) was observed with multimodal function. In all other cases, the proposed Crossover-Free *DE* is giving Successful Convergence (*SuC*). Table 4, 5 and 6 show the changes in population in case of *PrC*, *SuC* and *Stag* cases respectively. The BO_FV for the test functions considered is listed in Table 3. From the experiments conducted, following observations were made: -

- F = 0.0 gives *PrC* for all functions. This makes no mutation to happen for each vector in the population, as it makes the scaled difference vector in Equation (1) to zero and performs no addition to the base vector. Hence, for every candidate in the population another random candidate of the population which is selected as base vector becomes the mutant vector. If this random candidate is better than the current candidate it replaces the current candidate in the next generation. As a result, as the generations proceed, the best candidates in the initial population will take over the entire population and the search will end up with *PrC* with a local optimal value. Within 10 generations, the population landed in *PrC* for all functions. This shows that the necessity of bringing diversity in the population using the variation operators to proceed further with more generations to reach the global optimal solution. The evidences are visualized in Table 4 for all the functions f_1, f_2, f_3, f_4 and f_5.

- The population for functions f_1 and f_2 converge to the global optimum (0, 0) for F = 0.3, 0.6 and 0.9. All the candidates in the population are converging to the global optimum point (0, 0). A sample case of *SuC* is visualized in Table 5 for f_2.

- In case of function f_3, the candidates in the population get aligned in a vertical straight line. All the candidates did not converge to global optima, but there was at least one candidate that reached the point (0, 0).

- In case of function f_4, the candidates in the population get aligned in a horizontal straight line to the left of global optimum. All the candidates did not converge to global optima, but there was at least one candidate that reached the point (0, 0).

- In case of function f_5, the candidates in the population get converged to a value closer to global optimum. A sample case of *stag* for f_5 is visualized in Table 6. The BO_FV value is stagnated at 0.20557 (for F = 0.6). Visually, it is not well differentiated in the figures given in Table 6 (after generation 40), due to the scales used for X and Y axes.

- Major changes in the population happen between the generations 0 to 20. This is more evident for the cases with Low F value. In the remaining cases, population convergence happened before 40 generations with very little changes between the generations 20 to 40.

- Increased F values cause slight delay in population convergence. This is due to the fact that the large step sizes lead more exploration but not the exploitation.

- Visualizing population changes could distinguish *PrC* from the other two cases. *SuC* and *Stag* could not be distinguished visually alone as the values were really closer to global optimum.

From the results obtained, it can be concluded that 2-dimensional unimodal functions finds the global optima without crossover for Low, Medium and High values of F.

Table 4. Evidences for F = 0.0 giving Premature Convergence (a) at Initial (b) in Middle of Search (c) at the End of Search

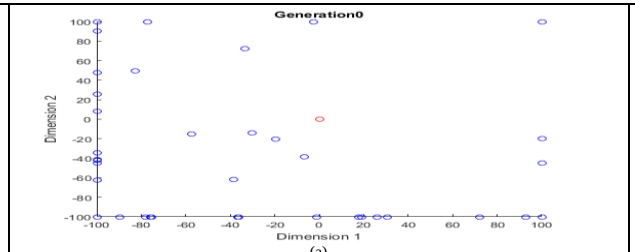
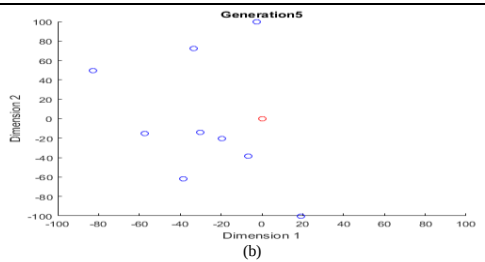
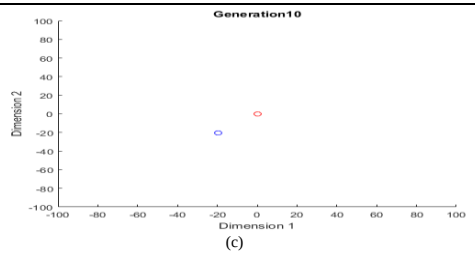
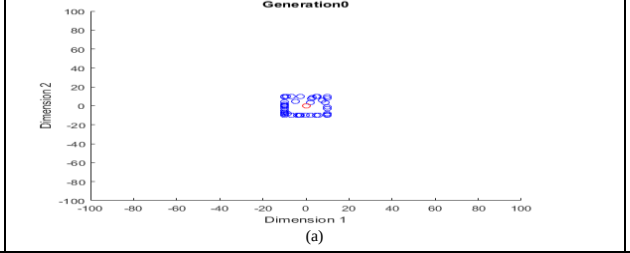
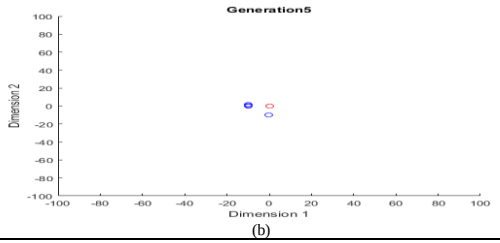
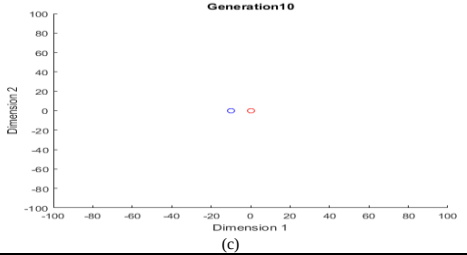
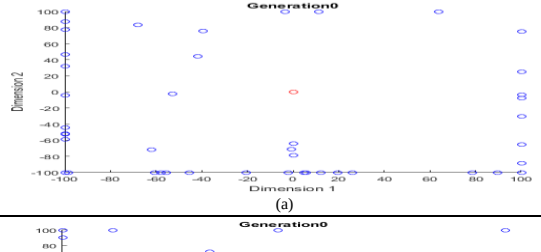
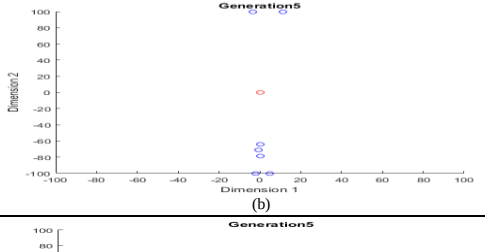
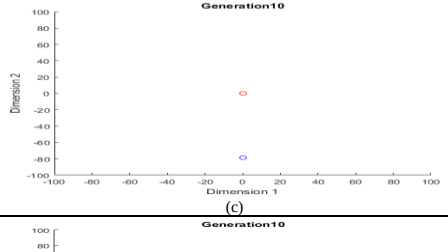
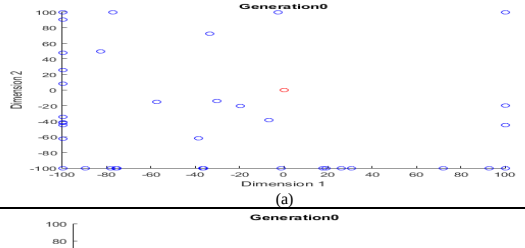
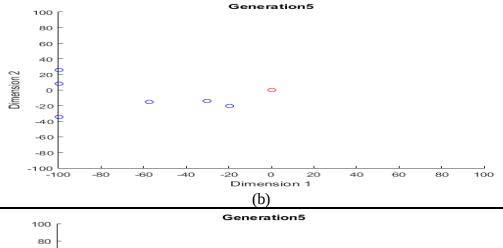
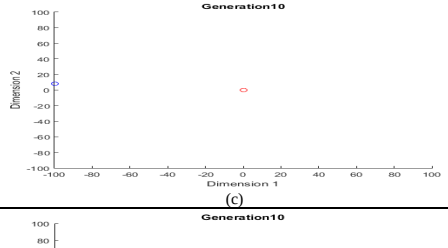
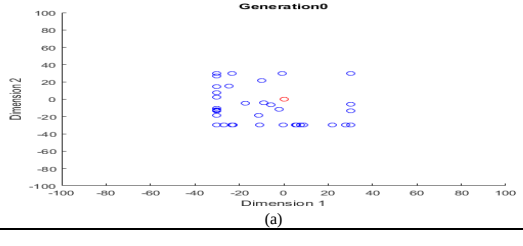
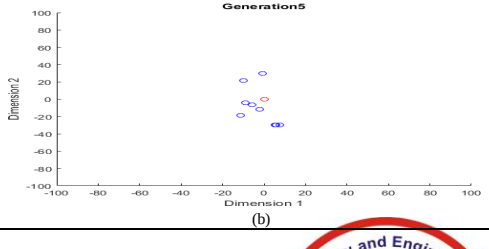
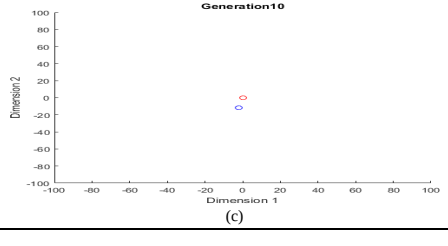
f_1	 (a)	 (b)	 (c)
f_2	 (a)	 (b)	 (c)
f_3	 (a)	 (b)	 (c)
f_4	 (a)	 (b)	 (c)
f_5	 (a)	 (b)	 (c)

Table 5. Visualizing change in population for Successful Convergence using f_2 - Schwefel's Problem 2.22

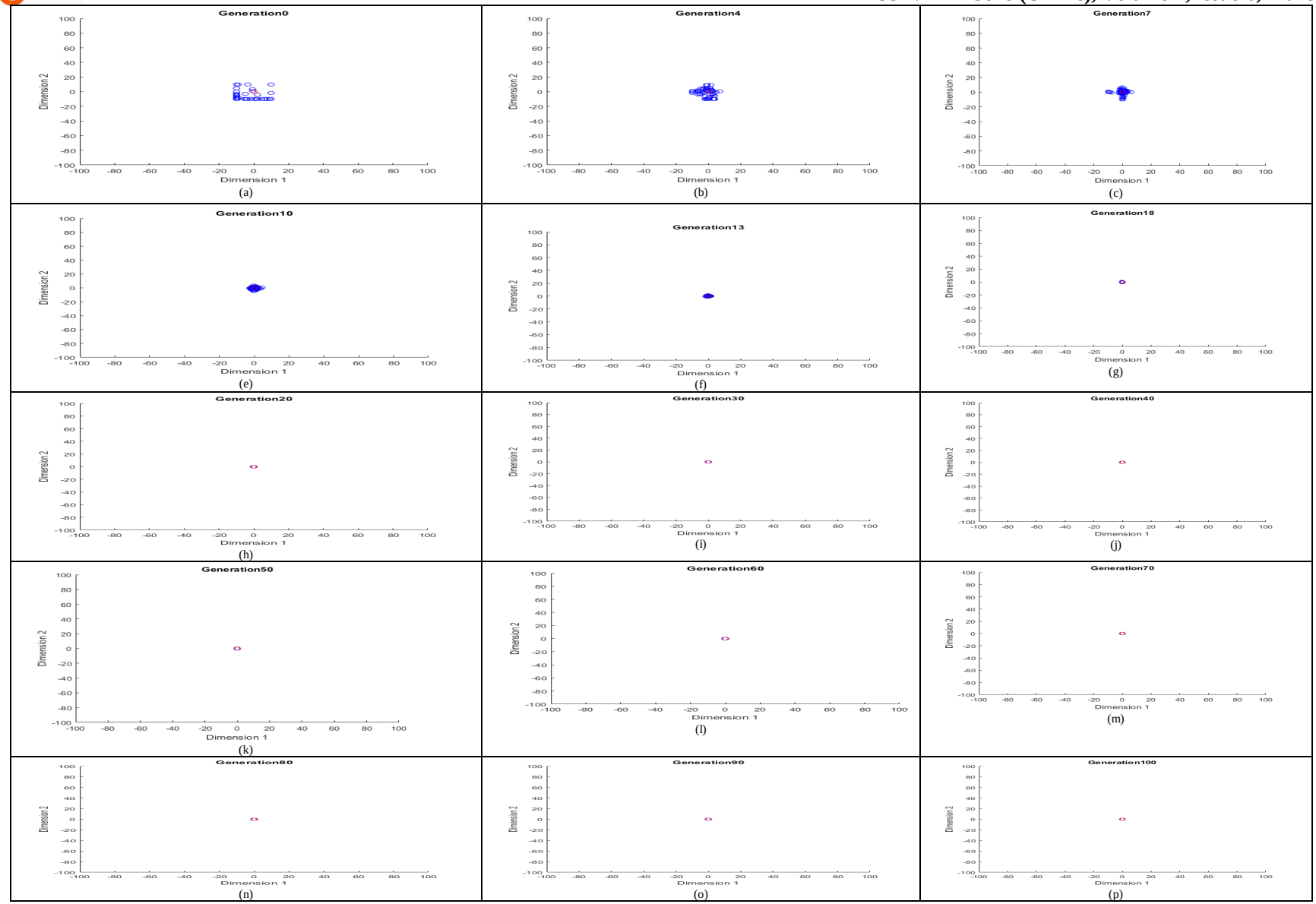
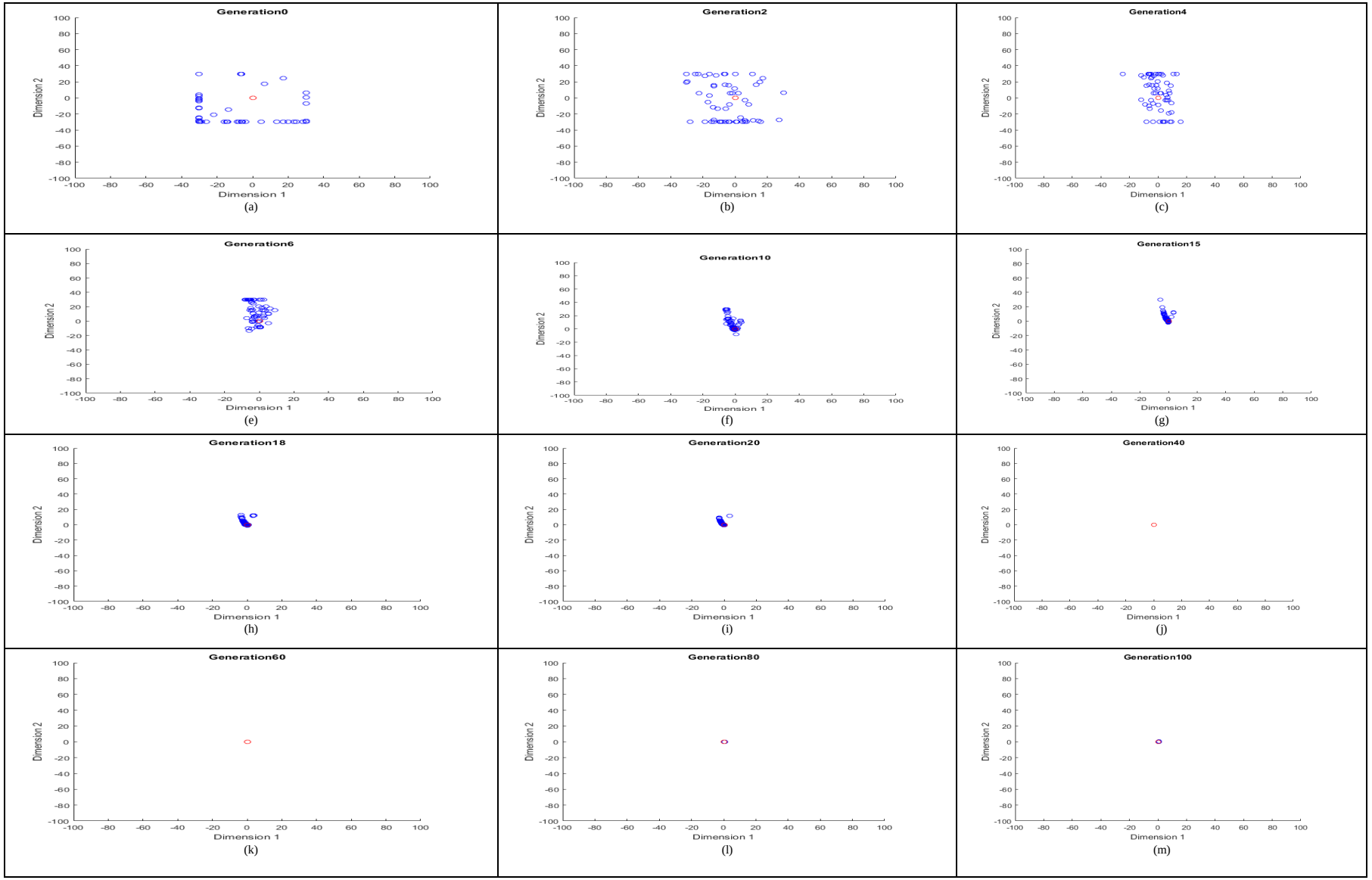


Table 6. Visualizing change in population for Stagnation using f_5 – Generalized Rosenbrock's Function


Generation0
Generation2
Generation4
Generation6
Generation10
Generation15
Generation18
Generation20
Generation40
Generation60
Generation80
Generation100
Dimension 1
Dimension 2
(a)
(b)
(c)
(e)
(f)
(g)
(h)
(i)
(j)
(k)
(l)
(m)

The best values are observed in Low F value 0.3 for all the four unimodal functions. For the multimodal function used in the experiment, the $BO_FV = 0.000008$ is obtained for $F = 0.9$. From the literature, it is a known fact that mutation is the main operation that brings about changes to the population in case of *DE* algorithm. Our experiment proves that for smaller dimension problems, *DE* is capable of producing solutions for unimodal problems with Mutation alone.

VI. CONCLUSIONS

This paper presented the empirical results and visualized the observations obtained running a crossover-free *DE* algorithm for different benchmarking functions, to validate the role of mutation scale factor (F) parameter with different values. Four different values of F are considered for the study to cover the low, medium and high values from the domain of F. This work considered five numbers of 2-dimensional benchmarking problems for easy visualization. The sample cases for successful convergence, premature convergence and stagnation are visualized. This work can be extended further to provide strong validations on the impact of F by increasing the numbers and dimensions of the benchmarking functions

REFERENCES

1. David H. Wolpert and William G. Macreedy. "No Free Lunch Theorems for Optimization", *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp.67-82, 1997.
2. Storn, R and Price, K, "Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces". *Technical Report* - TR-95-012, 1995.
3. Storn, R and Price, K, "Differential evolution—a simple and efficient heuristic strategy for global optimization and continuous spaces", *Journal of Global Optimization*, Vol.11, No. 4, pp. 341–359, 1997.
4. Pranav, P and Jeyakumar, G, "Control Parameter Adaptation Strategies for Mutation and Crossover Rates of Differential Evolution Algorithm – An Insight", *In Proceedings of IEEE International Conference on Computational Intelligence and Computing Research (ICCIC-2015)*, pp. 563 – 568, 2015.
5. Dhanya M Dhanalakshmy, Pranav P and Jeyakumar G., "A Survey on Adaptation Strategies for Mutation and Crossover Rates of Differential Evolution Algorithm", *International Journal on Advanced Science, Engineering and Information Technology*, Vol. 6, No. 5, pp. 613-623, 2016.
6. Storn R, "On the usage of differential evolution for function optimization", *In Biennial Conference of the North American - Fuzzy Information Processing Society*, 1996.
7. Roger Gamperle, Sibylle D. Muller and Petros Koumoutsakos, "A Parameter Study for Differential Evolution", *In WSEAS Int. Conf. on Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation (WSEAS ICAISFSEC)*, pp. 293, 2002.
8. Joseph Trvdik, "Competitive Differential Evolution", MENDEL'06, *12th international conference on soft computing*, pp 7–12, 2006.
9. Joseph Trvdik, "Differential evolution with competitive setting of control parameters", *In Proceedings of the international multiconference on computer science and information technology, Task Quarterly*, Vol. 10, No. 4, pp.1001-1011, 2007.
10. M. M. Ali and A. Törn, "Population set-based global optimization algorithms: Some modifications and numerical studies", *Computers and Operation Research*, Vol. 31, No. 10, pp.1703–1725, 2004.
11. Janez Brest, Saso Greiner, Borko Boskovic, Marjan Mernik and Viljem Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems", *IEEE Transactions on Evolutionary Computation*, Vol.10, No. 6, pp.646-657, 2006.
12. J. Liu, J. Lampinen, "A Fuzzy Adaptive Differential Evolution Algorithm", *Soft Computing*, Vol.9, pp. 448-462, 2005.
13. Jeyakumar, G and Shunmuga Velayutham, C, "Distributed Mixed Variant Differential Evolution Algorithms for Unconstrained Global Optimization", *Memetic Computing*, Vol. 5, No. 4, pp.275-293, 2013.
14. Jeyakumar, G and Shunmuga Velayutham, C, "Distributed Heterogeneous Mixing of Differential and Dynamic Differential Evolution Variants for Unconstrained Global Optimization", *Soft Computing*, Vol. 18, No. 10, pp.1949-1965, 2014.
15. Mezura-Montes, E., Velazquez-Reyes and J. Coello, C.A., "A comparative study on differential evolution variants for global optimization*", In Proceedings of the 8th annual conference on Genetic and evolutionary computation GECCO 2006*, 2006.
16. Akhila M. S, Vidhya C. R and Jeyakumar, G, "Population diversity measurement methods to analyze the behavior of differential evolution algorithm," *International Journal of Control Theory and Applications*, Vol. 8, No. 5, pp.1709-1717, 2016.
17. Vaishali Yadav, Tarun Kumar Sharma, Ajith Abraham and Jitendra Rajpurohit, " Enhanced Asynchronous Differential Evolution Using Trigonometric Mutation" , *In Advances in Intelligent Systems and Computing*, DOI: 10.1007/978-3-319-60618-7_38, 2018.
18. Karol Opara and Jaroł aw Arabasb, "Comparison of mutation strategies in Differential Evolution – A probabilistic perspective", *Swarm and Evolutionary Computation*, Vol. 39, pp. 53– 69, 2018.
19. Yao, X., Liu, Y., Liang, K.H., Lin, G., "Fast evolutionary algorithms", *Advances in Evolutionary Computing: Theory and Applications*, 2003.
20. P. R. Radhika and Shunmuga Velayutham C., "Visualization – A Potential Alternative for Analyzing Differential Evolution Search*", In Intelligent Systems Technologies and Applications*, Vol. 1, pp. 31-41, 2016.

AUTHORS PROFILE

Ms. Dhanya M Dhanalakshmy is working as Assistant Professor in the Department of Computer Science and Engineering, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore from 22.09.2011. She did Bachelor of Computer Application and Master of Computer Application from M.G. University, Kerala. She holds M.Tech in Computer Vision and Image Processing from Amrita Vishwa Vidyapeetham. Currently she is pursuing her Ph.D in the field of Evolutionary Computing – Differential Evolution. Her areas of interest are Image Processing, Video Processing and Evolutionary Computing. Currently, she guides students in the areas of Video Processing and Image Watermarking.

G. Jeyakumar currently an Associate Professor in the department of Computer Science and Engineering, Amrita School of Engineering, Amrita Vishwa Vidyapeetham University, Tamil Nadu, India since 2000. He received his B.Sc degree in Mathematics in 1994, M.C.A degree (under the faculty of Engineering) in 1998 from Bharathidasan University, and Ph.D degree in Distributed Differential Evolution Algorithm in 2013, from Amrita Vishwa Vidyapeetham University, Tamil Nadu, India. His research interest includes Parallelization and Applications of Evolutionary Algorithms, Artificial Intelligence Techniques and Human Modeling. He has published numerous papers in reputed journals and conference proceedings, out of which majority of the publications are indexed in SCOPUS. He has got best paper awards for few of his publications. He has guided many student projects/thesis belong to the courses B.Tech, M.Tech (CSE), M.Tech (Automative Eng), M.C.A and M,Phil and currently guiding PhD scholars, many UG and PG students.

Dr. C. Shunmuga Velayutham currently serves as an Associate Professor in the Department of Computer Science & Engineering, Amrita School of Engineering, Amrita Vishwa Vidyapeetham. He received his Ph.D. from Dayalbagh Educational Institute, Agra, Uttar Pradesh in 2005.

His current research interests include Evolutionary Computation specifically, Visualizing Genetic and Evolutionary Computation, Population-algorithm based portfolios, Tuning-free evolutionary algorithms etc. He has supervised two Ph.D, and currently supervising 3 Ph.D scholars. He has served as a Reviewer for several international journals and conferences including *IEEE Transactions on Fuzzy Systems*, *International Journal of Machine Learning and Cybernetics (Springer)*,

IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2015), and IEEE World Congress on Computational Intelligence WCCI 2016 (including Congress on Evolutionary Computation CEC 2016).