Chapter 4

Enhancing an Evolutionary Optimizer

The optimization task for estimating parameters of S-system is a real difficult one. Therefore, in this chapter the performance of an evolutionary algorithm namely differential evolution was enhanced by hybridizing with a local search strategy. Starting with the original algorithm the chapter presents the proposed new evolutionary algorithm. Then extensively study the performance of the algorithm using benchmark problems, random problems from landscape generators and finally study the performance of the algorithm for the basic MSE based fitness function of (3.8).

Because of some inherent advantages, like robustness, parallelism, global search capability, ease of implementation, etc., Evolutionary Algorithms (EAs) have become very popular for nonlinear optimization. However, hybridization with other strategies, such as meta-heuristics or local searches, can improve the efficiency of the evolutionary search [19]. GAs hybridized with local refinement procedures are commonly known as Memetic Algorithms (MAs) [78, 79].

MAs are motivated to provide an effective and efficient 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) [69]. 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 the population or with some specific probability and with various replacement strategies.

The other group employs crossover operators for local refinement. The crossover operator is a recombination operator that produces offspring around the parents.

For this reason, it may be considered to be a move operator for an LS strategy [69]. Over the past few years, much research effort has been spent on developing efficient crossover operators for real parameters that use probability distributions around the parents for creating offspring adaptively. Among numerous approaches, the BLX-crossover, the simulated binary crossover (SBX), the unimodal normal distribution crossover (UNDX), the simplex crossover (SPX), the parent centric crossover (PCX) deserve specific mention. And this type of crossover operators can be employed to create offspring distributed densely around the parents, favoring local tuning. In order to make the best use of this characteristic of these crossover operators a specific model of GA has also been suggested. This class of GA actually performs a local search by generating many offspring around the parents, mating the participating parents repeatedly using the crossover operator. The most common examples of XLS based MAs in literature are Minimal Generation Gap (MGG) [109] and Generalized Generation Gap (G3) [22]. Both of them employ same parents to spawn multiple offspring. The idea is to induce an LS on the neighborhood of the parents involved in crossover. The above mentioned crossover operators with inherent adaptive nature show promise for building effective XLS for continuous search space [69] and has received much attention recently.

Differential Evolution (DE), proposed by Storn and Price [123], is an effective, efficient and robust optimization method capable of handling nonlinear and multi-modal 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 [123, 122].

One key problem of all search algorithms is the "curse of dimensionality" [12]. This expression refers to the exponential growth of the search space's volume as a function of dimensionality. Thus even if there is an evolutionary computational solution to a problem of a particular size, the same problem with increased dimension might be completely intractable. And DE can not escape this curse. 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 [29, 86]. In this chapter, 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.

4.1 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 the combination of randomly chosen individuals from population. Specifically, for each individual x_i^G, $i = 1, \cdots, P$, where G denotes the current generation, a *mutated* individual y_i^G is generated according to the following equation

$$v_i^G = x_j^G + F(x_k^G - x_l^G) \tag{4.1}$$

where j, k and l are random integers such that j, k and $l \in \{1, \cdots, P\}$ 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 v_i^G is mated with x_i^G using a *crossover* operation to generate the *offspring* or *trial individual* u_i^G. Generally *binomial* or *exponential* crossover operations are used in DE. The genes of u_i^G are inherited from x_i^G and v_i^G, determined by a parameter called *crossover factor/rate* (CR). In exponential crossover CR regulates how many consecutive genes of the mutated individual on average are copied to the offspring. And in binomial crossover CR stochastically determines the source of each trial parameter from x_i^G and v_i^G. Storn and Price [123], suggested the values for F and CR are to be chosen from $F \in [0, 2]$ and $CR \in [0, 1]$ respectively. Finally, the offspring is evaluated and replaces its parent x_i^G 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 for mutation 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* when exponential crossover is used. 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 Eq. (4.1) by Eq. (4.2), (4.3) or (4.4) respectively. Also each of them has a mate based on binomial crossover.

$$v_i^G = x_{best}^G + F(x_j^G - x_k^G) \tag{4.2}$$

$$v_i^G = x_j^G + F(x_k^G - x_l^G) + F(x_m^G - x_n^G) \tag{4.3}$$

$$v_i^G = x_{best}^G + F(x_j^G - x_k^G) + F(x_l^G - x_m^G) \tag{4.4}$$

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

4.2 DE with Crossover based Local Search (XLS)

As mentioned earlier, XLSs are applied to search the neighborhood of the parents locally to improve the fitness of the parents. The same strategy has been applied to the neighborhood of a single individual from each generation. In other words it can be said, XLS was used 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 [145], where they search the neighborhood of each individual and they have named it *Family Competition* (FC).

In this proposed variant of DE with XLS, the basic DE (*DE/rand/1/exp*) is extended by applying some crossover based local search (XLS) for exploring the neighborhood of the best individual. That is, in this XLS procedure the best individual x_{best}^G 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 offspring and family father x_{best}^G the individual with the best score replaces the family father in next generation. It is named as crossover based local search for *Fittest Individual Refinement* (FIR). In this chapter DE algorithm was augmented by FIR in the general template of MA and the new algorithm was named as DEFIR. The formal algorithm for DEFIR can be described, as follows

DEFIR

```
1. Generate an Initial Population $P^G$
2. $P^G$ = FIR ($P^G$)
3.     $P^{G+1} = \{\phi\}$
4.     REPEAT for each individual $I \in P^G$
```

```
5.            Reproduce an offspring $J$ from $I$
6.            $P^{G+1} = P^{G+1} \cup$ Best $(I, J)$
7.        $P^{G+1} =$ FIR $(P^{G+1})$
8. REPEAT Step 3 to 7 Until Search Converged
```

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 two schemes namely DEfirDE and DEfirSPX are proposed.

In FIR strategy of DEfirDE scheme, the offspring is generated using the recombination operation of *DE/rand/1/exp*. In each generation G, for the best individual x_{best}^G three individuals x_j^G, x_k^G and x_l^G were selected such that j, k and $l \in \{1, \cdots, P\}$ and $best \neq j \neq k \neq l$. Then a mutated individual v_i^G is generated using Eq. (4.1). Finally, the offspring C is generated by crossover operation between mutated individual v_i^G and the best individual x_{best}^G . This procedure is repeated L times and then *selection* is performed.

On the other hand, for local search DEfirSPX scheme generates the offspring using *simplex crossover* (SPX) operation propsed by Tsutsui *et al.* [131]. SPX has various good characteristics, e.g. it does not depend on a coordinate system, the mean vector of parents and offspring generated with SPX are the same and SPX can preserve a covariance matrix of the population with an appropriate parameter setting. These properties make SPX a suitable operator for neighborhood search. Besides a preliminary study [86], had found SPX as a promising operation for local tuning. More details about the SPX crossover can be found in [131].

In FIR of DEfirSPX scheme the best individual and other $(p-1)$ random individuals are selected from the 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. In MAs often LS is applied to a selected portion of the population. This is because each application of the LS for exploring the neighborhood of an individual requires additional function evaluations. But there is no straightforward way of selecting the most promising individual which will undergo LS. However, the solution with best fitness value possibly is in the proximity of a promising basin of attraction. Therefore, some extra fitness evaluations can be allowed to search the neighborhood of this solution for a better solution. DE applies a more directive search in a greedy way. Augmenting this FIR process in the structure of DE the search was made more directive or greedier in a sense. In the context of continuous search space, it can be assumed that the solutions close to the current best individual are also potential good candidates that form the neighborhood of the current best solution. With the progress of the search, by exploring the potential candidate solutions in the vicinity of the best individual, it is expected to reach 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*. Using the above analysis it can easily be observed that the FIR strategy makes DE greedier by replacing the fittest solution of the generation with the best individual in its neighborhood through local refinement and thus accelerates the search.

In designing a XLS method several decisions involved, like LS application strategy, length of the LS, selecting parents for crossover operation. In this work a very simple approach has been selected for applying the FIR strategy, i.e. only on the best individual of each generation. However, the FIR strategy can also be applied selectively where some other random individuals also will undergo LS. The length L of the FIR strategy was kept fixed, whereas it can be varied dynamically with the progress of the search. In the beginning of the search, the populations is randomly scattered in the search space an extended local search will be useful to identify a better solution, but in later generations, as the individuals come closer, brief local searches will suffice to explore the neighborhood. In current implementation the other parents participating in FIR strategy were selected randomly. In order to promote high degree of population diversity, other mating schemes such as *assortative mating*, or roulette wheel based selection can be used. In current implementation it was tried to keep the model simplest and in every case the most straight forward decision was made.

4.3 Experiments with Benchmark Functions

In this section, the results of different experiments on the test suite described in Section 4.3.1 are presented. These minimization experiments were carried out in order to analyze the performance of the proposed DEfirDE and DEfirSPX algorithms comparing with DE. As mentioned in Section 4.2, DEfirDE and DEfirSPX algorithms are implemented by augmenting FIR with *DE/rand/1/exp*.

4.3.1 Test Suite

Five test functions commonly found 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}) were used. All benchmarks chosen are minimization problems. The definitions of the functions are as follows

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

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

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

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

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

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.

4.3.2 Experimental Setup

In this section, the general setup for different experiments in which the performance of DE, DEfirDE and DEfirSPX were investigated are specified. Targeting high dimensional optimization, N= 100 was chosen for most experiments. In order to make the performance comparison fairer the same sets of initial random populations were used for evaluating all algorithms. Each experiment was repeated 30 times. Maximum number of evaluations allowed for each algorithm was 500,000. For DEs most commonly used parameter setting were chosen and was not tuned to the best parameter values for each problem. The value of F was set to 0.5 and CR was chosen 0.8. Based on some preliminary experiments for FIR algorithms, $L = 10$ was chosen[85]. For simplex crossover operation, the number of parents $p = 3$ was used. The algorithms were evaluated by calculating average (AVG) and standard deviation (SD) of their attained minimum fitness value within the maximum number of allowed evaluations. The experiments were performed on a computer with 1700 MHz Intel Pentium processor and 512 MB of RAM in JBuilder X environment.

Notation: In different tables the data is presented using the following three notations:

1. **AVG ± SD** : represents the average and standard deviation of the best fitness values obtained after 500,000 evaluations of different trials.
2. **0.0 [AVG ± SD]** : represents the average and standard deviation of the number of fitness evaluation required to reach to optimal value (when all trails reach optimal).
3. **AVG ± SD (c %)** : represents the average and standard deviation of the best fitness values obtained after 500,000 evaluations and the percentage of trails that reached optimal value.

In all tables the best results are shown in bold letters.

4.3.3 Effect of Problem Dimensionality

As specified earlier, the proposed DEfirDE and DEfirSPX schemes are intended to speed up DE in high dimensional optimization because the original problem of genetic network inference using S-system is a high dimensional problem. Therefore, in the experiments presented in this section, the effect of problem dimensionality on the performance of DE, DEfirDE and DEfirSPX algorithms was studied. Here, the benchmarks were studied in six dimensions N=50, 100, 200, 300, 400, and 500 with population size P=N. Other parameter settings are the same as mentioned in section 4.3.2. In these experiments if the fitness value was less than 10^{-6} range of actual optimum point, it is assumed solution was detected. The results are shown in Table 4.1 and some representative graphs from different functions in different dimensions are presented to illustrate the convergence properties of the algorithms.

Inspecting Table 4.1 it can be found that DEfirDE and/or DEfirSPX strategy succeeded to reach to the optimal value in all trails for some cases (e.g. f_{Sph} (N=100), f_{Ack} (N=100)) in which DE failed for some trials (also showed in Fig.

4.1(a) and 4.1(b)). Even in cases, where DE could reach the optimal value in all trails it took higher number of function evaluations compared to that needed for DEfirDE and/or DEfirSPX. This can also be verified looking at the graphs of Fig. 4.1(c) and 4.2(a). And if the cases, where none of the four schemes could hit the global optimal, are considered, then it is found that the proposed DEfirDE and DEfirSPX scheme attained AVGs which are significantly better than that achieved by their parent algorithm (Fig. 4.2(b)). For example in Rosenbrock's 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 average minimum fitness values. Another observation from Table 4.1 is that with the increase of dimensionality the minimum fitness achieved by all the algorithms after 500,000 fitness evaluations increased polynomially. Nevertheless, with the increase of dimension, the performance difference between the proposed schemes and their parent scheme becomes more significant. And this testifies the claim that the proposed FIR schemes will speedup DE for higher dimensional function optimization.

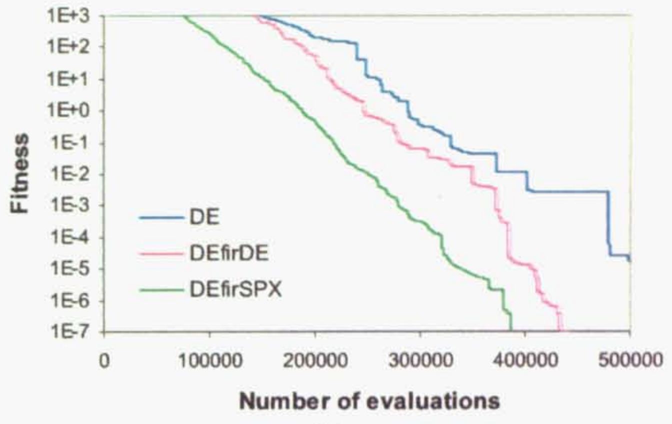
Looking at the graphs of Fig. 4.1 and 4.2, it can be found that in every case DEfirSPX started with the steepest convergence curve and continued to produce best fitness value compared to other strategies with the progress of the search. In the beginning of the search SPX performs local searching using individuals randomly scattered in the search space and becomes very successful in generating offspring with high fitness, but at later generations it generates individuals around the dense 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 towards DEfirSPX steadily. Results presented in Table 4.1 imply that, for both multimodal (f_{Grw}, f_{Ras} and f_{Ack}) and unimodal (f_{Sph} and f_{Ros}) functions DEfirDE and DEfirSPX exhibited superior performance compared to DE.

4.3.4 Sensitivities to Control Parameters

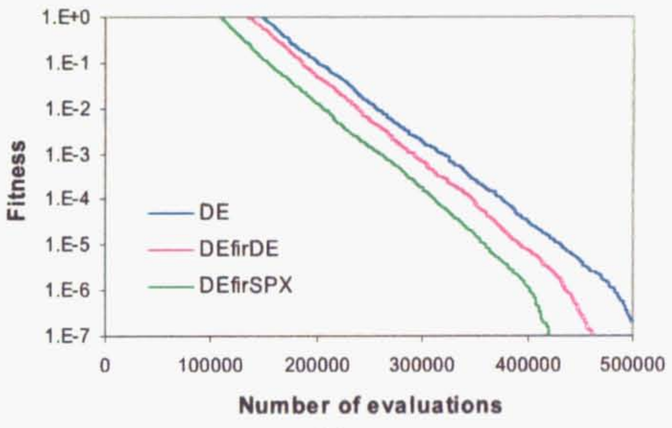
To assess the robustness of the proposed schemes to the variation of population size and other control parameters some experiments comparing with original DE were accomplished. During the experimentation, one of the three parameters, P, CR and F was varied within a certain range keeping the remaining two constant to their values specified in Section 4.3.2. For all the experiments in this section, the problem dimension N=100 was selected. The results are shown in Table 4.2 to 4.4.

Table 4.1: The effect of problem dimensionality (N)

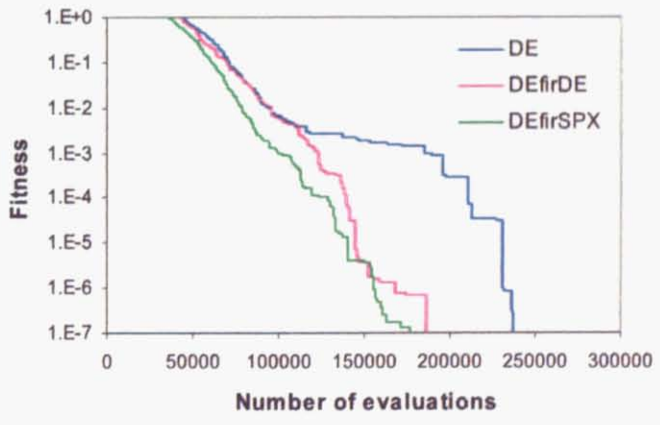
	N	DE	DEfirDE	DEfirSPX
f_{Sph}	50	0.0 [145703.7 ± 22113.7]	0.0 [137500.5 ± 31588.9]	**0.0 [101649.4 ± 14842.6]**
	100	1.75E-05 ± 8.95E-5 (93.3%)	0.0 [392370.7 ± 27285.6]	**0.0 [345124.5 ± 22515.7]**
	200	50 ± 16.38	36.67 ± 20.49	**1.503 ± 0.846**
	300	7950.1 ± 866.86	2155.12 ± 374.88	**281.77 ± 66.03**
	400	71987.92 ± 3678.58	20752.26 ± 2406.303	**3085.62 ± 289.15**
	500	258627.11 ± 7064.83	79685.98 ± 6286.47	**11576.36 ± 1530.79**
f_{Ack}	50	0.0 [161401.9 ± 17471.5]	0.0 [153469.2 ± 20731.4]	**0.0 [138447.63 ± 9712.05]**
	100	1.08E-06 ± 4.87E-7 (90%)	9.37E-07 ± 1.23E-7 (96.7%)	**0.0 [404276.2 ± 10553.6]**
	200	0.452 ± 0.0486	0.146 ± 0.021	**0.049 ± 0.0055**
	300	7.015 ± 0.107	4.39 ± 0.14	**3.02 ± 0.106**
	400	14.21 ± 0.083	9.7 ± 0.27	**5.84 ± 0.241**
	500	17.87 ± 0.075	13.75 ± 0.26	**7.942 ± 0.316**
f_{Grw}	50	0.0 [129545.0 ± 28418.3]	0.0 [129092.6 ± 23006.7]	**0.0 [114993.8 ± 17656.1]**
	100	0.0 [340765.7 ± 35477.28]	0.0 [318334.93 ± 43484.48]	**0.0 [297807.5 ± 50001.4]**
	200	0.78 ± 0.08	0.229 ± 0.0838	**0.052 ± 0.028**
	300	41.409 ± 1.54	10.52 ± 1.17	**3.167 ± 0.256**
	400	603.47 ± 16.84	168.21 ± 18.2	**33.92 ± 4.059**
	500	2317.95 ± 55.5	701.12 ± 51.97	**115.803 ± 9.332**
f_{Ras}	50	0.0 [98840.0 ± 26608.15]	0.0 [98333.7 ± 29017.8]	**0.0 [79041.6 ± 14065.3]**
	100	0.0 [261150.7 ± 17976.1]	0.0 [243793.0 ± 24529.1]	**0.0 [204120.8 ± 19879.2]**
	200	0.395 ± 0.168	0.0377 ± 0.0362	**4.54E-04 ± 6.65E-04**
	300	674.018 ± 28.33	116.98 ± 21.22	**50.66 ± 93.97**
	400	2864.027 ± 53.045	867.25 ± 90.87	**400.94 ± 179.87**
	500	7651.92 ± 245.1	2574.38 ± 122.86	**1353.606 ± 191.199**
f_{Ros}	50	79.89 ± 102.61	72.467 ± 75.578	**68.079 ± 68.079 42.34**
	100	130.09 ± 47.83	119.896 ± 37.426	**107.82 ± 107.82 26.94**
	200	9370.17 ± 3671.11	4581.29 ± 2959.51	**1483.42 ± 1483.42 472.46**
	300	1.98E+07 ± 5278030.01	3580853.11 ± 2240402.71	**63937.322 ± 63937.322 17040.933**
	400	4.88E+08 ± 5.74E+07	6.54E+07 ± 1.66E+07	**1197718.51 ± 1197718.51 330262.21**
	500	3.55E+09 ± 2.34E+08	4.98E+08 ± 9.53E+07	**8541688.2 ± 8541688.2 1477062.22**



(a)

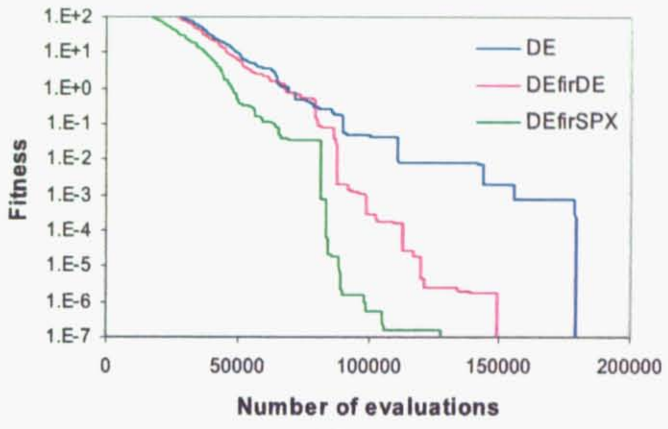


(b)

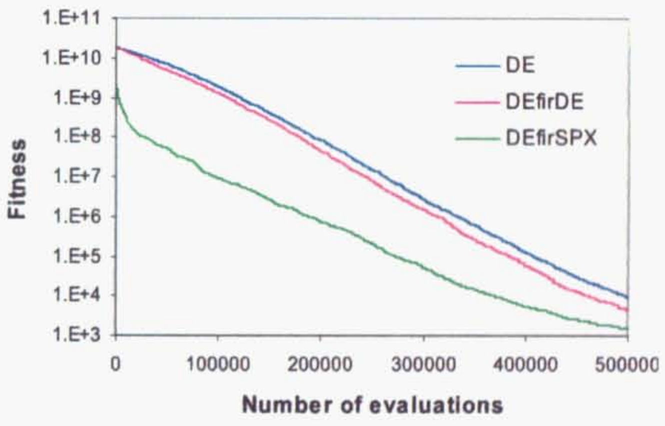


(c)

Figure 4.1: Convergence Graphs for (a) Sphere function N=100 (b) Ackley's function N=100 and (c) Griewank's function N=50



(a)



(b)

Figure 4.2: Convergence Graphs for (a) Rastrigin's function N=50 and (b) Rosenbrock's function N=200

Storn and Price suggested a larger population size (between 5N to 10N) for DE [123], although later many others found DE's performance good even with a smaller population. In this study it was 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 [91]. For example, consider the case of f_{Ras}: where with a smaller population size (say P=N=100) convergence was found for all 30 trails, but DE failed to reach even below the fitness value 10 with much high population size (say P = 7N or P=10N). In this study, it was found that in high dimensional search spaces DE works better with a population size near to dimension of the problem (in this case P=N). Since DEfirDE and DEfirSPX are just augmentations 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 4.2 show that. However, the performance of DEfirDE and DEfirSPX were found less susceptible to population variation compared to that of DE (Table 4.2). So it can be stated that use of FIR has increased the robustness of DE against the variation of population size.

Experimental results obtained varying the *crossover rate* (CR) and *amplification factor* (F) are shown in Table 4.3 and 4.4 respectively. From these tables it can be seen that at any parameter setting the FIR scheme can be effective and DEFIR can produce better results than basic DE. So it can be stated that DEFIR schemes show lower sensitivity to the variations in control parameters than the original DE does. Among the proposed two schemes DEfirSPX is found to be more robust to the changes of parameters. DEfirDE scheme also exhibited less sensitivity compared to DE against parameters changes. Only for F=0.1 and some cases of CR=1.0 setting DEfirDE's performance was slightly inferior compared to that of DE and for all other cases it showed superior performance. So it can be concluded that for all control parameters, P, CR, and F, DEFIR schemes showed less performance fluctuation compared to DE in an overall. In other words, because of hybridization with FIR strategy the overall robustness of DE has improved.

4.3.5 Comparison with Other Hybrid GAs

In this subsection, a performance comparison, among the proposed algorithms and some other hybrid GAs that have been proposed for solving real-parameter optimization problems, is presented. Since MGG and G3 are two of the most prominent models of EAs and one of the proposed enhancement makes explicit use of simplex crossover operator it is well justified to compare the proposed algorithms

Table 4.2: Sensitivity to the population size (P)

	N	DE	DEfirDE	DEfirSPX
f_{Sph}	100	9.15E-9 ± 3.04E-8	1.922E-10 ± 4.415E-10	**1.562E-12 ± 5.995E-12**
	400	9.23 ± 2.92	2.225 ± 1.159	**0.247 ± 0.100**
	700	443.79 ± 36.75	113.823 ± 29.070	**18.665 ± 7.59**
	1000	2495.99 ± 233.25	694.734 ± 163.311	**140.723 ± 38.049**
f_{Ack}	100	2.057E-7 ± 8.26E-8	1.668E-7 ± 1.507E-7	**8.556E-8 ± 7.195E-8**
	400	0.336 ± 0.023	0.128 ± 0.027	**0.055 ± 0.0133**
	700	4.159 ± 0.069	2.625 ± 0.140	**1.687 ± 0.152**
	1000	7.735 ± 0.153	5.163 ± 0.296	**3.568 ± 0.195**
f_{Grw}	100	1.918E-9 ± 9.059E-9	4.802E-12 ± 1.796E-11	**4.437E-14 ± 1.64E-13 (30%)**
	400	0.652 ± 0.069	0.195 ± 0.065	**0.030 ± 0.0126**
	700	3.319 ± 0.206	1.613 ± 0.128	**1.137 ± 0.027**
	1000	19.586 ± 1.357	6.430 ± 1.039	**2.286 ± 0.294**
f_{Ras}	100	0.0 [356115.4 ± 13528.2]	0.0 [330468.7 ± 18951.11]	**0.0 [284680.4 ± 15508.1]**
	400	0.0503 ± 0.022	0.003 ± 0.002	**4.17E-5 ± 3.36E-5**
	700	61.798 ± 7.422	8.383 ± 3.480	**1.716 ± 1.67**
	1000	232.302 ± 17.008	50.41 ± 12.10	**36.307 ± 39.07**
f_{Ros}	100	130.094 ± 47.826	119.896 ± 37.427	**107.823 ± 26.943**
	400	5734.962 ± 1877.322	1975.71 ± 874.12	**758.604 ± 228.396**
	700	334837.399 ± 140158.682	70747.33 ± 30688.68	**5446.84 ± 2115.32**
	1000	4174082.14 ± 864876.99	756465.43 ± 318283.99	**34114.58 ± 12611.29**

Table 4.3: Sensitivity to the crossover rate (CR)

	N	DE	DEfirDE	DEfirSPX
f_{Sph}	0.1	1487.44 ± 2300.48	1132.47 ± 1238.66	**1.94 ± 3.83**
	0.3	173.65 ± 315.43	102.60 ± 195.24	**0.069 ± 0.29**
	0.6	0.028 ± 0.06	0.0199 ± 0.095	**3.35E-8 ± 1.75E-7**
	0.8	9.15E-9 ± 3.04E-8	1.922E-10 ± 4.41E-10	**1.56E-12 ± 5.99E-12**
	1	14276.60 ± 3066.88	14121.71 ± 2934.80	**12340.09 ± 3133.27**
f_{Ack}	0.1	0.017 ± 0.015	0.014 ± 0.026	**0.002 ± 0.003**
	0.3	0.007 ± 0.014	9.85E-4 ± 7.57E-4	**1.87E-4 ± 1.33E-4**
	0.6	2.16E-4 ± 7.80E-4	2.12E-5 ± 4.68E-5	**1.57E-6 ± 2.43E-6**
	0.8	2.06E-7 ± 8.26E-8	1.67E-7 ± 1.51E-7	**8.56E-8 ± 7.2E-8**
	1	11.73 ± 0.536	11.79 ± 0.699	**11.36 ± 0.83**
f_{Grw}	0.1	0.006 ± 0.021	0.0026 ± 0.008	**0.0018 ± 0.0055**
	0.3	0.0015 ± 0.0046	2.79E-4 ± 7.47E-4	**5.42E-5 ± 1.12E-4**
	0.6	2.97E-5 ± 1.36E-4	1.25E-5 ± 6.297E-5	**7.38E-6 ± 2.77E-5**
	0.8	1.92E-9 ± 9.06E-9	4.802E-12 ± 1.796E-11	**4.43E-14 ± 1.64E-13 (30%)**
	1	116.94 ± 23.94	128.65 ± 29.98	**113.11 ± 19.21**
f_{Ras}	0.1	3.325E-4 ± 0.0017 (6.7		
	0.3	1.71E-7 ± 8.22E-7 (16.7%)	7.74E-8 ± 2.75E-7 (40%)	**1.84E-12 ± 9.25E-12 (80%)**
	0.6	6.22E-15 ± 2.396E-14 (93.3%)	0.0 [389469.9 ± 26722.2]	**0.0 [356382.9 ± 29942.0]**
	0.8	0.0 [356115.4 ± 13528.2]	0.0 [330468.7 ± 18951.1]	**0.0 [284680.4 ± 15508.1]**
	1	305.036 ± 41.46	**304.47 ± 34.07**	311.02 ± 37.87
f_{Ros}	0.1	2999490.58 ± 2659045.68	2055312.81 ± 2805028.96	**52458.93 ± 34150.37**
	0.3	187368.44 ± 308193.45	147354.10 ± 246626.38	**4984.48 ± 5090.42**
	0.6	677.16 ± 580.44	387.42 ± 162.55	**235.44 ± 116.89**
	0.8	130.09 ± 47.83	119.896 ± 37.43	**107.82 ± 26.94**
	1	2.508E7 ± 7974512.96	2.64E7 ± 1.01E7	**2.501E7 ± 1.01E7**

Table 4.4: Sensitivity to the amplification factor (F)

	N	DE	DEfirDE	DEfirSPX
f_{Sph}	0.1	10723.19 ± 4324.80	11191.92 ± 4148.83	**4238.12 ± 1905.65**
	0.5	9.15E-9 ± 3.04E-8	1.92E-10 ± 4.41E-10	**1.56E-12 ± 5.99E-12**
	0.8	0.0086 ± 0.0070	0.0042 ± 0.0046	**1.396E-4 ± 3.25E-4**
	1.2	29.59 ± 12.85	22.397 ± 14.07	**0.982 ± 0.545**
	1.6	812.064 ± 264.901	540.29 ± 337.85	**29.54 ± 11.566**
	1.9	3407.372 ± 1091.70	2006.70 ± 802.73	**115.80 ± 55.76**
f_{Ack}	0.1	4.367 ± 0.779	5.72 ± 1.137	**3.708 ± 0.681**
	0.5	2.06E-7 ± 8.26E-8	1.67E-7 ± 1.51E-7	**8.556E-8 ± 7.195E-8**
	0.8	0.0070 ± 0.0018	0.0040 ± 0.0015	**6.52E-4 ± 1.94E-4**
	1.2	0.94 ± 0.188	0.578 ± 0.1598	**0.125 ± 0.041**
	1.6	4.25 ± 0.27	2.371 ± 0.264	**1.53 ± 0.182**
	1.9	6.68 ± 0.429	3.72 ± 0.471	**2.67 ± 0.254**
f_{Grw}	0.1	11.56 ± 6.388	17.314 ± 6.943	**10.58 ± 8.96**
	0.5	1.918E-9 ± 9.059E-9	4.80E-12 ± 1.796E-11	**4.427E-14 ± 1.64E-13 (30%)**
	0.8	0.0024 ± 0.0053	0.0017 ± 0.0036	**1.39E-4 ± 6.31E-4**
	1.2	0.665 ± 0.113	0.395 ± 0.123	**0.087 ± 0.071**
	1.6	1.88 ± 0.251	1.284 ± 0.136	**0.978 ± 0.077**
	1.9	6.201 ± 1.019	2.642 ± 0.910	**1.225 ± 0.069**
f_{Ras}	0.1	11.46 ± 4.72	29.139 ± 18.796	**9.55 ± 4.35**
	0.5	0.0 [356115.4 ± 13528.2]	0.0 [330468.7 ± 18951.1]	**0.0 [284680.4 ± 15508.1]**
	0.8	6.52E-9 ± 2.088E-8	1.610E-9 ± 2.659E-9	**2.212E-12 ± 3.176E-12**
	1.2	0.038 ± 0.029	0.032 ± 0.104	**8.232E-5 ± 8.50E-5**
	1.6	14.467 ± 5.206	3.0132 ± 2.48	**1.141 ± 3.54**
	1.9	63.978 ± 16.173	**14.468 ± 5.584**	18.61 ± 20.57
f_{Ros}	0.1	1.91E7 ± 1.4921E7	3.987E7 ± 2.23E7	**1.073E7 ± 8120281.74**
	0.5	130.094 ± 47.83	119.896 ± 37.43	**107.82 ± 26.94**
	0.8	304.12 ± 125.53	281.046 ± 120.70	**144.22 ± 50.60**
	1.2	37509.90 ± 18865.15	21902.69 ± 13750.01	**2521.26 ± 1196.97**
	1.6	2848245.75 ± 1740160.20	1342155.42 ± 971885.38	**34883.58 ± 46325.42**
	1.9	1.35E7 ± 8343132.99	5765649.67 ± 3514865.16	**102106.28 ± 55769.57**

Table 4.5: Comparison with other algorithms (N=100)

Alg	f_{Sph}	f_{Ack}	f_{Grw}
MGG+SPX	337.312 ± 45.66	3.76 ± 0.148	3.88 ± 0.401
G3+SPX	286.69 ± 41.66	3.67 ± 0.141	3.676 ± 0.33
RCMA+XHC	0.0 [421451.0 ± 22622.9]	4.24E-4 ± 2.82E-4	0.027 ± 0.0461 (56.67%)
DE	1.75E-5 ± 8.95E-5 (93.3%)	1.08E-6 ± 4.87E-7 (90%)	0.0 [340765.7 ± 35477.28]
DEfirDE	0.0 [392370.7 ± 27285.6]	9.37E-7 ± 1.23E-7 (96.7%)	0.0 [318334.93 ± 43484.48]
DEfirSPX	0.0 [345124.5 ± 22515.7]	0.0 [404276.2 ± 10553.6]	0.0 [297807.5 ± 50001.4]
Alg	f_{Ras}	f_{Ros}	-
MGG+SPX	4.014 ± 0.744	25897.84 ± 6601.92	
G3+SPX	3.13 ± 0.941	18773.114 ± 4812.72	
RCMA+XHC	0.0 [355635.8 +/- 43611.3]	157.268 ± 73.192	
DE	0.0 [261150.7 ± 17976.1]	130.09 ± 47.83	
DEfirSPX	0.0 [243793.0 ± 24529.1]	119.896 ± 37.426	
DEfirSPX	0.0 [204120.8 ± 19879.2]	107.82 ± 26.94	

with MGG+SPX (MGG with Simplex crossover) and G3+SPX (G3 with Simplex crossover). Recently Lozano *et al.* [69] have proposed a real coded memetic algorithm (RCMA) that uses a real-parameter crossover hill-climbing (XHC) and using an extensive study the authors have shown that their proposed RCMA+XHC algorithm can outperform or competitive with many well known models of RCMAs like, Hybrid steady-state RCMA, G3, Family Competition (FC), Hybrid CHC. Therefore, this study also included this model of RCMA+XHC for comparing with the proposed algorithms. All the benchmarks were studied at dimension N=100. The control parameters of each algorithm were set according to the recommended parameters of each algorithm. For G3 and MGG population size P = 1500, the number of children generated by the simplex crossover per selection = 50, the number of parents participated in the crossover operation = N+1. For RCMA+XHC algorithm, P=300 was chosen and all other parameters are set to the values used in [69]. Maximum function evaluation allowed for each algorithm was 500,000 and the optimum was considered to be found only when the best fitness value reached less than 10^{-6}.

Table 4.5 shows the results of comparison. In this study none of the MGG or G3 models with simplex crossover could reach the global optima for any of the objective functions in any run using 500,000 function evaluations. Comparatively the performance of RCMA+XHC was better. However, RCMA+XHC could perform better than DE only for the simplest function f_{Sph}. And both of the proposed algorithms DEfirSPX and DEfirDE outperform all the three algorithms for each benchmark function.

4.4 Experiments with Landscape Generator

According to No Free Lunch (NFL) theorems [139] 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 an 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 [149]. 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. Furthermore, by allowing a large number of problem instances to be randomly generated, the predictive power of the simulation results can be also increased [52].

4.4.1 The Gaussian Landscape Generator

The proposed algorithms were compared with their parent algorithm evaluating them using a Continuous Landscape Generator proposed by Yuan and Gallagher [149]. The basic component of this landscape generator is N-dimensional Gaussian Function (GF) and each landscape contains M GFs, each constituting a "hill" in the landscape. The fitness of an individual is given by the maximum value of any of the Gaussian components at that point. So individuals are evaluated using the following fitness function

$$F(X) = \max_{i=1}^{M} \left[\frac{1}{(2\pi)^{n/2} \mid \sum_i \mid^{1/2}} exp\left(-\frac{1}{2}(X-\mu_i){\sum}_i^{-1}(X-\mu_i)^T \right) \right] \tag{4.5}$$

where X is the N-dimensional individual to be evaluated and $\sum_i$ and μ_i are the N-dimensional vector of mean and $(N \times N)$ dimensional covariance matrix of i-th $[i = 1, \cdots, M]$ Gaussian component respectively. According to Eq. (4.5), given a point X in the search space, its value is calculated regard to each GF and its fitness is set to the highest value returned among all GFs. So, the mean vector of each GF corresponds to an optimum and the mean vector of the GF that has the highest peak value corresponds to the global optimum.

The multivariate Gaussian functions often produce very small values (i.e. probalitity) in high dimensional spaces. Since test problems in high dimensional spaces were desirable the original fitness function was transformed as follows

$$F(X) = \max_{i=1}^{M} \left\{ \left[\frac{1}{(2\pi)^{n/2} \mid \sum_i \mid^{1/2}} exp\left(-\frac{1}{2}(X-\mu_i)\sum_i^{-1}(X-\mu_i)^T \right) \right]^{1/n} \right\} \quad (4.6)$$

So the parameters of the landscape generator are the number of GFs (M), the dimensionality of the landscape (N), the range of the search space, the mean vector and the covariance matrix of each GF. Again a Gaussian with arbitrary valid covariance structure can be conveniently generated through a series of rotations of the variable coordinate system [108]. Hence, each $\sum_i$ is parameterized using rotation angles and variance values. Using these parameters the landscape can be tuned to have specific geometric characteristics e.g. hills, valleys or other landscape features.

4.4.2 Experiment Setup

As specified earlier, the motivation of experimenting with landscape generator is to gain insights about the performance of different algorithms evaluating them using random test problems. In these experiments the above specified landscape generator was employed for investigating the performance of DE, DEfirDE and DEfirSPX using one set of randomly generated problem instances with similar structural features. Here the dimensionality N=50 and number of GFs M=100 was chosen. The rest of the parameters were chosen randomly. The search space was restricted to the interval [-10.0, 10.0] in each dimension. For covariance matrices the range of variance values was set to [0.50, 5.50] in order to avoid very sharp peaks and many flat areas, and the range of rotation angles was set to [$-\pi/4$, $\pi/4$] (according to the suggestion in [149]). Ten random problems were generated using the above specified configuration and on each of them each algorithm was run for 20 trials. The parameter settings for the algorithms were P=N=50, F=0.5, CF=0.8 and L=10. Each algorithm was allowed to evolve for maximum 10,000 times.

4.4.3 Results

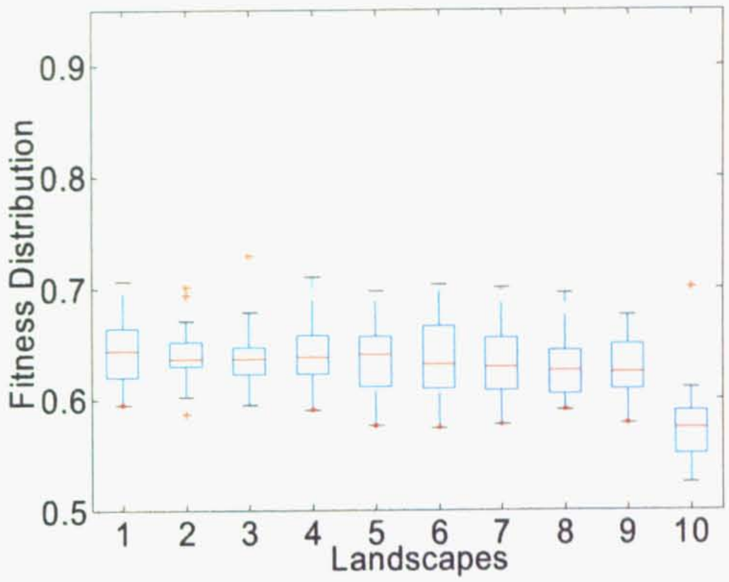
For performance evaluation the fitness value of the best individual found during the run of each algorithm was used. Since different landscapes may have different global optima in terms of fitness value, in order to have a uniform representation for different landscapes, raw fitness values were normalized in the range [0, 1]. For representing different algorithm's performance box-plots were used where each box shows the fitness distribution of an algorithm on a specific landscape over 20 run.

Graphs of Fig. 4.3 show the performance of DE, DEfirDE and DEfirSPX on different landscapes respectively. From these graphs it is clear that after equal number of evaluation the best fitness attained by DEfirDE and DEfirSPX algorithms are significantly better than that attained by DE. In Fig. 4.3(a), 4.3(b) and 4.3(c) fitness distribution of the same landscape takes the same position in the graph. So if these three graphs are compared then it will be found that if some landscape was comparatively difficult for DE (e.g. landscape 10), it was also difficult for DEfirDE and DEfirSPX. This is not unlikely as the proposed FIR schemes work with in the general framework of DE. Still, by searching locally in the neighborhood of the best individual of each generation, the memetic versions of DE was successful to reach a better fitness value compared to that attained by DE. Therefore, from these results it is obvious that because of using FIR schemes the performance of DE has become much better.

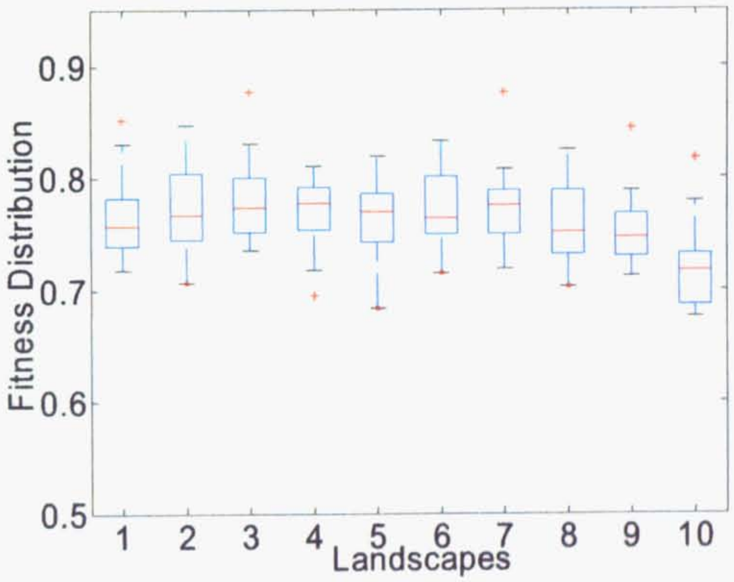
Fig. 4.4 shows the convergence curve of different algorithms for the first landscape. This graph of Fig. 4.4 is sort of representative graph for all other landscapes. As shown in this graph, performance of all algorithms faded with time, but starting with a steeper curve DEfirDE and DEfirSPX attained better fitness compared to DE. The convergence curve of DEfirSPX is especially notable. Because of the superiority of simplex crossover operation, the FIR strategy generated offspring of very high quality resulting in a sharp increase in fitness in the beginning of the search. Though in a substandard way, DEfirDE scheme also showed better convergence velocity than DE. Moreover, the relative performance of DEfirDE and DEfirSPX is similar to what observed in previous results. However, the results of these experiments were helpful to establish the claim about the use of FIR strategy for improving the performance of DE for highly multimodal problems.

4.5 Application to Optimize the S-System Model

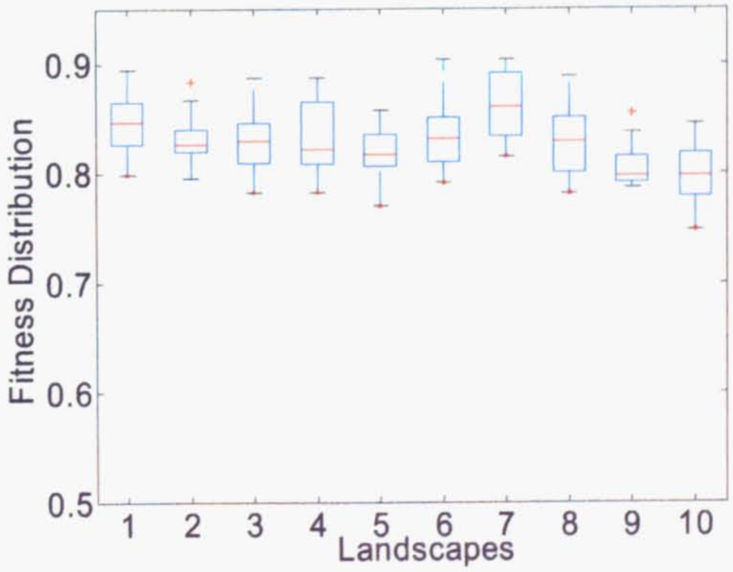
Finally, the optimization capabilities of the proposed variants of DE were tested by applying them in the S-system model parameter optimization. In this problem the best set of parameters were searched for the target S-system model of a genetic network through minimization of the basic MSE oriented fitness function of (3.8).



(a)



(b)



(c)

Figure 4.3: Experimental Results on Landscapes using (a) DE (b) DEfirDE and (c) DEfirSPX

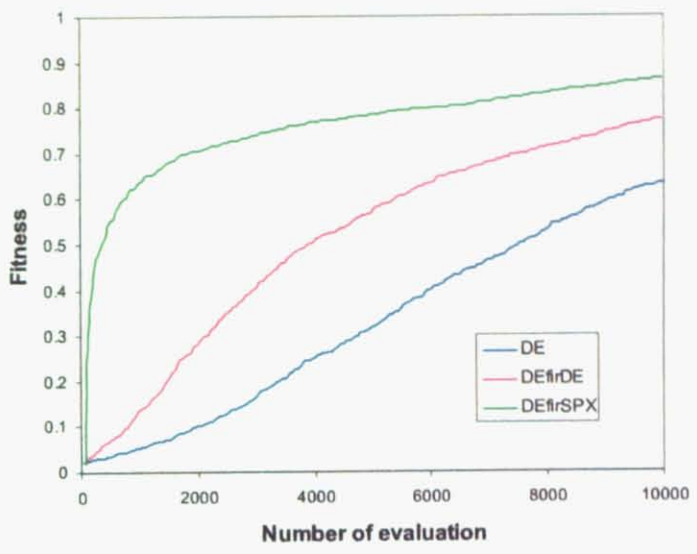


Figure 4.4: Mean Performance of DE, DEfirDE and DEfirSPX on landscape 1

Noman and Iba [86] have proposed an algorithm that finds the parameter values for S-system model based networks with higher accuracy using Differential Evolution (DE). They performed some experiments to evaluate the performance of DE for the deceptive and highly multimodal search space of gene network inference problem. In their experiments they found DE exhibited better performance compared to other conventional EC algorithms like GA or ES. In order to study the performance of DEfirDE and DEfirSPX compared to that of DE for genetic network reconstruction problem they were applied for optimizing the S-system based gene network. A comparison among the optimization performance of the parent-algorithm and the proposed algorithms will give insights about the suitability of the new algorithms for highly nonlinear and multimodal problem of gene regulatory network inference which is the principal theme of this study.

4.5.1 Experimental Setup

In this experiment an artificial gene network representing typical gene regulation process with $N' = 5$ was used. Every details about this particular network can be found in the Chapter 6 where it was studied extensively. This particular network has become a sort of benchmark for evaluating the algorithm performance as it has been widely studied by many other researchers [54, 56, 87, 77, 130]. Artificial time series data was created by integrating the S-system model parameters in [54] from $t_0 = 0$ to t_{max} using fourth order Runge-Kutta algorithm and taking equidistant sample points. The time series data used for optimization are same as used in

[54, 87]. From each time-course of the network, 50 sample points were used for optimization. These artificial microarray data sets were reverse engineered by DE, DEfirDE and DEfirSPX algorithms for estimating the underlying gene network and their performance were compared.

Like in other experiments, the same sets of initial random populations were used for evaluating different algorithms. Each experiment was repeated 20 times in this manner. Maximum number of evaluations allowed for each algorithm was 2,000,000. The parameter setting was as follows: P=60, F = 0.5, CF = 0.8 and L=10.

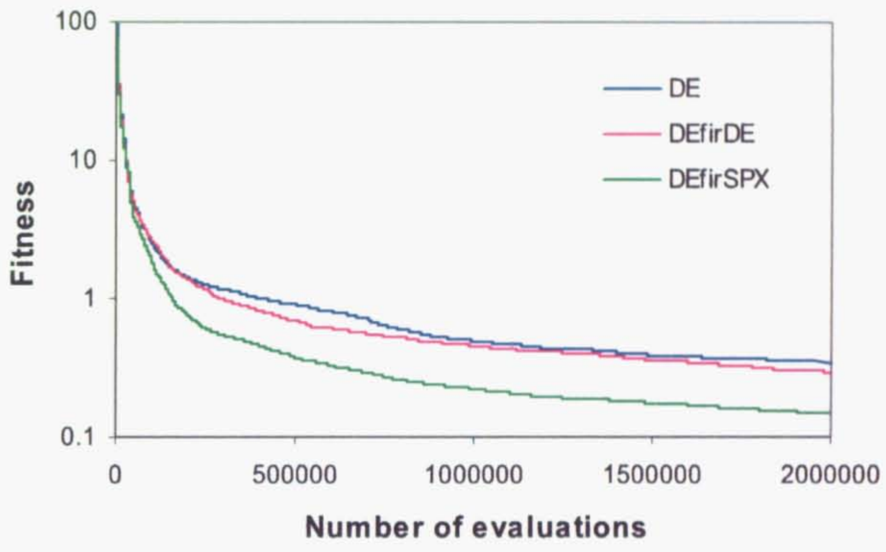


Figure 4.5: Convergence courses for different algorithms for optimizing the MSE fitness function for S-system

4.5.2 Results

Fitness transitions for different algorithms in the experiment are shown in Fig. 4.5. None of the algorithms succeed to reach the optimal after 2000,000 numbers of evaluations. Though not clear from the graph, all the three schemes, initialized with same and equal number of individuals, started with same fitness value. The proposed memetic DEs, augmented with FIR, started with steeper convergence curves and continued to produce better fitness values compared to that by basic DE, in equal number of fitness evaluations. However, based on the convergence curves that do not seem to be converged at the end of optimization, it can be predicted that the proposed memetic versions of DE will reach the optimal value or a targeted minimum fitness value with fewer evaluations than that will be required by original DE. Nevertheless, between DEfirDE and DEfirSPX the later started with steepest convergence curve as usual and managed to find the best average fitness value. The performance of DEfirDE was marginally better than that of DE. These results prove the superiority of DEfirSPX scheme once again. Briefly, Fig. 4.5 suggests that DEFIR algorithms are able to find a network structure as well as parameter values with higher accuracy and velocity compared to that by DE.