A New Generation Alternation Model for Differential Evolution

Nasimul Noman and Hitoshi Iba
Department of Frontier Informatics
The University of Tokyo
Chiba 277-8561, Japan
{noman,iba}@iba.k.u-tokyo.ac.jp

ABSTRACT

We present a modified version of Differential Evolution (DE) for locating the global minimum at a higher convergence velocity. The proposed model differs from conventional DE by applying selection both for reproduction and survival, whereas the original model applies exclusively "knock-out" selection mechanism for survival. Because of its one-to-one reproduction strategy DE often consumes too many fitness evaluations to locate the global optimum. In this work we show that selecting parents for breeding and offspring for survival, DE's search capability can be further accelerated, which will be particularly useful for expensive function optimizations. Computational results using many benchmark functions are reported which show significant improvements in the convergence characteristics of the proposed algorithm over the original one.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic methods; G.1.6 [**Optimization**]: Global optimization; G.3 [**Probability and statistics**]: Probabilistic algorithms

General Terms

Algorithms, Performance, Experimentation

Keywords

Differential Evolution, Real parameter optimization, Evolutionary Computation, Generational model

1. INTRODUCTION

Global optimization is the field of computational sciences which deals with finding the absolutely best solution for any mathematical problem. For finding a global optimal solution for the problem, a set of parameters that minimizes/maximizes a systems desirable properties is searched. The desirable properties to be minimized/maximized are often formulated as a function commonly known as *objective function*. Finding the global optimum in the continuous domain is particularly difficult to solve exactly. Presence of many local optimal solutions, epistasis among the parameters, noise and inherent peculiarity of the objective function make the optimization task even more obscure. Therefore, a useful global optimization algorithm should be reliable to locate global optima, simple to implement, easy to use, robust and fast in spite of such difficulties [2].

In last few decades, different kinds of deterministic and stochastic algorithms have been proposed for optimization in continuous domain. Among the stochastic approaches, Evolutionary Computation (EC) offers a number of exclusive characteristics e.g. easy to design, robust and reliable performance, little or no information requirement etc. which makes it an attractive choice. Therefore, there had been many studies related to real-parameter optimization using EC resulting in many variants such as *Evolutionary Strategies* (ES) [4], *Real Coded Genetic Algorithms* (RCGA) [8, 14], *Differential Evolution* (DE) [12], *Particle Swarm Optimization* (PSO) [5] etc.

Differential Evolution (DE) is one of the most recent Evolutionary Algorithms (EAs) for solving real-parameter optimization problems. Like other EAs, DE is a population-based, stochastic global optimizer capable of working reliably in nonlinear and multimodal environments [12]. Using a few and easily chosen parameters DE exhibits an overall excellent performance for a wide range of benchmark functions. Because of it simple but powerful search capability it has got many real world applications, such as pattern recognition, digital filter design, neural network training etc [9]. The advantages of DE, such as simple and easy-to-understand concept, compact structure, ease of use, high convergence characteristics and robustness have proved it as a high-class technique for real-valued parameter optimization.

Though DE was designed using the common concepts of EAs, such as multipoint searching, use of recombination and selection operators, it has some unique characteristics that make it different from many others in the family. The major differences are in the way offspring are generated from parents and the selection mechanism that DE applies to transit from one generation to next. DE uses a one-to-one spawning and selection relationship between each individual of the population and its offspring. Though these features are the strength of the algorithm can sometime turn into weakness especially when the global optimization should be located with an increased velocity. By breeding an offspring for each individual DE sometimes explores too many search points before locating the global optimum. Therefore, there remain possibilities of exploiting other generational models available in EC for accelerating the convergence characteristics of DE. In this work we have presented such an effort.

The paper is organized as follows. The next section of this paper contains a brief overview of DE. The third section presents the newly proposed model for DE. Section 4 reports experiments on benchmark functions with results. Section 5 discusses the results focusing on the proposed model characteristics. Finally Section 6 concludes the paper.

2. DIFFERENTIAL EVOLUTION

Like other evolutionary algorithms, DE is a population-based stochastic optimizer that starts to explore the search space by sampling at multiple, randomly chosen initial point [11, 9]. Thereafter, the algorithm guides the population towards the vicinity of the global optimum through repeated cycles of reproduction and selection. The generation alternation model used in classic DE to refine candidate solutions in successive generations is described as

DE

```
1. Generate an Initial Population $P^G$
2. ***Evaluate*** $P^G$
3. For each individual $I$ in $P^G$
4.         ***Reproduce*** an offspring $J$ from $I$
5.         $P^{G+1} = P^{G+1} \cup$ ***Select*** $(I, J)$
6. Set $G = G + 1$
7. Repeat Step 3 to 6 until termination criteria
   is met
```

Now different components of DE algorithm are summarized as follows:

Parent Choosing: As shown in the DE model, each individual in the current generation is allowed to breed through mating with other randomly selected individuals from the population. Specifically, for each individual $x_i^G, i = 1, \cdots, P$, where G denotes the current generation, three other random individuals x_j^G, x_k^G and x_l^G are selected from the population such that j, k and $l \in \{1, \cdots, P\}$ and $i \neq j \neq k \neq l$. This way a parent pool of four individuals is formed for breeding an offspring.

Reproduction: After choosing the parent individuals DE applies a *differential mutation* operation for generating a mutated individual y_i^G, according to following equation

$$y_i^G = x_j^G + F(x_k^G - x_l^G) \tag{1}$$

where F, commonly known as *scaling factor* or *amplification factor*, is a positive real number typically less than 1.0 that controls the rate at which population evolves. Next, to complement the differential mutation search strategy, DE employs a crossover operation often referred to as *discrete recombination* in which the mutated individual y_i^G is mated with x_i^G to generate the *offspring* or *trial individual* x_i^{G+1}. The genes of x_i^{G+1} are inherited from x_i^G and y_i^G determined by a parameter called *crossover probability* ($C_r \in [0, 1]$) as follows

$$x_{i,t}^{G+1} = \begin{cases} y_{i,t}^G & \text{with probability } C_r \\ x_{i,t}^G & \text{with probability } (1 - C_r) \end{cases} \tag{2}$$

where $t = \{1, \cdots, N\}$ denotes t-th parameter of individual vectors. From above description another difference between DE and GA becomes clear; that is in DE the mutation is applied before crossover which is just opposite in GA. Moreover in GA mutation is applied occasionally to maintain diversity in the population whereas is in DE mutation is a regular operation applied to generate each offspring.

Selection: DE applies selection pressure only when replacing individuals. A *knock-out* competition is played between each individual x_i^G and its offspring x_i^{G+1} and the winner is selected deterministically and promoted to next generation.

3. PROPOSED DE VARIANT

Generally two types of selection methods are applied in EC, *selection for reproduction* and *selection for survival* [1]. The first one determines how to distribute reproductive opportunity among the individuals of the population and the later one determines how to administer the life span of different individuals for favoring the survival of promising individuals. Different EAs apply different combinations and implementations of these two selection criteria. In DE no individual is favored for reproduction compared to others [9]. In other words each individual gets an opportunity to spawn its offspring mating with other individuals. Generally DE is expected to work with a larger generation, typically between $5N$ to $10N$, where N is the problem dimension [12, 11]. Therefore generating an offspring for each individual of current population often DE visits too many search points before reaching the global optimal.

On the other hand using one-to-one survivor selection criteria, DE ignores many promising individuals, exploitation of which could accelerate the search. Actually DE exercises a *knock-out* competition that retains only the best individual that each index or position has ever experienced. Due to this positional elitism strategy it discards an offspring which is better than the most of the current population but worse than its parent [9]. However such rejected individual could be useful to accelerate the search for global optimum.

Therefore, in an attempt to improve the classic DEs selection mechanism we propose a generation alternation model that is commonly found in different EAs. The modified DE algorithm which we call genDE can be described as follows

genDE

```
1. Generate an Initial Population $P^G$
2. ***Evaluate*** $P^G$
3. ***Choose*** a parent pool $P_p$
4. For each individual $I$ in $P_p$
5.         ***Reproduce*** an offspring $J$
6.         $P_c = P_c \cup J$
7. ***Select*** $P^{G+1}$ from $P^G \cup P_c$
8. Set $G = G + 1$
9. Repeat Step 3 to 8 until termination criteria
   is met
```

Now implementation of different portions of the genDE algorithm is described in following lines.

Parent Choosing: The set of parents each member of which will get the opportunity of reproduction is selected in two parts. First, a deterministic selection operator is used to select P_1 best individuals of current population as parents.

Then a stochastic selection operator is used to select P_2 individuals from the rest of the population. Selecting the first P_1 parents based on their objective function value, reproduction preference is given to superior individuals. The other P_2 parents are selected randomly to maintain the population diversity. Thus selecting $P_p = P_1 + P_2$ parents partially deterministically and partially stochastically a balance in the selection pressure is triggered.

Reproduction: Now these P_p selected parents are employed to generate P_p offspring just like what is done in classic DE. That is for each parent vector, three other distinct random individuals are chosen from the current population P^G and then they are mated using differential mutation and discrete recombination operation to generate the offspring.

Selection: The survivor selection is performed using a selection operator similar to one found in $(\mu + \lambda)$ Evolutionary Strategy (ES). In this elitist strategy we choose survivors (based on fitness) from a combined population consisting of all the individuals of current generation and offspring. That is the best P individuals from $(P + P_p)$ individuals of current and offspring population are chosen as survivors.

4. EXPERIMENTS

4.1 Test Suite

We evaluate the performance of the proposed genDE algorithm comparing with classic DE algorithm using a test suite consisting of 15 benchmark functions. First five test functions of the suite are commonly found in the literature namely Sphere, Ackley's, Griewank's, Rastrigin's and Rosenbrock's functions. The other benchmarks are the first 10 functions from the newly defined test suite for CEC 2005 special session on real-parameter optimization [13]. Our test suite was as follows:

1. F_{sph}: Sphere Function
2. F_{ack}: Ackley's Function
3. F_{grw}: Griewank's Function
4. F_{ras}: Rastrigin's Function
5. F_{ros}: Rosenbrock's Function
6. F_1: Shifted Sphere Function
7. F_2: Shifted Schwefel's Problem 1.2
8. F_3: Shifted Rotated High Conditioned Elliptic Function
9. F_4: Shifted Schwefel's Problem 1.2 with Noise in Fitness
10. F_5: Schwefel's Problem 2.6 with Global Optimum on Bounds
11. F_6: Shifted Rosenbrock's Function
12. F_7: Shifted Rotated Griewank's Function without Bounds
13. F_8: Shifted Rotated Ackley's Function with Global Optimum on Bounds
14. F_9: Shifted Rastrigin's Function
15. F_{10}: Shifted Rotated Rastrigin's Function

Definitions of the first five functions are follows

$$F_{sph}(\vec{x}) = \sum_{i=1}^{n} x_i^2,$$
$$-100 \le x_i \le 100; \qquad F_{sph}^* = F_{sph}(0, \cdots, 0) = 0$$

$$F_{ack}(\vec{x}) = 20 + exp(1) - 20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)$$
$$- exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)$$
$$-32 \le x_i \le 32; \qquad F_{ack}^* = F_{ack}(0, \cdots, 0) = 0$$

$$F_{grw}(\vec{x}) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\frac{x_i}{\sqrt{i}} + 1$$
$$-600 \le x_i \le 600; \qquad F_{grw}^* = F_{grw}(0, \cdots, 0) = 0$$

$$F_{ras}(\vec{x}) = 10n + \sum_{i=1}^{n} x_i^2 - 10\cos(2\pi x_i)$$
$$-5 \le x_i \le 5; \qquad F_{ras}^* = F_{ras}(0, \cdots, 0) = 0$$

$$F_{ros}(\vec{x}) = \sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$$
$$-100 \le x_i \le 100; \qquad F_{ros}^* = F_{ros}(1, \cdots, 1) = 0$$

Functions F_1 to F_{10} are designed by modifying classical benchmark functions to test the optimizers ability to locate a global optimum under a variety of circumstances such as translated and/or rotated landscape, optimum placed on bounds, Gaussian noise and/or bias added etc [13]. A complete definition of these functions are available in [13] and online at http://www.ntu.edu.sg/home/epnsugan. In our test suit F_1 to F_5, F_{sph} and F_{ros} are unimodal and the rest are multimodal functions.

4.2 Performance Evaluation Criteria

For evaluating the performance of the algorithms we used criteria similar to that defined in [13]. We investigated the performance of genDE compared to DE for the above mentioned benchmark functions using *function error value*. The *function error value* is defined as $(f(x) - f(x^*))$ where x^* is the global optimum of the function. The maximum number of fitness evaluations we allowed for each algorithm to minimize this error was $500,000$. We repeated 25 trails on each function. The fitness evaluation criteria were as follows

1. **Error:** The minimum function error value that algorithm can find, using 500,000 fitness evaluations at maximum, was recorded in each run and the average and standard deviation of the error values were calculated. Also the number of runs in which the error reached zero was counted and presented. For this criterion the notation $AVG_{Er} \pm SD_{Er}(CNT)$ was used in Table 1 and Table 3. We evaluated all the benchmark functions at dimension $N = 30$ with various population size and reported the results in Table 1.

2. **Evaluation:** Number of function evaluations required to reach the error value less than 10^{-6} range were recorded in different runs and then the average and standard deviation of the number of evaluations were calculated. Again the number of runs in which the algorithms could reach this accuracy level using maximum $500,000$ fitness evaluations were counted. For this criterion the notation $AVG_{Ev} \pm SD_{Ev}(CNT)$ was used. We evaluated all the benchmark functions at dimension $N = 10$ and $N = 30$ and reported the results in Table 2.

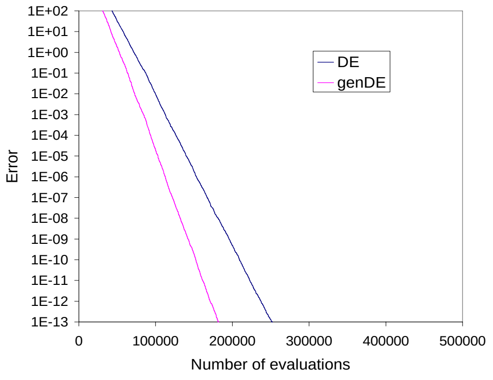

Figure 1: Convergence curves for F_{sph}

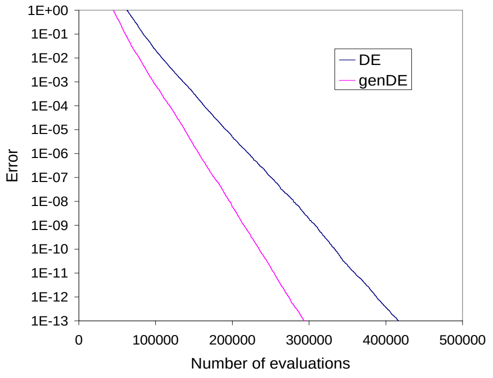

Figure 2: Convergence curves for F_{ack}

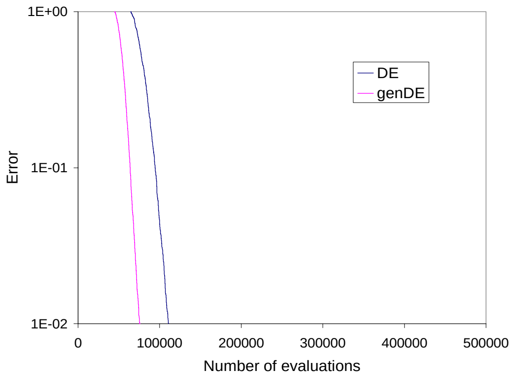

Figure 3: Convergence curves for F_{grw}

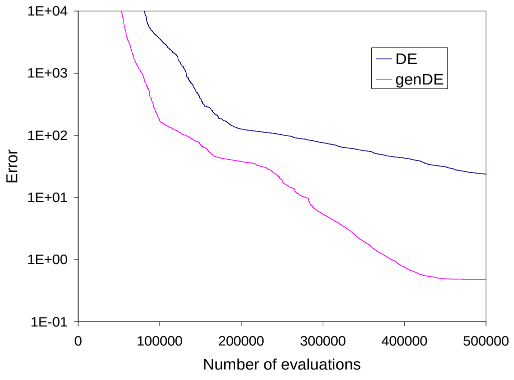

Figure 4: Convergence curves for F_{ros}

3. **Convergence Graphs:** Convergence graphs of the algorithms for $N = 30$. The graphs (Fig 1 to Fig 11) show the average **Error** performance of the total runs.

4.3 Experimental Setup

In our experimentation we used the same sets of initial random population for evaluating the algorithms. Though classic DE uses three control parameters namely *Population Size* P, *Scaling Factor* F and *Crossover Rate* C_r, choice of these parameters are very critical for its performance. F is generally related to the convergence speed. To avoid premature convergence it is crucial that F be of sufficient magnitude [9]. $F = 0.9$ is suggested as a good compromise between convergence speed and probability of convergence in [10]. Between C_r and F, C_r is much more sensitive to problems property and multimodality. For searching in non-separable and multi-modal landscapes $C_r = 0.9$ is a good choice [10]. Therefore we choose $F = 0.9$ and $C_r = 0.9$ for all the functions in every experiments with out tuning them to their optimal values for different problems. Population size is a critical choice for the performance of DE. Therefore in our first set of experiments we investigate the performance of the DE and genDE with different population sizes $P = 30, 50, 100$ and 200. For the proposed genDE there are two more parameters to choose. Based on some preliminary experiments we choose $P_1 = P/4$ and $P_2 = P/2 - P_1$.

The experiments were performed on a computer with 4400 MHz AMD Athlon TM 64 dual core processors and 2GB of RAM in Java 2 Runtime Environment.

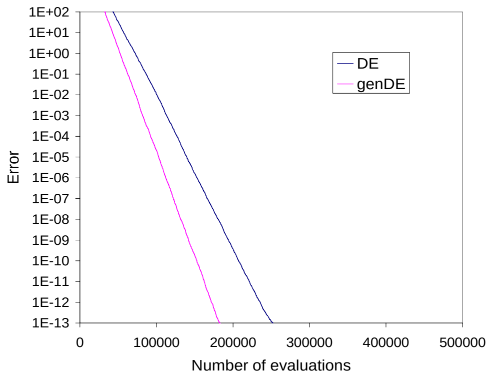

Figure 5: Convergence curves for F_1

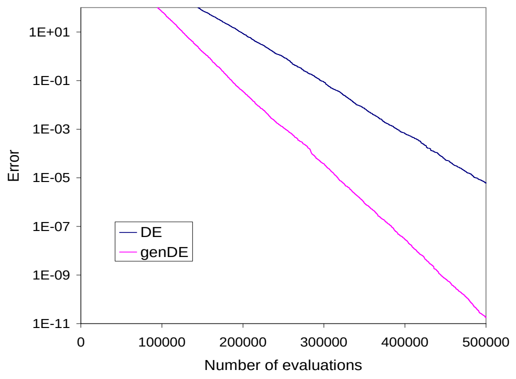

Figure 6: Convergence curves for F_2

Table 1: Best Error values at N=30, after 500,000 fitness evaluation

PopSize=30			**PopSize=50**		
	DE	genDE		DE	genDE
F_{sph}	4.10E-32±6.74E-32	4.98E-46±1.04E-45	F_{sph}	7.63E-07±1.38E-06	7.17E-12±1.01E-11
F_{ack}	5.36E-15±1.52E-15	4.37E-15±1.77E-15	F_{ack}	2.16E-04±1.51E-04	7.81E-07±7.37E-07
F_{grw}	2.17E-03±3.96E-03 (19)	2.17E-03±4.42E-03 (20)	F_{grw}	1.28E-03±3.56E-03	1.33E-10±2.86E-10
F_{ras}	27.7948±9.373461	25.14128±7.685797	F_{ras}	18.04426±6.257083	15.85754±6.015394
F_{ros}	23.5816±93.39593	0.478703±1.295851	F_{ros}	52.24671±46.02755	21.45182±23.45338
F_1	2.27E-15±1.11E-14 (24)	0.0 ±0.0 (25)	F_1	5.52E-07±7.32E-07	9.38E-12±1.95E-11
F_2	6.16E-06±1.18E-05	1.78E-11±2.77E-11	F_2	29.71349±20.74556	3.14E-01±2.50E-01
F_3	2.16E+06±1.68E+06	9.40E+05±6.97E+05	F_3	1.17E+07±6.29E+06	4.98E+06±2.88E+06
F_4	2.57E-01±3.39E-01	4.48E-04±6.72E-04	F_4	514.5162±353.8749	27.84493±17.42823
F_5	742.9728±391.3238	737.1592±373.3289	F_5	1210.688±490.4832	671.0513±438.3644
F_6	4.839094±19.23881	0.608340±1.886657	F_6	108.2925±132.7632	61.17370±110.6009
F_7	7.98E-03±1.06E-02 (1)	3.94E-03 ±5.95E-03 (13)	F_7	8.83E-03±1.00E-02	1.58E-03±3.68E-03
F_8	20.91188±0.12258	20.89110±0.143423	F_8	20.92413±0.039713	20.91692±0.044140
F_9	27.07304±6.92113	25.35556±9.517137	F_9	18.63529±5.051625	17.34090±4.218965
F_{10}	43.42575±31.45708	41.93541±34.10729	F_{10}	201.1525±33.59441	154.3591±78.47720
PopSize=100			**PopSize=200**		
	DE	genDE		DE	genDE
F_{sph}	466.3771±200.9077	48.03943±23.01009	F_{sph}	24493.47±5661.917	15214.46±2473.895
F_{ack}	7.019878±1.528160	3.673443±0.516853	F_{ack}	19.56072±0.484112	18.19396±0.889659
F_{grw}	4.966620±2.005038	1.406541±0.306817	F_{grw}	221.3048±41.25594	146.1406±36.73740
F_{ras}	204.2888±27.80162	160.2033±25.06192	F_{ras}	332.8443±24.98655	314.6762±24.64466
F_{ros}	2.60E+07±6.25E+07	4.34E+05±4.21E+05	F_{ros}	7.44E+09±2.99E+09	2.64E+09±9.47E+08
F_1	7.44E+02±3.63E+02	7.16E+01±4.66E+01	F_1	3.51E+04±7.87E+03	2.38E+04±6.57E+03
F_2	2.95E+04±1.19E+04	1.16E+04±5.53E+03	F_2	1.09E+05±1.45E+04	9.36E+04±1.68E+04
F_3	6.97E+08±1.83E+08	3.89E+08±1.48E+08	F_3	1.17E+09±1.88E+08	1.02E+09±1.97E+08
F_4	4.70E+04±1.43E+04	2.92E+04±1.46E+04	F_4	1.33E+05±2.41E+04	1.25E+05±2.34E+04
F_5	1.56E+04±3.18E+03	1.03E+04±2.80E+03	F_5	3.13E+04±3.44E+03	2.97E+04±2.46E+03
F_6	2.27E+07±1.44E+07	6.64E+05±5.21E+05	F_6	1.33E+10±3.92E+09	5.58E+09±2.24E+09
F_7	102.8455±51.465923	18.99315±9.887694	F_7	2574.956±496.2930	1901.613±418.5901
F_8	20.93837±0.044455	20.91569±0.042073	F_8	20.93323±0.044967	20.91348±0.057117
F_9	217.3055±24.98303	180.5073±23.84669	F_9	368.6602±19.03951	343.0230±24.16168
F_{10}	275.8617±16.73121	258.4102±19.43306	F_{10}	537.6126±49.20362	445.4743±36.35172

Table 2: Fitness evaluations required to achieve accuracy level less than 10^{-6}

N=10			**N=30**		
	DE	genDE		DE	genDE
F_{sph}	32049.08±1214.10 (25)	20172.24±1035.06 (25)	F_{sph}	152329.2±8353.68 (25)	105109.2±3837.74 (25)
F_{ack}	49959.72±1400.72 (25)	31680.76±1325.05 (25)	F_{ack}	228786±56347.93 (24)	155647.8±6581.22 (25)
F_{grw}	-	-	F_{grw}	251914.8±154855.63 (18)	189975±155110.02 (20)
F_{ras}	431417.28±157221.79 (4)	410938.56±178170.23 (5)	F_{ras}	-	-
F_{ros}	132677.28±136841.88 (22)	48155.64±9530.32 (25)	F_{ros}	-	470057.4±35906.69 (15)
F_1	32943.36±1379.18 (25)	20230 ±913.05 (25)	F_1	152683.2±7336.47 (25)	107032.2±5147.29 (25)
F_2	51636±3117.18 (25)	33984.4±1935.71 (25)	F_2	496692±8988.22 (3)	344943±28723.85 (25)
F_3	97590.56±7871.40 (25)	66092.64±5777.24 (25)	F_3 -	-	
F_4	57634.4±3810.59 (25)	40508.16±2207.83 (25)	F_4	-	-
F_5	134144.24±6581.86 (25)	95426.16±3956.56 (25)	F_5	-	-
F_6	95408±84923.25 (24)	65470.28±89047.60 (24)	F_6	498927.6±5302.65 (1)	473542.8±44190.38 (11)
F_7	-	-	F_7	359076±125612.33 (14)	293336.4±155508.94 (16)
F_8	-	-	F_8	-	-
F_9	448548.04±139365.88 (3)	428327.12±164319.31 (4)	F_9	-	-
F_{10}	-	-	F_{10}	-	-

4.4 Comparison with EA

In this set of experiments we compared the performance off the proposed algorithm with two EAs. Both of these EAs were designed using same recombination operators namely *arithmetic crossover* and *Gaussian mutation*, but different selection strategies. The arithmetic crossover operator, a two parent crossover operator, was applied with probability p_c. Two parent individuals x_i^G and x_j^G are mated to generate offspring y_i^G such that $y_{i,t}^G = w_t x_{i,t}^G + (1 - w_t) x_{j,t}^G$ where t is the index of each dimension of different solution vectors and w_t are uniform random numbers in $[0, 1]$. Moreover, we applied Gaussian mutation to each offspring y_i^G with a fixed *mutation rate* and *probability* p_m such that $y_{i,t}^G = y_{i,t}^G + N(0,1)\sigma_m(x_t^{max} - x_t^{min})$. Where σ_m is the *mutation variance*. The only difference between these two EAs, denoted as EA1 and EA2, is the selection model used.

EA1: This algorithm uses ordinary selection strategy commonly found in literature. We used the model in [6]. This implementation preserves P_e elite individuals (determined by fitness) which are promoted to next generation without applying any recombination operator. And a tournament selection of size 2 is used for selecting parents. For each individual x_i^G another random individual x_j^G is chosen from the population and the better one (if it does not belong to elite population) goes through crossover and mutation operation before placing in the next generation. For crossover the other parent is randomly picked from the population.

EA2: This model uses the same selection strategy that we have proposed for DE. That is, P_p parents are selected for reproduction in the same manner as described in section (3). Then each of them produces an offspring using crossover and mutation operation. The other parent for crossover is randomly selected. Then a $(\mu + \lambda)$ selection mechanism is used to choose the survivors.

The choice of parameters for EA1 and EA2 were as follows, $p_c = 1.0$, $p_m = 0.3$, $\sigma_m = 0.01$, $P_e = 30$, $P_1 = P/4$ and $P_2 = P/2 - P_1$. We experimented with Popsize $P = 50, 100, 200$ and 300 and the best average results, found with $P = 100$, are reported here for comparison in Table 3.

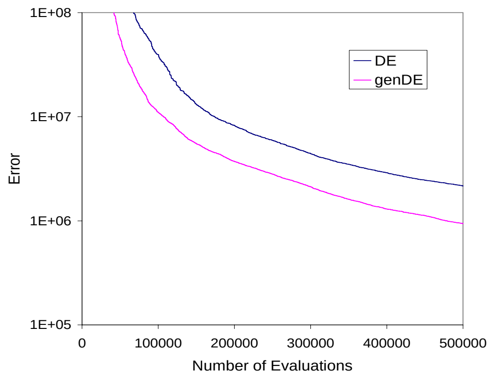

Figure 7: Convergence curves for F_3

5. DISCUSSION OF RESULTS

As shown in Table 1 for most of the functions the average Error values achieved by the proposed genDE is lower than that achieved by classic DE. In every case where DE was going towards the optimum genDE also succeeded to

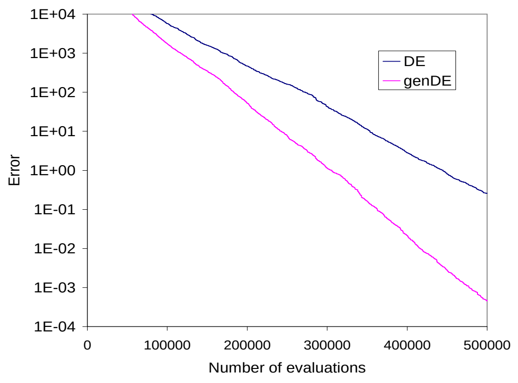

Figure 8: Convergence curves for F_4

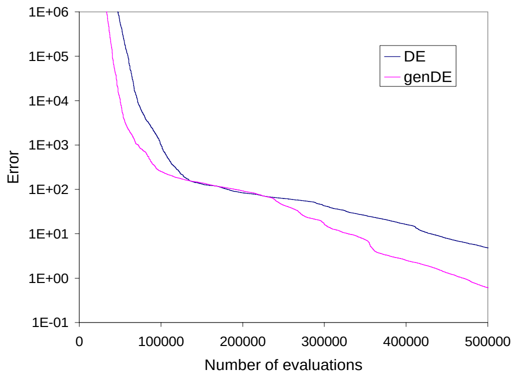

Figure 9: Convergence curves for F_6

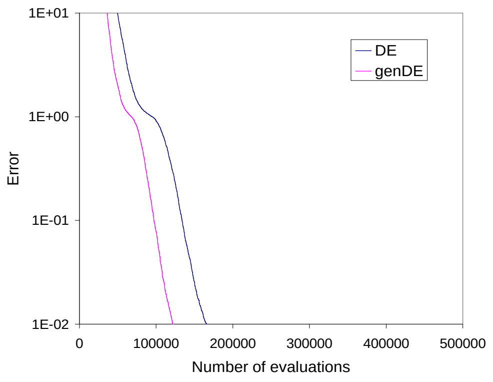

Figure 10: Convergence curves for F_7

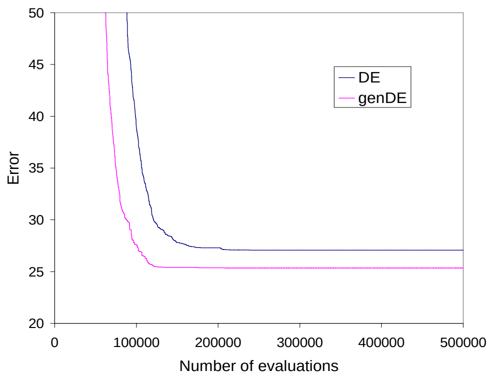

Figure 11: Convergence curves for F_9

Table 3: Comparison of best Error values at N=30, after 500,000 fitness evaluation

	genDE	EA1	EA2
F_{sph}	4.98E-46±1.04E-45	4.24E-07±2.17E-07	0.732647±0.297487
F_{ack}	4.37E-15±1.77E-15	1.96E-04±7.03E-05	0.281236±0.065895
F_{grw}	2.17E-03±4.42E-03 (20)	4.63E-03±5.50E-03	0.446236±0.118072
F_{ras}	25.14128±7.685797	9.352615±2.742911	10.75672±3.0997483
F_{ros}	0.478703±1.295851	74.18848±90.18786	363.9337±220.53187
F_1	0.0 ±0.0 (25)	5.42E-07±3.51E-07	0.7227136±0.2284896
F_2	1.78E-11±2.77E-11	0.48569±0.099001	31.48045±6.655541
F_3	9.40E+05±6.97E+05	1.26E+06±2.39E+05	3.04E+06±5.53E+05
F_4	4.48E-04±6.72E-04	2170.915±698.89357	9.33E+03±2.17E+03
F_5	737.1592±373.3289	6661.574±982.26638	6728.375±995.6071
F_6	0.608340±1.886657	590.7449±307.04394	724.2358±390.1111
F_7	3.94E-03 ±5.95E-03 (13)	0.160041±0.017901	1.095835±0.013463
F_8	20.89110±0.143423	20.52869±0.084398	20.95351±0.053194
F_9	25.35556±9.517137	123.8919±11.03680	114.7719±12.07329
F_{10}	41.93541±34.10729	249.016±26.925252	225.3852±30.0104

do so; moreover it reached closer to the global optimum using equal number of fitness evaluation. Again, since genDE requires fewer fitness evaluations it reached the Error value zero in more trials than that of classic DE (e.g. in F_{grw}, F_1, in F_7). For some functions such as F_8, F_9, F_{10} the performance difference was not significant. We believe that the learning strategy (i.e. DE/rand/1/bin of Eq. (1)) was not good enough to locate the global optimum therefore the classic DE failed. As well as genDE, using the same learning strategy, failed because the selection method alone can not be sufficient to guide a search algorithm towards the global optimum. In some cases, the parameter settings of DE or genDE was not also good enough to locate the global optimum, e.g. none of the algorithm could find a respectable error value for function F_{ras}. But in [7], using a different parameter setting it was shown DE can successfully reach the global optimum for F_{ras} function at higher dimensions using fitness evaluations lower than the limit specified here.

Population size has always substantially influenced DE's performance. DE is especially sensitive to population size because it produces an offspring for each individual in the population. Therefore, if a too large population size is selected then DE exhausts the fitness evaluations very quickly with being able to locate the optimum. To investigate the effect of population size we varied P using 30, 50, 100 and 200. And a quick look at Table 1 will reveal how drastically the performance of DE changes with the change of this parameter. A similar effect on the performance of genDE was also observed. But since it used a selection model that is parsimonious on fitness evaluation, genDE could achieve better Error values for each function than the original model.

A further illustration on how the proposed model can improve the convergence characteristics of DE, can be obtained looking at the convergence curves (Fig 1 to Fig 11). Because, for some functions the performance difference between the algorithms will not be clear looking at the results of Table 1 (e.g. F_{ack}, F_{grw} or F_9). Inspection of the convergence curves will reveal the fact that though at the end of equal number of fitness evaluation both algorithms got similar error values, genDE reached that error values using fewer fitness evaluation.

To have an direct focus on how the proposed method improve the convergence speed of the original algorithm we presented the results of Table 2 that shows the average fitness evaluation required to reach accuracy level less than 10^{-6}. We experimented in $N = 10$ and $N = 30$ dimensional search space. In every case (except in which none could reach the required accuracy), the number of trails in which genDE got the accuracy was higher than that for DE and/or the number of evaluations required for genDE to reach that accuracy level was significantly lower that that needed for DE. For F_{ros} function at $N = 30$ dimension genDE reached the target accuracy in 15 trials whereas DE failed in each.

Table 4: p values of t-Distribution calculated from Table 2

F	p-val	F	p-val
$F_{sph}(N = 10)$	1.19E-36	$F_{ack}(N = 10)$	1.49E-41
$F_{ros}(N = 10)$	4.06E-03	$F_1(N = 10)$	2.69E-37
$F_2(N = 10)$	4.91E-28	$F_3(N = 10)$	1.18E-20
$F_4(N = 10)$	5.06E-24	$F_5(N = 10)$	6.11E-29
$F_{sph}(N = 30)$	2.66E-29	$F_{ack}(N = 30)$	8.24E-08
$F_1(N = 30)$	3.87E-29	$F_2(N = 30)$	6.10E-29

Our last set of experiments was conducted to investigate whether the proposed generation model is suitable for other evolutionary algorithms. As the results of Table 3 suggests, because of the different generation alternation model the algorithm EA2 performed poorly compared to EA1 the original one. We hypothesize that the selection pressure was too high for the applied crossover or mutation operators; therefore EA2 quickly converged without reaching the global optimum. We further speculate, the proposed generation model pairs nicely with the recombination operator of DE and therefore, the increased selection pressure improves the performance of the algorithm.

Results of Table 2 are statistically assessed using Student's t-test. Those functions, in which at least one algorithm reached the accuracy level in all runs, the number of evaluations needed to reach the accuracy level were examined for statistical significant difference and the p-values are shown in Table 4. In every case t-tests indicate there are significant differences between the means in terms of number of evaluations required. This indicates the proposed genDE performed better than the classic DE and the differences are statistically significant.

In real-world situations, fitness evaluation is the most expensive part of the search process; therefore the solution should be located using lowest possible number of fitness evaluation. On the other hand, sampling very few points an algorithm may converge to a local minimum and fail to locate global optimum. Therefore a trade of should be determined that will guarantee convergence using fewer fitness evaluations.

As discussed earlier DE does not give preference to any individual for reproduction and can not work very reliably with a very small population size [9]. Because of it's one-to-one replication strategy DE often visits many points in the search space before locating the global optimum. Since minimum fitness evaluation is always sought, in this work we attempted to increase DEs convergence speed using a different generation alternation model common in other EAs.

While modeling this generation alternation model, we tried to preserve some of the characteristics of classic DE's selection strategy. For example, DE applies elitism by preserving the best solution found in any generation. Our generational model also conserves elite individual applying a $(\mu + \lambda)$ selection for replacement. And none of the models accept any solution that is worse than the worst solution in current population. The fitness independent parent selection model of DE is partially retained as we randomly select P_2 parents (independent of their fitness) in the total parent pool. In choosing the parent population we tried to take advantage of both selection paradigms: stochastic and deterministic. In current implementation we have chosen equal number of deterministic parents (P_1) and stochastic parents (P_2) to trigger a balance between preferring good individuals for reproduction and preserving the population diversity. Furthermore $(\mu+\lambda)$ selection assures that any individual better than most of that in current population is not ignored.

In general selection tends to reduce the diversity of a population whereas the recombination operations increase it. Therefore, we need a selection scheme that pairs off with the recombination operators for designing a successful EA. For example, a selection scheme with a very high selection pressure can be used with highly disruptive recombination operators [3]. From the experiment results it can be empirically stated the selection pressure exerted by the proposed generation model matches well with DE recombination operator and hence genDE performs better than the original model. The change in DE parameters (F and C_r) can be adjusted by changing the size of the parent pool P_p. We can also vary the selection pressure by changing the proportion of elite parents P_1 and random parents P_2 in the parent population. Actually, the parent selection scheme of the proposed genDE is a more generalized version of that in classic DE, because if the number elite individuals in the parent pool is equal to the population size (i.e. $P_1 = P$) then both becomes the same.

6. CONCLUSION

In this work, we made a preliminary study on the use of different generational model for improving the convergence characteristics of conventional DE. Using the concepts of existing schemes in evolutionary computation, we proposed a new model for generation alternation in DE. We studied the performance of the proposed DE model using a test suite consisting five well known benchmark functions and ten newly proposed benchmarks. In our experiment we found the proposed model exhibits an average higher convergence speed than the conventional model. We anticipate the reason of such performance improvement is better pairing of the selection pressure, exerted by the generation alternation model, with the DE recombination operators. However the proposed model should be further studied varying the parent population size, proportion of elite and random parents in the parent pool. There still remains scope of trying other selection mechanisms for reproduction and/or survival.

7. REFERENCES

[1] T. Back, D. B. Fogel, and Z. M. (eds). *Evolutionary Computation 1 : Basic Algorithms and Operators.* Institute of Physics, Bristol, UK, 2000.

[2] T. Back, D. B. Fogel, and Z. M. (eds). *Evolutionary Computation 2 : Advanced Algorithms and Operators.* Institute of Physics, Bristol, UK, 2000.

[3] D. E. Goldberg, K. Deb, and D. Thierens. Toward a better understanding of mixing in genetic algorithms. *Journal of the Society of Instrument and Control Engineers*, 32(1):10–16, Januray 1993.

[4] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, Jun 2001.

[5] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. of IEEE Int'l Conference on Neural Networks*, pages 1942–1948, Dec. 1995.

[6] T. Krink, B. Filipic, and G. B. Fogel. Noisy optimization problems - a particular challenge for differential evolution. In *Proceedings of Congress on Evolutionary Computation 2004 (CEC2004)*, pages 332–339, June 2004.

[7] N. Noman and H. Iba. Enhancing differential evolution performance with local search for high dimensional function optimization. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, pages 967–974, June 2005.

[8] I. Ono, H. Kita, and S. Kobayashi. *Advances in Evolutionary Computing*, chapter A Real-Coded Genetic Algorithm Using the Unimodal Normal Distribution Crossover, pages 213–237. Springer, 2003.

[9] K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization.* Springer, Berlin, Heidelberg, 2005.

[10] J. Rnkknen, S. Kukkonen, and K. Price. Real-parameter optimization with differential evolution. In *Proceedings of 2005 IEEE Congress on Evolutionary Computation*, pages 506–513, 2005.

[11] R. Storn. System design by constraint adaptation and differential evolution. *IEEE Transactions on Evolutionary Computation*, 3(1):22–34, April 1999.

[12] R. Storn and K. V. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, December 1997.

[13] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. In *Technical Report, Nanyang Technological University, Singapore, And KanGAL Report No 2005005, IIT Kanpur, India*, May 2005.

[14] S. Tsutsui, M. Yamamura, and T. Higuchi. Multi-parent re- combination with simplex crossover in real coded genetic algorithms. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'99)*, pages 657–664, July 1999.