Enhancing Differential Evolution Performance with Local Search for High Dimensional Function Optimization

Nasimul Noman
Graduate School of Frontier Sciences
The University of Tokyo
noman@iba.k.u-tokyo.ac.jp

Hitoshi Iba
Graduate School of Frontier Sciences
The University of Tokyo
iba@iba.k.u-tokyo.ac.jp

ABSTRACT

In this paper, we proposed *Fittest Individual Refinement* (FIR), a crossover based local search method for Differential Evolution (DE). The FIR scheme accelerates DE by enhancing its search capability through exploration of the neighborhood of the best solution in successive generations. The proposed memetic version of DE (augmented by FIR) is expected to obtain an acceptable solution with a lower number of evaluations particularly for higher dimensional functions. Using two different implementations DEfirDE and DEfirSPX we showed that proposed FIR increases the convergence velocity of DE for well known benchmark functions as well as improves the robustness of DE against variation of population. Experiments using multimodal landscape generator showed our proposed algorithms consistently outperformed their parent algorithms. A performance comparison with reported results of well known real coded memetic algorithms is also presented.

Categories and Subject Descriptors: G.1.6 [Problem Solving, Control Methods, and Search]: Heuristic methods

General Terms: Algorithms, Experimentation, Performance.

Keywords: Differential Evolution, Local Search, Memetic Algorithm, Function Optimization, Landscape Generator.

1. INTRODUCTION

In last few decades *Evolutionary Algorithms* (EAs) have become very popular as function optimizers, because they are easy to implement, and exhibit fair performance for a wide range of functions. However, continued development in the community has established that pure *Genetic Algorithms* (GAs) are often not good enough for fine tuning in complex search spaces. As well as new developments have shown that hybridization with other strategies, such as metaheuristics or local searches, can improve the efficiency of search [2, 3]. GAs hybridized with local refinement procedures are known as *Memetic Algorithms* (MAs) [10, 11].

GECCO'05, June 25–29, 2005, Washington, DC, USA

MAs are population based heuristic search approaches for optimization problems closely related to GAs [9]. One MA model that has received attention uses crossover based local search (XLS) procedure. This is particularly attractive for real-coding since there are real-parameter crossover operators that have a self-adaptive nature in that they can generate offspring adaptively according to the distribution of parents without any adaptive parameter [1]. This kind of crossover operator shows promise for building effective XLS.

Differential Evolution (DE) is an effective, efficient and robust optimization method [14] capable of handling nonlinear and multimodal objective functions. The beauty of DE is its simple and compact structure which uses a stochastic direct search approach and utilizes common concepts of EAs. Furthermore DE uses few, easily chosen, parameters and works surprisingly very reliably with excellent overall results over a wide set of benchmark and real-world problems. Experimental results have shown that DE has good convergence properties and outperforms other well known EAs [14, 13].

Despite having a relatively high convergence performance in comparison with the other EAs for nonlinear optimization of multi-modal functions, DE's convergence velocity is still low for optimizing computationally most expensive objective functions, specially at higher dimensions. In this paper, DE algorithm has been hybridized with XLS strategies in an attempt to accelerate the convergence velocity of DE so that better solutions can be obtained with higher speed and increased robustness. The paper is organized as follows. In the next section we briefly focus on several MA models with their properties and review DE. In Section 3, our proposed *Fittest Individual Refinement* (FIR) strategy and the memetic version of DE is presented. Section 4 reports experiments with benchmark functions. Simulations using a landscape generator are presented in Section 5. Finally Section 6 summarizes the findings of our work and concludes.

2. XLS BASED MA AND DE

2.1 XLS Based Memetic Algorithms (MAs)

MAs are evolutionary algorithms that apply a separate *Local Search* (LS) process to refine individuals, e. g. to improve their fitness. MAs are motivated to provide an effective global optimization method by taking advantage of both the exploration abilities of GA and the exploitation abilities of LS. MAs have evolved in mainly two groups depending on the type of LS employed, namely *Local Improvement Process* (LIP) and *Crossover based LS* (XLS) [8]. The first category refines the solutions of each generation by applying efficient

LIPs e.g. hill-climbers. LIPs can be applied to every member of population or with some specific probability and with various replacement strategies.

The other group employs crossover operators for local refinement. A crossover operator is recombination operator that produces offspring around the parents. For this reason, it may be considered to be a move operator for LS strategy. Moreover now there are many sophisticated crossover operators that can generate offsprings adaptively according to the distribution of parents without any adaptive parameter. So they can be employed to create offspring distributed densely around the parents, favoring local tuning. The most common examples of XLS based MAs in literature are *Minimal Generation Gap* (MGG) [4] and *Generalized Generation Gap* (G3) [5]. Both of them employ same parents to spawn multiple offsprings. The idea is to induce an LS on the neighborhood of the parents involved in crossover. In this way, this type of crossover operators constitute an XLS [8].

2.2 Differential Evolution (DE)

DE is a stochastic search algorithm, related to *Evolutionary Computation* (EC), which exploits a population of potential solutions, individuals, to probe the search space. New individuals are generated by combination of randomly chosen individuals from the population. Specifically, for each individual x_G^i, $i = 1, \cdots, N$, where G denotes the current generation, a new individual $y_{G+1}^i$is generated according to the following equation

$$y_{G+1}^i = x_G^j + F(x_G^k - x_G^l) \tag{1}$$

where j, k and l are random integers such that j, k and $l \epsilon \{1, \cdots, N\}$ and $i \neq j \neq k \neq l$ and F is called *scaling factor* or *amplification factor*. This operation is similar to what is commonly known as *mutation* to EC community. In order to achieve higher diversity the mutated individual y_{G+1}^i is mated with x_G^i using a *crossover* operation to generate the *offspring* or *trial individual* x_{G+1}^i. The genes of x_{G+1}^i are inherited from x_G^i and y_{G+1}^i determined by a parameter called *crossover factor* (CF) which regulates how many consecutive genes of the mutated individual on average are copied to the offspring. Finally the offspring is evaluated and replaces its parent x_G^i in next generation if and only if its fitness is better than that of its parent. This is the *selection* process.

The above scheme is not the only variant of DE which has proven to be useful. In order to distinguish among its variants the notation $DE/a/b/c$ is used, where $'a'$ specifies the vector to be mutated which can be random or the best vector; $'b'$ is the number of difference vectors used and $'c'$ denotes the crossover scheme, *binomial* or *exponential*. Using this notation the DE-strategy described above can be denoted as $DE/rand/1/exp$. Other well known variants are $DE/best/1/exp$, $DE/rand/2/exp$ and $DE/best/2/exp$ which can be implemented by simply replacing equation (1) by the equation (2), (3) and (4) respectively. Also each of them has a mate based on binomial crossover.

$$y_{G+1}^i = x_G^{best} + F(x_G^j - x_G^k) \tag{2}$$

$$y_{G+1}^i = x_G^j + F(x_G^k - x_G^l) + F(x_G^m - x_G^n) \tag{3}$$

$$y_{G+1}^i = x_G^{best} + F(x_G^j - x_G^k) + F(x_G^l - x_G^m) \tag{4}$$

where, x_G^{best} represents the best individual in current generation and m and $n \epsilon \{1, \cdots, N\}$ and $i \neq j \neq k \neq l \neq m \neq n$.

3. DE WITH CROSSOVER BASED LS

As mentioned earlier in Section 2.1, XLS are applied to search the neighborhood of the parents locally to improve the parents. We have applied the same strategy to the neighborhood of a single individual. In other words we can say we used XLS for exploring the neighborhood of an individual by mating it repeatedly with different individuals. A similar model of XLS has been proposed by Yang and Kao [16] where they search the neighborhood of each individual and they have named it *Family Competition* (FC).

In our proposed scheme of DE with XLS the basic DE ($DE/rand/1/exp$) is extended by applying some crossover based local search (XLS) to search the neighborhood of the best individual. That is in our XLS procedure the best individual x_G^{best}becomes the family father and its family is explored. This family father and other individual(s), randomly chosen from the rest of the current population, are mated to generate offspring. And this procedure is repeated L times. Finally, L solutions $(C_1, C_2, \cdots, C_L)$ are produced and among these offsprings and family father x_G^{best} the individual with the best score replaces the family father in next generation. We call it crossover based local search for *Fittest Individual Refinement* (FIR). In this paper we have augmented DE algorithm by applying FIR in the general template of MA and call it DEFIR. The formal algorithm for DEFIR can be described, as follows

1. *Randomly Initialize the population* P_G
2. *REPEAT Until Search Converged*
3. $\quad P_{G+1} = \{\phi\}$
4. $\quad$ *REPEAT for each individual* $I \epsilon P_G$
5. $\quad\quad$ *REPRODUCE* I' *from* I
6. $\quad\quad P_{G+1} = P_{G+1} \bigcup Best(I, I')$
7. $\quad$ *ENDREPEAT*
8. $\quad P_{G+1} = FIR(P_{G+1})$
9. *ENDREPEAT*

The only structural difference between basic DE and DEFIR algorithm is application of FIR in each generation for refining the best individual. Now for implementing FIR, currently we propose two schemes: DEfirDE and DEfirSPX.

In DEfirDE scheme, the offspring is generated in the same way offspring generated in $DE/rand/1/exp$. In each generation G for the best individual x_G^{best} we select three individuals x_G^j, x_G^k and x_G^l such that j, k and $l \epsilon \{1, \cdots, N\}$ and $best \neq j \neq k \neq l$. Then a mutated individual $y_{G+1}^i$is generated using equation (1). Finally the offspring C is generated by crossover operation between mutated individual y_{G+1}^i and the best individual x_G^{best}. This procedure is repeated L times and then *selection* is performed.

On the other hand DEfirSPX scheme generates the offspring using *simplex crossover* (SPX) operation [12]. To review, SPX operator uses multi-parent, p parental vectors, for recombination as follows

1. Choose p parents x_G^i , $i = 1, \cdots, p$ according to the generational model used (G denotes the generation) and calculate their center of mass O

$$O = \frac{1}{p} \sum_{i=1}^{p} x_G^i \tag{5}$$

2. Generate random numbers r_i

$$r_i = u^{\frac{1}{i+1}}, (i = 1, \cdots, p-1) \quad (6)$$

where u is a uniform random number ϵ $[0, 1]$

3. Calculate y_i and C_i

$$y_i = O + \varepsilon(x_G^i - O), \quad (i = 1, \cdots, p) \quad (7)$$

$$C_i = \begin{cases} 0, & (i = 1) \\ r_{i-1}(y_{i-1} - y_i + C_{i-1}), & (i = 2, \cdots, p) \end{cases} \quad (8)$$

where ε is *expansion rate*, a control parameter of SPX.

4. Generate an offspring C

$$C = y_p + C_p \quad (9)$$

In DEfirSPX scheme we select the best individual and other $(p-1)$ random individuals from current generation. Then SPX is applied on these p parents to generate offspring. *Selection* is performed after repeating this procedure L times.

The justification for the design of DEFIR is as follows. The basic strategy of EAs is *many points, few neighbors*, i.e. they work by searching the single neighborhood of multiple individuals in parallel over successive generations of populations. On the other hand the XLS based MAs work with *few points, many neighbors strategy*, i.e. they work by searching on a greater neighborhood of one individual in successive generations. DE applies a more directive search in a greedy way. Augmenting this FIR process in the structure of DE we make the search more directive or greedier in a sense. With the progress of the search by exploring the neighborhood of the best individual we hope to find the global optimal at a higher speed. This is similar to *few points, many neighbors strategy* but a greedier one. Hence analogously it can be called *best point neighborhood strategy*.

4. EXPERIMENTS

In our experiments, we investigate the performance of the proposed DEfirDE and DEfirSPX algorithms comparing with $DE/rand/1/exp$ (DE) and $DE/best/1/exp$ (DE(best)) algorithms. As mentioned in Section 3 DEfirDE and DEfirSPX algorithms are implemented by augmenting FIR with DE/rand/1/exp. A brief performance comparison is also performed by comparing the proposed algorithms with some reported results of other *Real Coded Genetic Algorithms* (RCGAs) and *Real Coded Memetic Algorithms* (RCMAs).

4.1 Test Suite

We use 5 test functions commonly used in the literature, which includes *Sphere* model (f_{Sph}), *Ackley* function (f_{Ack}), *Griewank's* function (f_{Grw}), *Generalized Rastrigin's* function (f_{Ras}) and *Generalilzed Rosenbrock's* function (f_{Ros}). All benchmarks chosen are minimization problems. The definitions of the functions are as follows

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

$$f_{Ack}(\vec{x}) = 20 + exp(1) - 20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)$$
$$-32.768 \le x_i \le 32.768; \qquad f^*_{Ack} = f_{Ack}(0, \cdots, 0) = 0$$

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

$$f_{Ras}(\vec{x}) = 10n + \sum_{i=1}^{n} x_i^2 - 10\cos(2\pi x_i)$$
$$-5.12 \le x_i \le 5.12; \qquad f^*_{Ras} = f_{Ras}(0, \cdots, 0) = 0$$

$$f_{Ros}(\vec{x}) = \sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$$
$$-50 \le x_i \le 50; \qquad f^*_{Ros} = f_{Ros}(1, \cdots, 1) = 0$$

f_{Sph} and f_{Ros} are unimodal functions on the other hand f_{Ack}, f_{Grw} and f_{Ras} are multimodal functions. f^* denotes the global minimum for the function.

4.2 Experimental Setup

In our first set of experiments we investigated the performance of DE, DE(best), DEfirDE, DEfirSPX for the benchmark functions mentioned above. Targeting high dimensional optimization, dimensions of the search spaces were chosen as 50, 100 and 200. In order to make the performance comparison fairer we used the same sets of initial random populations for evaluating all algorithms. Each experiment was repeated 30 times. Maximum number of evaluations allowed for each algorithm was $500,000$. For DEs we chose most commonly used parameter setting and did not tune to best parameter values for each problem. The value of F was set to 0.5 and CF was chosen 0.8. Based on some preliminary experiments, for FIR algorithms, $L = 25$ when population size $P \le 200$ otherwise $L = 50$, was chosen. The performance of DE variants is highly dependent on the chosen population size. Therefore to investigate the sensitivity of the proposed algorithms to the change of population size we experimented with three different population sizes $P = N$, $P = 5N$ and $P = 10N$, where N is the dimension of the problem. For simplex crossover operation number of parents $p = 3$ was chosen.

We evaluated the algorithms by calculating average (AVG) and standard deviation (SD) of their attained minimum fitness value within the maximum number of allowed evaluations. If the fitness value was less than 10^{-6} range of actual optimum point, we assume solution is detected. The experiments were performed on a computer with 1700 MHz Intel Pentium processor and 512 MB of RAM in JBuilder X environment.

The second set of experiments was conducted to compare proposed algorithms with some other real coded GAs and MAs proposed in the literature. Comparison was performed with some of the results reported by Lozano et al. in their work [8]. The performance of DE, DE(best), DEfirDE, DEfirSPX were compared with CHC algorithm, *Generalized Generation Gap*($G3-1$), *hybrid steady-state RCMA* (SW-100), *Family Competition* (FC) and *RCMA with crossover Hill Climbing* (RCMA-XHC). For these experiments the setup of parameters were as follows, $N=25$, $P=60$, $L=15$, $F=0.5$, $CF=0.8$ and number of maximum evaluations $100,000$.

4.3 Results

Results of our first set of experiments are reported in Table 1. For each function the results are arranged as AVG $\pm$ SD (Number of Convergence). In all figures, the graphs represent the mean of the best evaluation in 30 runs. Because of limited space, only some representative graphs for different functions are presented.

As shown in Table 1, for ($f_{Sph}, f_{Ack}, f_{Grw}$ *and* f_{Ras}) functions DEfirDE and DEfirSPX succeeded to reach the optimal value in all trials for some cases in which DE and/or DE(best) failed in all trails or at least in some trials. Even in cases, where DE or DE(best) could reach the optimal value in all trails they took higher number of evaluations compared to that needed for DEfirDE or/and DEfirSPX. This can be verified looking at the graphs of Figure 1, 5, 6, 7 and 8. And if we look at cases where none of the four schemes could hit the global optimal, we find that the proposed DEfirDE and DEfirSPX scheme attained AVGs which are significantly better than that achieved by their parent algorithms (Figure 2, 3, 4, 9 and 10). For example in Rosenbrock function none of the above schemes were able to reach the optimal within the maximum number of evaluations. But in all experiments DEfirDE and/or DEfirSPX were able to reach the minimum fitness values. Another observation from Table 1 is, with the increase of dimension, the performance difference between proposed schemes and their parent schemes becomes more significant. And this testifies our claim that the proposed FIR schemes will speedup DE for higher dimensional function optimization.

If we look at the graphs of Figure 1 to 10, then it is found that in every case DEfirSPX started with the steepest convergence curve but in some cases, with the progress of the search it becomes flattened. This is because at the beginning of the search SPX performs local searching using individuals randomly scattered in the search space and becomes very successful in generating offspring with high fitness, but at later generations it generates individuals densely around the populations which slows down the effect of local search. On the other hand DEfirDE strategy, makes use of the operators of DE, starts slowly compared to DEfirSPX but continue to improve the fitness to the end of the search and in later generations approaches to DEfirSPX steadily (Fig 2, 3, 4, 9 and 10). Though for both multimodal ($f_{Grw}, f_{Ras}, f_{Ack}$) and unimodal ($f_{Sph}, f_{Ros}$) functions DEfirDE and DEfirSPX exhibited superior performance, results show that they were more effective to multimodal functions compared to unimodal one.

Storn and Price suggested a larger population size (between 5N to 10N) for DE [14], although later many others found DE's performance good even with a smaller population. To check the sensitivity of the proposed schemes to the

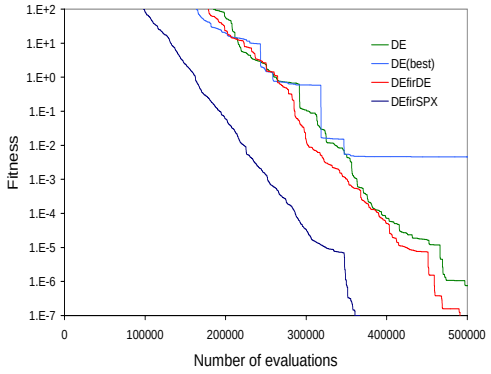

Figure 1: Sphere Function N = 100 P = 100

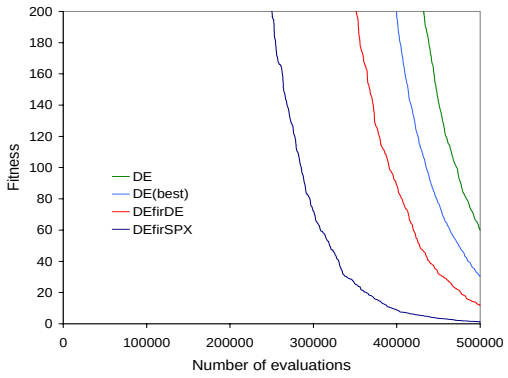

Figure 2: Sphere Function N = 100 P = 1000

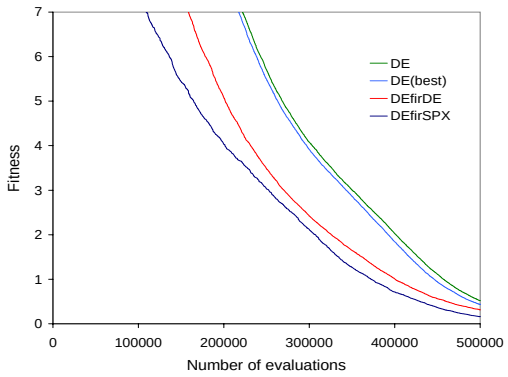

Figure 3: Ackley Function N = 200 P = 200

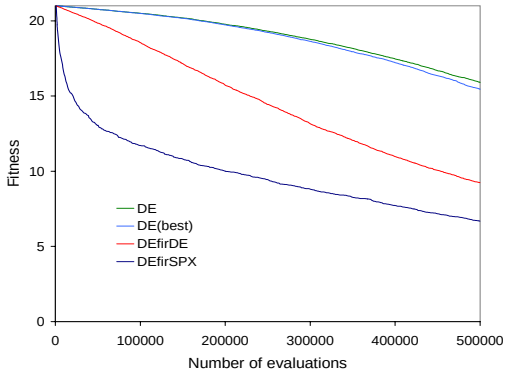

Figure 4: Ackley Function N = 200 P = 1000

Table 1: AVG ± SD (Number of Convergence) of final results for f_{Sph}, f_{Ack}, f_{Grw}, f_{Ras} and f_{Ros}

Func	N	P	DE	DE(best)	DEfirDE	DEfirSPX
f_{Sph}	50	N	0±0 (30)	309.74±481.05 (0)	0±0 (30)	0±0 (30)
		5N	0±0 (30)	0±0 (30)	0±0 (30)	0±0 (30)
		10N	0.0535±0.0520 (0)	0.0027±0.0013 (0)	0.0026±0.0023 (0)	1.0E-4±4.75E-5 (0)
	100	N	1.58E-6±3.75E-6 (28)	0.0046±0.0247 (28)	0±0 (30)	0±0 (30)
		5N	59.926±16.574 (0)	30.242±5.93 (0)	11.731±5.0574 (0)	1.2614±0.4581 (0)
		10N	2496.82±246.55 (0)	1729.40±172.28 (0)	358.57±108.12 (0)	104.986±22.549 (0)
	200	N	50.005±16.376 (0)	26.581±7.4714 (0)	17.678±9.483 (0)	0.8568±0.2563 (0)
		5N	5.45E4±2605.73 (0)	4.84E4±1891.24 (0)	9056.0±1840.45 (0)	2782.32±335.69 (0)
		10N	1.82E5±6785.18 (0)	1.74E5±6119.01 (0)	44090.5±6122.35 (0)	9850.45±1729.9 (0)
f_{Ack}	50	N	0 ±0 (30)	0.2621 ± 0.5524 (1)	0 ± 0 (30)	0 ± 0 (30)
		5N	9.36E-6 ±3.67E-6 (0)	6.85E-6 ± 6.06E-6 (0)	2.28E-5 ± 1.45E-5 (0)	3.0E-6 ± 1.07E-6 (0)
		10N	0.0104 ±0.0015 (0)	0.0067 ± 0.0015 (0)	0.0060 ± 0.0015 (0)	0.0019 ± 4.32E-4 (0)
	100	N	1.02E-6 ±1.6E-7 (24)	9.5E-7 ± 1.1E-7 (29)	1.2E-6 ± 6.07E-7 (22)	0 ±0 (30)
		5N	1.6761±0.0819 (0)	1.2202±0.0965 (0)	0.5340±0.1101 (0)	0.3695±0.0734 (0)
		10N	7.7335±0.1584 (0)	6.7251±0.1373 (0)	3.7515±0.2773 (0)	3.4528±0.1797 (0)
	200	N	0.5208±0.0870 (0)	0.4322 ±0.0427 (0)	0.3123±0.0426 (0)	0.1589±0.0207 (0)
		5N	15.917±0.1209 (0)	15.46±0.1205 (0)	9.2373±0.4785 (0)	6.6861±0.3286 (0)
		10N	19.253±0.0698 (0)	19.138±0.0772 (0)	14.309±0.3706 (0)	9.4114±0.4581 (0)
f_{Grw}	50	N	0 ± 0 (30)	0.1651 ± 0.2133 (0)	0 ± 0 (30)	0 ± 0 (30)
		5N	9.95E-7 ± 4.3E-7 (29)	0 ± 0 (30)	0 ± 0 (30)	0 ± 0 (30)
		10N	0.0053 ± 0.010 (0)	0.0012 ± 0.0028 (0)	4.96E-4 ± 6.68E-4 (0)	5.27E-4 ± 0.0013 (0)
	100	N	0 ± 0 (30)	0 ± 0 (30)	0 ± 0 (30)	0 ± 0 (30)
		5N	1.1316 ± 0.0124 (0)	1.0530 ± 0.0100 (0)	0.7725 ± 0.1008 (0)	0.5433 ± 0.1331 (0)
		10N	20.037 ± 0.9614 (0)	13.068 ± 0.8876 (0)	3.7439 ± 0.7651 (0)	2.2186 ± 0.3010 (0)
	200	N	0.7687 ± 0.0768 (0)	0.5707 ± 0.0651 (0)	0.5984 ± 0.1419 (0)	0.1631 ± 0.0314 (0)
		5N	490.29 ± 21.225 (0)	441.97 ± 15.877 (0)	78.692 ± 11.766 (0)	28.245 ± 4.605 (0)
		10N	1657.93 ± 47.142 (0)	1572.51 ± 53.611 (0)	368.90 ± 41.116 (0)	85.176 ± 12.824 (0)
f_{Ras}	50	N	0 ± 0 (30)	0.61256 ± 1.1988(8)	0 ± 0 (30)	0 ± 0 (30)
		5N	0 ± 0 (30)	0 ± 0 (30)	0 ± 0 (30)	0 ± 0 (30)
		10N	0 ± 0 (30)	0 ± 0 (30)	0 ± 0 (30)	0 ± 0 (30)
	100	N	0 ± 0 (30)	0 ± 0 (30)	0 ± 0 (30)	0 ± 0 (30)
		5N	2.6384 ± 0.7977 (0)	0.7585 ± 0.2524 (0)	0.1534 ± 0.1240 (0)	0.0094 ± 0.0068 (0)
		10N	234.588± 13.662 (0)	198.079 ± 18.947 (0)	17.133 ± 7.958 (0)	27.0537 ± 20.889 (0)
	200	N	0.4245 ± 0.2905 (0)	0.2255 ± 0.1051 (0)	0.1453 ± 0.2771 (0)	0.0024 ± 0.0011 (0)
		5N	1878.61± 60.298 (0)	1761.55 ± 43.3824 (0)	352.93 ± 46.11 (0)	369.88 ± 136.87 (0)
		10N	5471.35± 239.67 (0)	5094.97 ± 182.77 (0)	1193.83± 145.477 (0)	859.03± 99.76 (0)
f_{Ros}	50	N	79.8921 ± 102.611 (0)	3.69E5 ± 5.011E5 (0)	72.0242 ± 47.1958 (0)	65.8951 ± 37.8933 (0)
		5N	52.4066 ± 19.9109 (0)	54.5985 ± 25.6652 (0)	53.1894 ± 26.1913 (0)	45.8367 ± 10.2518 (0)
		10N	90.0213 ± 33.8734 (0)	58.1931 ± 9.4289 (0)	66.9674 ± 23.7196 (0)	52.0033 ±13.6881 (0)
	100	N	120.917 ± 41.8753 (0)	178.465 ± 60.938 (0)	107.5604 ± 28.2529 (0)	99.1086 ± 18.5735 (0)
		5N	12312.16± 3981.44 (0)	7463.633 ± 2631.92 (0)	2923.108 ± 1521.085 (0)	732.85 ± 142.22 (0)
		10N	3.165E6 ± 6.052E5 (0)	1.798E6 ± 3.304E5 (0)	2.822E5 ± 3.012E5 (0)	16621.32 ± 6400.43 (0)
	200	N	9370.17 ± 3671.11 (0)	6725.48 ± 1915.38 (0)	5302.79 ± 2363.74 (0)	996.69 ± 128.483 (0)
		5N	4.22E8 ± 3.04E7 (0)	3.54E8 ± 3.54E7 (0)	2.39E7 ± 6.379E6 (0)	1.19E6± 4.10E5 (0)
		10N	3.29E9 ± 2.12E8 (0)	3.12E9 ± 1.65E8 (0)	3.48E8 ±1.75E8 (0)	1.21E7 ± 4.73E6 (0)

Table 2: Comparison with other RCMA models

Algorithm	Griewank	Rastrigin	Rosenbrock
CHC	6.5E-003	1.6E+001	1.9E+001
G3-1	5.1E-001	7.4E+001	2.8E+001
SW-100	2.7E-002	7.6E+000	1.0E+001
FC	3.5E-004	5.5E+000	2.3E+001
RCMA-XHC	1.3E-002	1.4E+000	2.2E+000
DE	5.1E-003	1.5E-007	3.4E+002
DE(best)	8.4E-002	6.0E-001	3.0E+004
DEfirDE	2.1E-003	1.8E-015	2.9E+002
DEfirSPX	7.6E-005	3.9E-012	6.1E+001

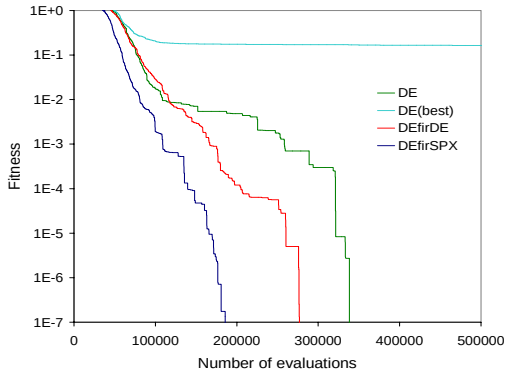

Figure 5: Griewank Function N = 50 P = 50

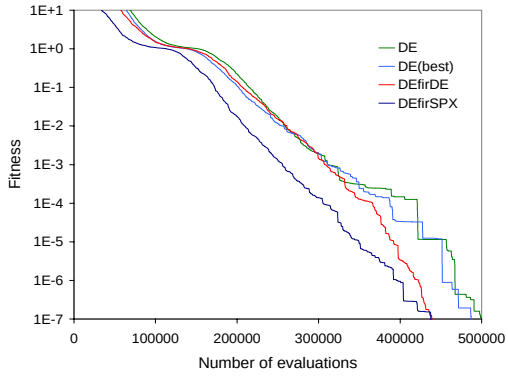

Figure 6: Griewank Function N = 50 P = 250

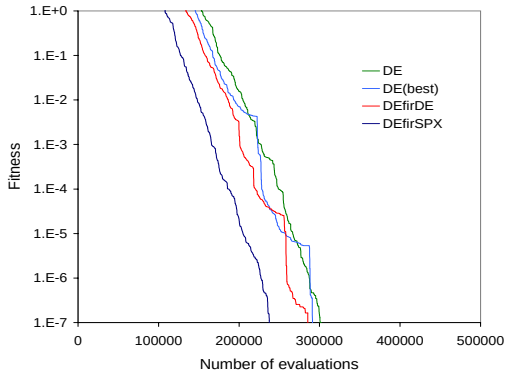

Figure 7: Rastrigin Function N = 50 P = 250

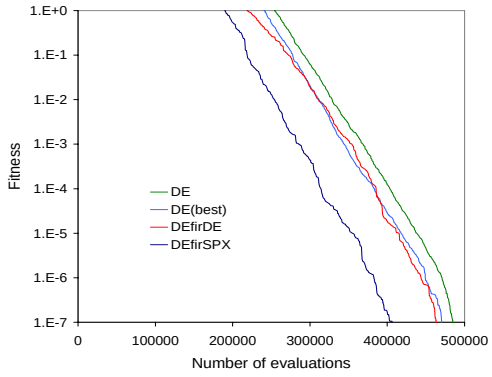

Figure 8: Rastrigin Function N = 50 P = 500

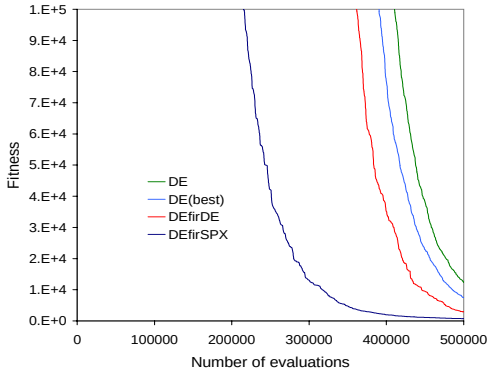

Figure 9: Rosenbrock Function N = 100 P = 500

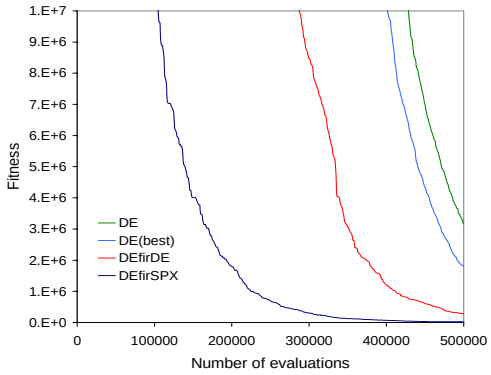

Figure 10: Rosenbrock Function N = 100 P = 1000

variation of population size we experimented with three different population sizes (P=N, P=5N and P=10N) for each dimension and function and the results are reported in Table 1. In our study we found that for high dimensional optimization if the maximum number of evaluation is fixed then the choice of population size may be crucial for the performance of DE and DE(best). For example in most cases when we found convergence for all 30 trails with a smaller population size (say P=N), we failed to reach even a single convergence with much high population size (say P = 10N). In our study we found that DE works better with a population size near to dimension of the problem (in our case P=N), but in some cases performance of DE(best) is better with a medium sized population (in our case P=5N). Since DEfirDE and DEfirSPX is just an augmentation of basic DE with XLS, it is expected that their sensitivity to the variation of population will be more or less similar to basic DE and the results of Table 1 show that. However we found the performance of DEfirDE and DEfirSPX less susceptible to population variation compared to that of DE or DE(best) (Table 1). So it can be stated that use of FIR has increased the robustness of DE against the variation of population size.

Table 2 compares the result of our second set of experiments with results reported in [8]. Here the values are the average (over 30 runs) of the best fitness function found at the end of each run. In case of Griewank and Rastrigin function the proposed DEfirDE and DEfirSPX was better than other algorithms. In case of Rosenbrock function other RC-MAs' performances were better than that of all DE based algorithms, but the results of DEfirDE and DEfirSPX are

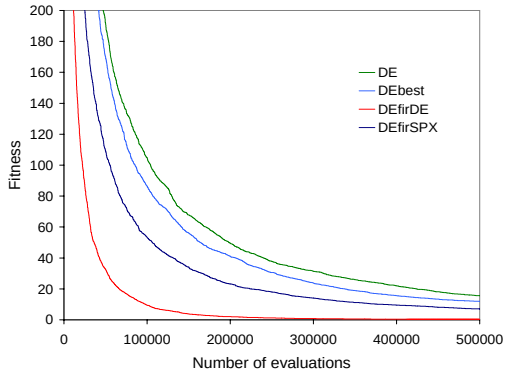

Figure 11: Mean Performance of the algorithms for $\mathbb{P}$ = 200 N = 100

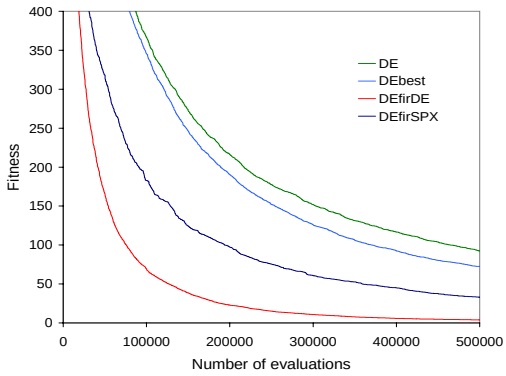

Figure 12: Mean Performance of the algorithms for $\mathbb{P}$ = 200 N = 200

comparable to that of other RCMAs. One further point about this second sets of experiments, the performances of DEfirDE and DEfirSPX were better than that of DE and DE(best) for all test functions just like in the other set of experiments.

5. EXPERIMENT WITH LANDSCAPE GENERATOR

According to No Free Lunch (NFL) theorems [15] no algorithm is superior to others when their average performance over all possible problems is considered. Therefore any general comment made about the performance of any algorithm based on results of experiments with a couple of test problems is similar to general conclusion made about a very large data set depending on a few samples; and such conclusion is often incomplete and misleading. In fact, algorithms operate on landscapes, not on problems, so the information about how a algorithm interacts with landscapes and the relationship between its performance and properties of landscapes will be more useful to predict its performance on other problems. Therefore it is more useful to use landscape generators, which do not take into account any specific problem rather landscapes on which an algorithm will conduct searching, as the ground for algorithms evaluation and testing. Another advantage of using landscape generators is that they can remove the opportunity to hand-tune algorithms to a specific problem and, by allowing a large number of problem instances to be randomly generated, the predictive power of the simulation results can be also increased [7].

Therefore we compared our proposed algorithms with their parent algorithms evaluating them using a Continuous Multimodal Landscape Generator [6]. The original landscape generator was for binary spaces, proposed in [7]. The idea is to randomly choose $\mathbb{P}$ N-bit individuals as peaks in the search space and each of these peaks have equal fitness value 1.0. Then the fitness of a string is decided by its Hamming distance to the closest peak. The continuous multimodal landscape generator is a logical extension of it for continuous space. The original model creates $\mathbb{P}$ peaks on N dimensions uniformly distributed through the interval $[0.0, 1.0]$. Individuals are evaluated using the following fitness function:

$$f(\vec{x}) = \frac{1}{N} \min_{i=1}^{\mathbb{P}} \left\{ \sum_{j=1}^{N} (sigmoid(x_j) - \mathbb{P}_i^j)^2 \right\} \qquad (10)$$

i.e. all the variables are restricted to the interval [0.0, 1.0] with a logistic transformation included as sigmoid. We have adapted the fitness function in our experiments excluding the sigmoid function as follows

$$f(\vec{x}) = \frac{1}{N} \min_{i=1}^{\mathbb{P}} \left\{ \sum_{j=1}^{N} (x_j - \mathbb{P}_i^j)^2 \right\} \qquad (11)$$

We have chosen the peaks in a wider interval [-50, 50] and allow the variables to range in this interval. So the fitness value is decided by the distance between the individual to be evaluated and the closest peak. The purpose of the search is to minimize the fitness value and the minimum value is zero. In our experiment we employed this multimodality generator to investigate the effects of number of peaks ($\mathbb{P}$) and problem dimension (N) on the performance of four algorithms, DE, DE(best), DEfirDE and DEfirSPX. We studied four scenarios $(N = 100, \mathbb{P} = 100)$, $(N = 100, \mathbb{P} = 200)$, $(N = 200, \mathbb{P} = 100)$ and $(N = 200, \mathbb{P} = 200)$. For each setting of N and $\mathbb{P}$, 20 random problems were generated on each of which each algorithm was run once. Like other experiments, all algorithms were evaluated using same random initial population. The parameter settings were $P = N, L = 25$ and the rest of the parameters were just as before. Each algorithm was allowed to evolve for maximum 500,000 times.

As expected, all of the algorithms interacted with time, problem dimension and problem difficulty. However their relative performance for each set of (N, P) was more or less consistent. Because of space, results of two experiments are shown in Figure 11 and 12. Although performance of all the algorithms faded with time, starting with a steeper curve DEfirDE and DEfirSPX attain better fitness compared to DE and DEbest. The most notable impact on the performance of the algorithm was the dimensionality of the problem. With increasing number of peaks the performance of all algorithms deteriorated except for DEfirDE which was slightly improved. Moreover the overall performance of DEfirDE was better than DEfirSPX, which is just opposite what observed in other experiments. The possible reason for substandard performance of DEfirSPX is use of crossover operation for local search. Earlier, it has been hypothesized that crossover may hurt GA performance on problems with multiple peaks [8]. However, the results of these experiments were helpful to establish our claim about use of FIR strategy for improving the performance of DE at least for highly multimodal problems.

6. DISCUSSIONS AND CONCLUSIONS

It is well known that DE is a greedy search heuristics with remarkable numerical optimization capability. But the convergence rate of DE for expensive and high dimensional objective functions is not high enough for real world situations. Therefore in an attempt to speed up DE we proposed a memetic version of DE using crossover based local search (XLS). In our proposed XLS which we call FIR, the search space around the best individual is greedily explored in each generation. To the best of our knowledge for the first time this type of XLS is being applied to any EA especially to DE.

As DE applies deterministic selection and lacks mutation operator it tries to estimate the features of the search space iteratively based on the distribution of its individuals. There are EAs available in literature which exhibit very good performance without mutation like MGG. The reason behind their success is the use of XLS whose performance depends largely on the capability of the crossover operator used. Motivated by their methodology we proposed the FIR strategy for DE algorithm. For our FIR two implementations were proposed DEfirDE and DEfirSPX where the first uses DE like recombination and the later uses simplex crossover (SPX) for searching the neighborhood of the best solution. In our experiments comparing with two well known variants of DE we found both of the schemes speed up DE for a set of well known test functions specially for higher dimensions. Between DEfirDE and DEfirSPX the overall performance of later was much better. This is expected because SPX, a much sophisticated crossover operator that works well on functions having multimodality and/or epsitasis [12], will search the neighborhood of best solution more effectively than recombination process of DE. Therefore our DEfirSPX scheme was more successful to accelerate the search and improve performance. Experimenting with various population sizes we also found that use of FIR also increases the robustness of DE against the variation of population size. In real world problems often a trade off has to be made between algorithm's convergence rate and robustness. Consequently the proposed versions of DE can be used for a higher convergence rate without sacrificing the search precision or search robustness in real world problems. Comparison with other RCMA's also proves the proposed schemes worthy.

Experimenting with random problems, yielded by a multimodal landscape generator, we verified the superiority of the proposed schemes over their parent algorithms. We found that in highly multimodal environment performance of DEfirDE was better than DEfirSPX. This is because of the adverse effect of increasing multimodality on the simplex crossover operator which is used for local search. Nevertheless our experiments showed that both FIR schemes accelerate the basic DE algorithm in highly multimodal environments.

Though in the current work we only experimented with two variants of DE, we expect that it will also able to accelerate all other variants of DE in a similar way. Some future experiment will try to verify this anticipation as well as judge the potency of FIR scheme for other EAs. Implementation of FIR with other elegant crossover operations like UNDX, some theoretical analysis of the FIR and exploring FIR on real life problems are also to be tried.

7. REFERENCES

[1] H. G. Beyer and K. Deb. On self-adaptive features in real-parameter evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 5(3):250–270, June 2001.

[2] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[3] D. E. Goldberg and S. Voessner. Optimizing global-local search hybrids. In *Genetic and Evolutionary Computation Conference (GECCO'99) Proceedings*, pages 220–228. Morgan Kaufmann, July 1999.

[4] M. Y. H. Satoh and S. Kobayashi. Minimal generation gap model for gas considering both exploration and exploitation. In *Proceedings of IIZUKA'96*, pages 494–497, 1996.

[5] A. A. Kalyanmoy Deb and D. Joshi. A computationally efficient evolutionary algorithm for real-parameter optimization. *Evolutionary Computation*, 10(4):371–395, Winter 2002.

[6] J. Kennedy. *Continuous Valued Multimodality Generator for Evolutionary Algorithms*. Retrieved from: http://www.cs.uwyo.edu/ wspears/multi.kennedy.html (16th January, 2005).

[7] J. Kennedy and W. M. Spears. Matching algorithms to problems: An experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. In *International Conference on Evolutionary Computation (ICEC'98) Proceedings*, pages 78–83. IEEE, May 1998.

[8] N. K. Manuel Lozano, Francisco Herrera and D. Molina. Real-coded memetic algorithms with crossover hill-climbing. *Evolutionary Computation*, 12(3):273–302, Fall 2004.

[9] P. Merz and B. Freisleben. Fitness landscapes, memetic algorithms, and greedy operators for graph bipartitioning. *Evolutionary Computation*, 8(1):61–91, Spring 2000.

[10] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. In *Technical Report 826, Caltech Concurrent Computation Program*. California Institute of Technology, Pasadena, California, 1989.

[11] P. Moscato and M. G. Norman. A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. In *Parallel Computing and Transputer Applications*, pages 177–186. IOS Press, Amsterdam, The Netherlands, 1992.

[12] M. Y. Shigeyoshi Tsutsui and T. Higuchi. Multi-parent recombination with simplex crossover in real coded genetic algorithms. In *Genetic and Evolutionary Computation Conference (GECCO'99) Proceedings*, pages 657–664. Morgan Kaufmann, July 1999.

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

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

[15] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[16] J. M. Yang and C. Y. Kao. Integrating adaptive mutations and family competition into genetic algorithms as function optimizer. *Soft Computing*, 4(2):89–102, June 2000.