# Chapter 5

# An Adaptive local search for DE

Several studies have shown that incorporation of some form of domain knowledge can greatly improve the search capability of Evolutionary Algorithms (EAs). As discussed in previous chapter, among different techniques of domain knowledge incorporation in EAs neighborhood exploration is a promising one. However, often the neighborhood searches are performed in a problem depending manner hampering the superior quality of EAs of being problem-independent. Therefore, adaptation of operators and parameters has become a very promising research field in Memetic Algorithms (MAs). Because of the superior performance of adaptive MAs, this chapter presents an adaptive form of the early presented crossover based local search for Differential Evolution (DE).

The previous chapter has already presented a study on the use of local search operation for improving the performance of DE particularly for high dimensional optimization problems. In this chapter, a more generalized and efficient local search process in the spirit of Lamarckian learning for accelerating classic DE is presented.

The adaptive nature of the newly proposed local search scheme helps to accomplish a more effective neighborhood search and thus significantly improves the convergence characteristics of the original algorithm. The performance improvement was shown using a set of benchmark functions with different properties. Performance comparisons with some well known MAs are also presented.

## 5.1 Differential Evolution with Adaptive XLS

In order to design an effective and efficient MA for global optimization, we need to take advantage of both the exploration abilities of EA and the exploitation abilities of LS by combining them in a well-balanced manner [48]. As mentioned before, in order

to incorporate a crossover based LS (XLS) in an EA, several issues must be resolved, such as the length of the XLS, the selection of individuals which undergo the XLS, the choice of the other parents which participate in the crossover operation, whether deterministic or stochastic application of XLS should be used, etc. Depending on the way the search length is selected, different XLS can be classified into three categories

*Fixed Length XLS* generates a predetermined number of offspring to search the neighborhood of the parent individuals. This type of search has been presented in the previous chapter and also has been used in [145, 69, 86].

*Dynamic Length XLS* varies the length of the local search gradually with the progress of the search, e.g. by applying longer XLS in the beginning, and gradually applying shorter length XLS towards the end of the search.

*Adaptive Length XLS* determines the direction and length of the search by taking some sort of feedback from the search.

In fixed length XLS, identifying a proper length for the LS is most important because too short an XLS may be unsuccessful to explore the neighborhood of the solution, and so to improve the quality of the search. On the other hand, too long an XLS may backfire by consuming additional fitness evaluations unnecessarily. However, finding a single length for XLS that gives optimized results for each problem in each dimension is almost impossible [69]. Similarly, determining a robust adjustment rate is not easy for dynamic length XLS. Therefore, this chapter proposes a Lamarckian local search that adaptively determines the length of the search by taking feedback from the search. This local search strategy was named as Adaptive Hill Climbing XLS (AHCXLS) because it uses a simple hill climbing algorithm to determine the search length adaptively. The pseudo-code of AHCXLS is shown in Fig. 5.1.(a).

Another issue in designing XLS is selecting the individuals that will undergo the local search process. XLS can be applied on every individual or on some deterministically/stochastically selected individuals. In principle, the XLS should be applied only to individuals that will productively take the search towards the global optimum. This is particularly important because application of XLS on an ordinary individual may unnecessarily waste function evaluations and turn out to be expensive. Unfortunately, there is no straightforward way to select the most promising individuals for XLS. In EC, the solutions with better fitness values are generally preferred for reproduction, as they are more likely to be in the proximity of a basin of attraction. Therefore, the best individual of the population was deterministically

**AHCXLS**($I$, $n_p$)

1. P[1] = $I$
2. Repeat i=2 to $n_p$ times
3. P[i] = select random individuals from the population
4. End Repeat
5. C = ***Crossover*** (P)
6. If C is better than P[1]
   P[1] = C
7. Else
   ***Return*** (P[1])
8. Go to step 5

(a) AHCXLS local search

**DEahcSPX**

1. Generate an Initial Population $P^G$
2. ***Evaluate*** $P^G$
3. B = ***BestIndex***($P^G$)
4. $P^G$.[B] = ***AHCXLS*** ($P^G$.[B], $n_p$)
5. For each individual $I$ in $P^G$
6. ***Reproduce*** an offspring $J$ from $I$
7. $P^{G+1} = P^{G+1} \cup$ ***Select*** (I,J)
8. Set G = G+1
9. Repeat Step 3 to 8 until termination criteria is met

(b) DEahcSPX algorithm

Figure 5.1: Proposed DEahcSPX algorithm and the adaptive local search scheme AHCXLS. $I$ is the individual on which the AHCXLS is applied and $n_p$ is the total number of individuals that take part in the crossover operation. *BestIndex* return the index of the best individual of the current generation. Other symbols represent standard notations.

selected for exploring its neighborhood using the XLS and thereby it is expected to end with a nearby better solution. The other individuals that participate in the crossover operation of XLS are chosen randomly to keep the implementation simple and to promote population diversity. And the final decision is about the crossover operation could be used in the XLS scheme. Tsutsui *et al.* have proposed simplex crossover (SPX) for real-coded GAs [131]. The SPX operator uses $n_p$ parental vectors for recombination and offers various advantages as mentioned before. Besides, it was shown in previous chapter that SPX was a promising operation for local tuning, and therefore SPX was used as the fundamental crossover operation in this study for comparison purpose. More details about the SPX crossover can be found in [131]. The new version of DE with the AHCXLS and SPX operation is titled as DEahcSPX and is described in Fig. 5.1.(b).

The primary difference between the newly proposed DEahcSPX algorithm and previously proposed DEfirSPX algorithm is that we are no more required to look for a good search length for the XLS operation. The simple rule of hill-climbing adaptively determines the best length by taking feedback from the search. Hence, using the best length (according to the heuristics) for the local search adaptively, the new algorithm makes best use of the function evaluations and thereby identifies the optimum at a higher velocity compared to the earlier proposal. Furthermore, the earlier DEfirSPX is only suitable for high-dimensional optimization problems

because of its fixed-length XLS strategy that consumes a fixed number of function evaluation in each call. On the other hand, because of the adaptive XLS-length adjustment capability of AHCXLS, the newly proposed DEahcSPX algorithm is applicable to optimization problems of any dimension. Finally, because of the simple hill-climbing mechanism, the new adaptive local-search does not add any additional complexity or any additional parameter to the original algorithm.

## 5.2 Experiments

Different experiments have been carried out to assess the performance of DEahcSPX using the test suite described in Appendix C. The focus of the study was to compare the performance of the proposed DEahcSPX algorithm with the original DE algorithm in different experiments. Also the performance of DEahcSPX was studied comparing with other EAs, and the efficiency of AHCXLS comparing with other XLS strategies. Here, DE was used to denote the *'DE/rand/1/bin'* variant (if not otherwise specified) of the algorithm and the DEahcSPX algorithm was implemented by embedding the AHCXLS strategy in the same variant of DE.

### 5.2.1 Performance Evaluation Criteria

For evaluating the performance of the algorithms a criteria similar to that defined in [125] was used. The performance of DEahcSPX was compared with DE for the test suite using the *function error value*. The *function error value* for a solution $x$ is defined as $(f(x) - f(x^*))$ where $x^*$ is the global optimum of the function. The maximum number of fitness evaluations allowed for each algorithm to minimize this error was $10,000 \times N$, where $N$ is the dimension of the problem. 50 trials were repeated on each function. The fitness evaluation criteria were as follows

1. **Error:** The minimum function error value that an algorithm can find, using $10,000 \times N$ fitness evaluations at maximum, was recorded in each run and the average and standard deviation of the error values were calculated. The number of trials in which the algorithms could reach the accuracy level $\varepsilon$ (explained in next paragraph) using maximum $10,000 \times N$ fitness evaluations was counted. For this criterion the notation $AVG_{Er} \pm SD_{Er}(CNT)$ was used in different tables.

2. **Evaluation:** The number of function evaluations (FEs) required to reach an error value less than $\varepsilon$ (provided that the maximum limit is $10,000 \times N$ FEs) was also recorded in different runs and the average and standard deviation of the number of evaluations were calculated. For the functions $F_1$ to $F_5$ the accuracy level $\varepsilon$ was

fixed at $10^{-6}$ and for the functions $F_6$ to $F_{10}$ $\varepsilon$ was fixed at $10^{-2}$ as in [125]. The accuracy level $\varepsilon$ was fixed for the rest of the functions at $10^{-6}$. For this criterion the notation $AVG_{Ev} \pm SD_{Ev}(CNT)$ was used where CNT is the number of runs in which the algorithms could reach this accuracy level $\varepsilon$ using $10,000 \times N$ FEs at maximum.

3. **Convergence Graphs:** Convergence graphs of the algorithms for $N = 30$. These graphs show the average **Error** performance of the total runs, in respective experiments.

### 5.2.2 Experimental Setup

In these experiments, the same set of initial random populations were used to evaluate different algorithms. Though classic DE uses only three control parameters, namely *Population Size* $P$, *Scaling Factor* $F$ and *Crossover Rate* $C_r$, choice of these parameters is critical for its performance [68, 32]. $F$ is generally related to the convergence speed. To avoid premature convergence it is crucial for $F$ to be of sufficient magnitude [101]. $F = 0.9$ is suggested as a good compromise between convergence-speed and convergence-probability in [106]. 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 [106]. Therefore, $F = 0.9$ and $C_r = 0.9$ were chosen for all the functions in every experiments without tuning them to their optimal values for different problems. These parameter settings are also studied elsewhere [106, 68]. Population size is a critical choice for the performance of DE. Here, the performance of the DE and DEahcSPX was investigate with population size $P = N$. The effect of population size was also studied . For the proposed DEahcSPX no additional parameter setting is required. For the SPX operation the number of parents participating in the crossover operation was chosen to be $n_p = 3$ as suggested in [131].

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.

## 5.3 Effect of AHCXLS on DE

The results of this section are intended to show how the proposed AHCXLS strategy can improve the performance of DE. In order to show the superiority of the newly proposed DEahcSPX, it was compared with DE carrying out experiments on the test suite at dimension N=30 and the results are presented in Table 5.1 and Table

5.2. The functions for which no convergence was achieved were removed from Table 5.2. All the settings are the same as mentioned in Section 5.2.2. Some representative graphs comparing the convergence characteristics of DE with DEahcSPX are shown in Fig. 5.2.

Table 5.1: Best **Error** values at N=30, after 300,000 fitness evaluation

| | DE | DEahcSPX | | DE | DEahcSPX |
|---|---|---|---|---|---|
| $F_{sph}$ | 5.73E-17 ± 2.03E-16 | **1.75E-31 ± 4.99E-31** | $F_1$ | 3.87E-14 ± 2.71E-14 | **0.00E+00 ± 0.00E+00** |
| $F_{ros}$ | 5.20E+01 ± 8.56E+01 | **4.52E+00 ± 1.55E+01** | $F_2$ | 8.50E-02 ± 7.94E-02 | **6.52E-05 ± 4.84E-05** |
| $F_{ack}$ | 1.37E-09 ± 1.32E-09 | **2.66E-15 ± 0.00E+00** | $F_3$ | 3.63E+06 ± 2.06E+06 | **1.29E+06 ± 9.22E+05** |
| $F_{grw}$ | 2.66E-03 ± 5.73E-03 | **2.07E-03 ± 5.89E-03** | $F_4$ | 5.54E+01 ± 6.37E+01 | **4.62E+00 ± 8.78E+00** |
| $F_{ras}$ | 2.55E+01 ± 8.14E+00 | 2.14E+01 ± 1.23E+01 | $F_5$ | 1.08E+03 ± 5.31E+02 | **9.00E+02 ± 4.79E+02** |
| $F_{sch}$ | 4.90E+02 ± 2.34E+02 | 4.70E+02 ± 2.96E+02 | $F_6$ | 6.67E+01 ± 1.51E+02 | **3.84E+00 ± 3.75E+00** |
| $F_{sal}$ | 2.52E-01 ± 4.78E-02 | 1.80E-01 ± 4.08E-02 | $F_7$ | 7.59E-03 ± 8.96E-03 | **7.39E-03 ± 6.32E-03** |
| $F_{wht}$ | 3.10E+02 ± 1.07E+02 | 3.06E+02 ± 1.10E+02 | $F_8$ | 2.09E+01 ± 1.33E-01 | 2.09E+01 ± 1.12E-01 |
| $F_{pn1}$ | 4.56E-02 ± 1.31E-01 | **2.07E-02 ± 8.46E-02** | $F_9$ | 2.43E+01 ± 6.23E+00 | 2.04E+01 ± 8.19E+00 |
| $F_{pn2}$ | 1.44E-01 ± 7.19E-01 | **1.71E-31 ± 5.35E-31** | $F_{10}$ | 7.33E+01 ± 6.62E+01 | **5.27E+01 ± 4.84E+01** |

Table 5.2: **FEs** required to achieve accuracy levels less than $\varepsilon$ (N=30)

| | DE | DEahcSPX |
|---|---|---|
| $F_{sph}$ | 148650.8 ± 6977.7 (50) | **87027.4 ± 3967.3 (50)** † |
| $F_{ros}$ | - | **299913.0 ± 519.5 (2)** |
| $F_{ack}$ | 215456.1 ± 9721.4 (50) | **129211.6 ± 5168.6 (50)** † |
| $F_{grw}$ | 190292.5 ± 63478.8 (38) | **121579.2 ± 79563.4 (43)** † |
| $F_{pn1}$ | 160955.2 ± 63176.3 (43) | **96149.0 ± 61787.7 (46)** † |
| $F_{pn2}$ | 156016.9 ± 31515.8 (48) | **85360.2 ± 6390.6 (50)** † |
| $F_1$ | 153450.1 ± 5780.4 (50) | **89417.8 ± 4117.6 (50)** † |
| $F_2$ | - | **299279.4 ± 3685.9 (3)** |
| $F_7$ | 211778.8 ± 70080.3 (33) | **148067.7 ± 68996.3 (42)** † |

† The $t$ value is significant at a $1^{-04}$ level of significance by two-tailed t-test

Depending on the relative performance of DEahcSPX and DE we can divide the functions into three classes. The *first class* contains the functions ($F_{sph}$, $F_{ack}$, $F_{grw}$, $F_{pn1}$, $F_{pn2}$, $F_1$ and $F_7$ ) for which DEahcSPX reached the target accuracy level using fewer fitness evaluation, or achieved that in an equal or higher number of trials compared to DE (Table 5.2). The *second class* consists of the functions in which none of the algorithms achieved the desired accuracy level but the newly proposed one reached at a smaller error value. This class contains the functions $F_{ros}$, $F_2$, $F_3$, $F_4$, $F_5$, $F_6$ and $F_{10}$ (Table 5.1). The *third class* contains the functions in which no significant difference was observed in the achieved error values attained by the algorithms. This class consists of the functions $F_{ras}$, $F_{sch}$, $F_{sal}$, $F_{wht}$ and $F_9$. Although no significant difference was noticed in the error values, it was revealed by the convergence curves that these error values were achieved using fewer fitness evaluation in DEahcSPX algorithm compared to DE (Fig. 5.2). Only in the case of $F_8$ no significance difference was observed in the algorithms performance. It seems

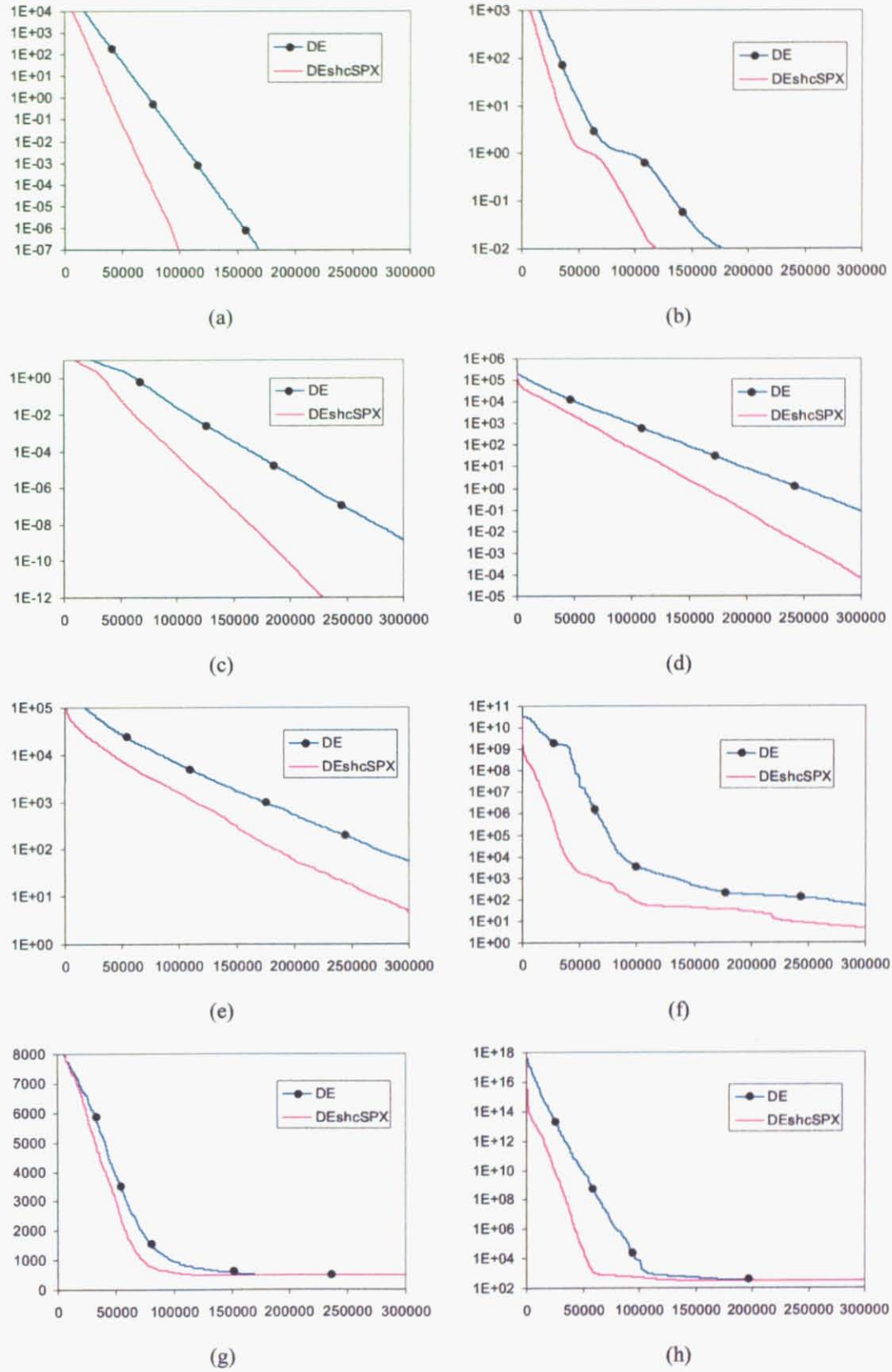


Figure 5.2: Convergence curves of DE and DEahcSPX algorithm for selected functions (N=30). X-axis represents fitness evaluations (FEs) and Y-axis represents **Error** values. (a) $F_1$, (b) $F_7$, (c) $F_{ack}$, (d) $F_2$, (e) $F_4$, (f) $F_{ros}$, (g) $F_{sch}$ and (h) $F_{wht}$

that the learning strategy of Eq. (4.1) together with the binomial crossover operation used in DE was not good enough to locate the global optimum for the functions belonging to the second class and the third class. Since the DEahcSPX algorithm depends mostly on the working principle of DE, it is natural that it also could not locate the global optimal using the same learning strategy. However, hybridization of DE with the AHCXLS scheme notably speeds up the original algorithm. In general, the overall results of Table 5.1 and Table 5.2 and the graphs of Fig. 5.2 substantiate the claim that the proposed AHCXLS strategy accelerates the classic DE algorithm.

Table 5.3: Best **Error** values for varying PopSize at N=30, after 300,000 FEs

| PopSize=50 | | | PopSize=100 | | |
|---|---|---|---|---|---|
| | DE | DEahcSPX | | DE | DEahcSPX |
| $F_{sph}$ | 2.31E-02 ± 1.92E-02 | **6.03E-09 ± 6.86E-09 (50)** | $F_{sph}$ | 3.75E+03 ± 1.14E+03 | **3.11E+01 ± 1.88E+01** |
| $F_{ros}$ | 3.70E+02 ± 4.81E+02 | **4.98E+01 ± 6.22E+01** | $F_{ros}$ | 4.03E+08 ± 2.59E+08 | **1.89E+05 ± 1.47E+05** |
| $F_{ack}$ | 3.60E-02 ± 1.82E-02 | **1.89E-05 ± 1.19E-05** | $F_{ack}$ | 1.36E+01 ± 1.48E+00 | **3.23E+00 ± 5.41E-01** |
| $F_{grw}$ | 5.00E-02 ± 6.40E-02 | **1.68E-03 ± 4.25E-03 (42)** | $F_{grw}$ | 3.57E+01 ± 1.26E+01 | **1.29E+00 ± 1.74E-01** |
| $F_{ras}$ | 5.91E+01 ± 2.65E+01 | **2.77E+01 ± 1.31E+01** | $F_{ras}$ | 2.63E+02 ± 2.79E+01 | 1.64E+02 ± 2.16E+01 |
| $F_{sch}$ | 7.68E+02 ± 8.94E+02 | **2.51E+02 ± 1.79E+02** | $F_{sch}$ | 6.56E+03 ± 4.25E+02 | 6.30E+03 ± 4.80E+02 |
| $F_{sal}$ | 8.72E-01 ± 1.59E-01 | **2.44E-01 ± 5.06E-02** | $F_{sal}$ | 5.97E+00 ± 6.54E-01 | **1.20E+00 ± 2.12E-01** |
| $F_{wht}$ | 8.65E+02 ± 1.96E+02 | **4.58E+02 ± 7.56E+01** | $F_{wht}$ | 1.29E+14 ± 1.60E+14 | **3.16E+08 ± 4.48E+08** |
| $F_{pn1}$ | 2.95E-04 ± 1.82E-04 | **1.12E-09 ± 2.98E-09 (50)** | $F_{pn1}$ | 6.94E+04 ± 1.58E+05 | **2.62E+00 ± 1.31E+00** |
| $F_{pn2}$ | 9.03E-03 ± 2.03E-02 | **4.39E-04 ± 2.20E-03 (47)** | $F_{pn2}$ | 6.60E+05 ± 7.66E+05 | **4.85E+00 ± 1.59E+00** |
| $F_1$ | 1.69E-02 ± 1.80E-02 | **1.67E-08 ± 2.19E-08 (50)** | $F_1$ | 5.68E+03 ± 2.63E+03 | **4.31E+01 ± 2.16E+01** |
| $F_2$ | 8.38E+02 ± 7.20E+02 | **1.55E+01 ± 1.09E+01** | $F_2$ | 5.79E+04 ± 1.53E+04 | **4.34E+03 ± 1.57E+03** |
| $F_3$ | 5.86E+07 ± 2.61E+07 | **4.75E+06 ± 1.82E+06** | $F_3$ | 8.82E+08 ± 2.61E+08 | **1.97E+07 ± 4.84E+06** |
| $F_4$ | 3.65E+03 ± 2.03E+03 | **2.31E+02 ± 1.42E+02** | $F_4$ | 9.45E+04 ± 2.77E+04 | **9.55E+03 ± 3.93E+03** |
| $F_5$ | 3.20E+03 ± 1.31E+03 | **1.04E+03 ± 3.67E+02** | $F_5$ | 2.33E+04 ± 4.03E+03 | **5.88E+03 ± 1.24E+03** |
| $F_6$ | 5.64E+02 ± 7.58E+02 | **7.00E+01 ± 1.28E+02** | $F_6$ | 7.27E+08 ± 5.08E+08 | **4.05E+05 ± 3.01E+05** |
| $F_7$ | 9.54E-01 ± 9.75E-02 | **3.19E-03 ± 5.14E-03 (44)** | $F_7$ | 5.73E+02 ± 1.85E+02 | **1.18E+01 ± 5.78E+00** |
| $F_8$ | 2.09E+01 ± 5.94E-02 | 2.09E+01 ± 5.25E-02 | $F_8$ | 2.09E+01 ± 3.84E-02 | 2.09E+01 ± 5.89E-02 |
| $F_9$ | 5.23E+01 ± 2.36E+01 | **2.56E+01 ± 1.48E+01** | $F_9$ | 2.73E+02 ± 1.53E+01 | **1.83E+02 ± 2.25E+01** |
| $F_{10}$ | 2.24E+02 ± 1.85E+01 | 1.55E+02 ± 4.53E+01 | $F_{10}$ | 3.31E+02 ± 3.53E+01 | **2.05E+02 ± 1.55E+01** |

| PopSize=200 | | | PopSize=300 | | |
|---|---|---|---|---|---|
| | DE | DEahcSPX | | DE | DEahcSPX |
| $F_{sph}$ | 4.01E+04 ± 6.26E+03 | **1.10E+03 ± 2.98E+02** | $F_{sph}$ | 1.96E+04 ± 2.00E+03 | **6.93E+02 ± 1.34E+02** |
| $F_{ros}$ | 1.53E+10 ± 4.32E+09 | **1.49E+07 ± 7.82E+06** | $F_{ros}$ | 3.97E+09 ± 8.92E+08 | **5.35E+06 ± 2.82E+06** |
| $F_{ack}$ | 2.02E+01 ± 2.20E-01 | **9.11E+00 ± 7.81E-01** | $F_{ack}$ | 1.79E+01 ± 3.51E-01 | **7.23E+00 ± 4.50E-01** |
| $F_{grw}$ | 3.73E+02 ± 6.03E+01 | **1.08E+01 ± 2.02E+00** | $F_{grw}$ | 1.79E+02 ± 1.60E+01 | **7.26E+00 ± 1.74E+00** |
| $F_{ras}$ | 3.62E+02 ± 2.12E+01 | **2.05E+02 ± 1.85E+01** | $F_{ras}$ | 2.75E+02 ± 1.27E+01 | 2.03E+02 ± 1.49E+01 |
| $F_{sch}$ | 6.88E+03 ± 2.55E+02 | 6.72E+03 ± 3.24E+02 | $F_{sch}$ | 6.87E+03 ± 2.72E+02 | 6.80E+03 ± 3.37E+02 |
| $F_{sal}$ | 1.34E+01 ± 8.41E-01 | **3.25E+00 ± 4.55E-01** | $F_{sal}$ | 1.52E+01 ± 5.43E-01 | **3.59E+00 ± 4.54E-01** |
| $F_{wht}$ | 2.29E+16 ± 1.16E+16 | **5.47E+10 ± 6.17E+10** | $F_{wht}$ | 2.96E+16 ± 1.09E+16 | **1.83E+11 ± 1.72E+11** |
| $F_{pn1}$ | 2.44E+07 ± 7.58E+06 | **9.10E+00 ± 2.42E+00** | $F_{pn1}$ | 3.71E+07 ± 1.29E+07 | **1.09E+01 ± 3.76E+00** |
| $F_{pn2}$ | 8.19E+07 ± 1.99E+07 | **6.18E+01 ± 6.30E+01** | $F_{pn2}$ | 1.03E+08 ± 1.87E+07 | **3.42E+02 ± 4.11E+02** |
| $F_1$ | 5.51E+04 ± 6.74E+03 | **2.04E+03 ± 5.09E+02** | $F_1$ | 5.18E+03 ± 7.23E+02 | **4.37E+02 ± 8.16E+01** |
| $F_2$ | 1.16E+05 ± 1.60E+04 | **1.09E+04 ± 3.00E+03** | $F_2$ | 2.88E+04 ± 3.54E+03 | **7.08E+03 ± 1.22E+03** |
| $F_3$ | 1.19E+09 ± 1.63E+08 | **4.02E+07 ± 1.48E+07** | $F_3$ | 1.56E+08 ± 3.07E+07 | **2.69E+07 ± 6.84E+06** |
| $F_4$ | 1.43E+05 ± 2.63E+04 | **1.68E+04 ± 3.80E+03** | $F_4$ | 3.49E+04 ± 4.96E+03 | **1.10E+04 ± 1.93E+03** |
| $F_5$ | 3.29E+04 ± 2.71E+03 | **9.12E+03 ± 1.63E+03** | $F_5$ | 1.10E+04 ± 5.32E+02 | **7.64E+03 ± 4.72E+02** |
| $F_6$ | 2.61E+10 ± 9.11E+09 | **4.64E+07 ± 1.65E+07** | $F_6$ | 1.86E+08 ± 4.07E+07 | **1.48E+06 ± 5.45E+05** |
| $F_7$ | 3.46E+03 ± 4.31E+02 | **1.50E+02 ± 2.82E+01** | $F_7$ | 4.01E+03 ± 5.13E+02 | **2.57E+02 ± 5.97E+01** |
| $F_8$ | 2.09E+01 ± 6.07E-02 | 2.09E+01 ± 5.99E-02 | $F_8$ | 2.10E+01 ± 4.44E-02 | 2.09E+01 ± 5.22E-02 |
| $F_9$ | 4.13E+02 ± 2.46E+01 | **2.31E+02 ± 2.10E+01** | $F_9$ | 2.22E+02 ± 1.03E+01 | 2.05E+02 ± 1.35E+01 |
| $F_{10}$ | 6.00E+02 ± 5.28E+01 | **2.65E+02 ± 1.61E+01** | $F_{10}$ | 2.75E+02 ± 1.30E+01 | 2.13E+02 ± 1.21E+01 |

### 5.3.1 Sensitivities to Population Size

Performance of DE is always sensitive to the selected population size [32, 86]. This is easily conceivable because DE employs a one-to-one reproduction strategy. Therefore, if a very large population size is selected then DE exhausts the fitness evaluations very quickly without being able to locate the optimum. Storn and Price

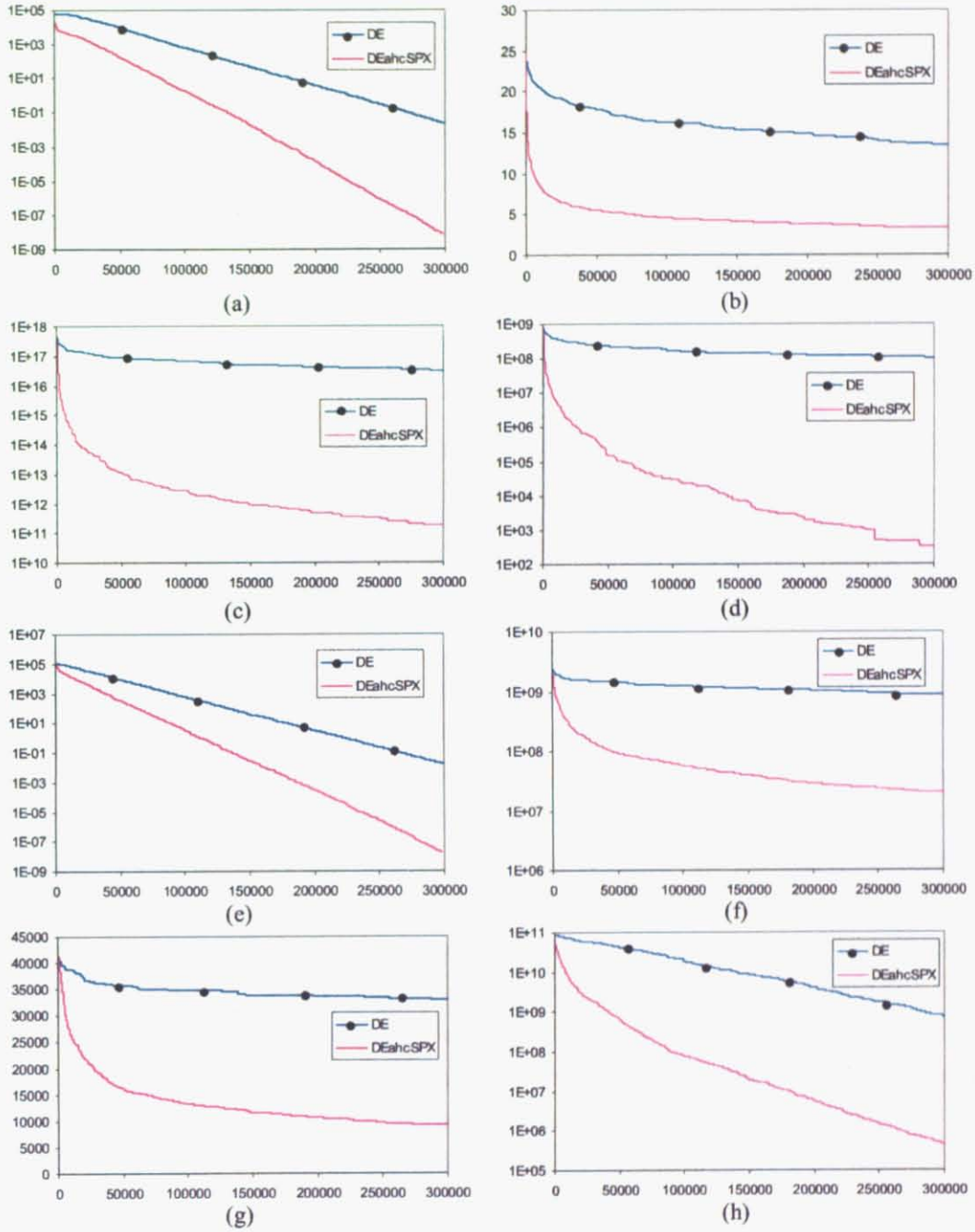


Figure 5.3: Convergence curves to show the sensitivities of DE and DEahcSPX to population-size for selected functions (N=30). X-axis represents fitness evaluations (FEs) and Y-axis represents **Error** values. (a) $F_{sph}$(P=50), (b) $F_{sal}$(P=200), (c) $F_{wht}$(P=300), (d) $F_{pn2}$(P=300), (e) $F_1$(P=50), (f) $F_3$(P=100), (g) $F_5$(P=200) and (h) $F_6$(P=100)

Table 5.4: Scalability study in terms of **Error** values

| | N=10 | | | N=50 | |
|---|---|---|---|---|---|
| | DE | DEahcSPX | | DE | DEahcSPX |
| $F_{sph}$ | 3.26E-28 ± 5.83E-28 (50) | **1.81E-38 ± 4.94E-38 (50)** | $F_{sph}$ | 5.91E-02 ± 9.75E-02 | **8.80E-09 ± 2.80E-08 (50)** |
| $F_{ros}$ | 4.78E-01 ± 1.32E+00 (43) | **3.19E-01 ± 1.10E+00 (46)** | $F_{ros}$ | 1.13E+10 ± 2.34E+10 | **1.63E+02 ± 3.02E+02** |
| $F_{ack}$ | 8.35E-15 ± 8.52E-15 (50) | **2.66E-15 ± 0.00E+00 (50)** | $F_{ack}$ | 2.39E-02 ± 8.90E-03 | **1.69E-05 ± 8.86E-06** |
| $F_{grw}$ | 5.75E-02 ± 3.35E-02 | **4.77E-02 ± 2.55E-02** | $F_{grw}$ | 7.55E-02 ± 1.14E-01 | **2.96E-03 ± 5.64E-03 (36)** |
| $F_{ras}$ | 1.85E+00 ± 1.68E+00 (13) | **1.60E+00 ± 1.61E+00 (18)** | $F_{ras}$ | 6.68E+01 ± 2.36E+01 | **3.47E+01 ± 9.23E+00** |
| $F_{sch}$ | 14.21272743 ± 39.28155167 | **4.73766066 ± 23.68766692** | $F_{sch}$ | 1.07E+03 ± 5.15E+02 | **9.56E+02 ± 2.88E+02** |
| $F_{sal}$ | 0.107873375 ± 0.027688791 | **0.099873361 ± 3.47E-08** | $F_{sal}$ | 1.15E+00 ± 1.49E-01 | **4.00E-01 ± 1.00E-01** |
| $F_{wht}$ | 18.11229734 ± 15.85783313 | 18.00697444 ± 13.11270338 | $F_{wht}$ | 1.43E+05 ± 4.10E+05 | **1.41E+03 ± 2.90E+02** |
| $F_{pn1}$ | 3.85E-29 ± 7.28E-29 (50) | **4.71E-32 ± 1.12E-47 (50)** | $F_{pn1}$ | 3.07E-02 ± 7.93E-02 | **2.49E-03 ± 1.24E-02 (48)** |
| $F_{pn2}$ | 1.49E-28 ± 2.20E-28 (50) | **1.35E-32 ± 5.59E-48 (50)** | $F_{pn2}$ | 2.24E-01 ± 3.35E-01 | **2.64E-03 ± 4.79E-03 (38)** |
| $F_1$ | 0.00E+00 ± 0.00E+00 (50) | **0.00E+00 ± 0.00E+00 (50)** | $F_1$ | 1.50E-02 ± 1.09E-02 | **1.06E-08 ± 1.22E-08 (50)** |
| $F_2$ | 2.27E-15 ± 1.14E-14 (50) | **0.00E+00 ± 0.00E+00 (50)** | $F_2$ | 2.89E+04 ± 1.03E+04 | **1.44E+03 ± 5.95E+02** |
| $F_3$ | 8.76E-06 ± 2.78E-05 (38) | **2.42E-06 ± 7.11E-06 (40)** | $F_3$ | 5.40E+08 ± 2.62E+08 | **2.27E+07 ± 8.01E+06** |
| $F_4$ | 8.87E-14 ± 1.24E-13 (50) | **0.00E+00 ± 0.00E+00 (50)** | $F_4$ | 6.04E+04 ± 1.74E+04 | **1.04E+04 ± 3.90E+03** |
| $F_5$ | 1.07E-03 ± 2.40E-03 | **1.12E-05 ± 1.75E-05 (12)** | $F_5$ | 5.81E+03 ± 1.12E+03 | **3.71E+03 ± 6.57E+02** |
| $F_6$ | 3.19E-01 ± 1.10E+00 (46) | 3.19E-01 ± 1.10E+00 (46) | $F_6$ | 1.29E+03 ± 1.98E+03 | **2.24E+02 ± 3.99E+02** |
| $F_7$ | 1.56E-01 ± 1.63E-01 | **1.47E-01 ± 1.16E-01 (3)** | $F_7$ | 1.05E+00 ± 6.24E-02 | **2.11E-02 ± 2.29E-02 (18)** |
| $F_8$ | 2.04E+01 ± 1.08E-01 | 2.04E+01 ± 1.45E-01 | $F_8$ | 2.11E+01 ± 2.93E-02 | 2.11E+01 ± 3.63E-02 |
| $F_9$ | 2.01E+00 ± 1.41E+00 (5) | **1.23E+00 ± 9.65E-01 (10)** | $F_9$ | 7.65E+01 ± 2.30E+01 | **5.23E+01 ± 1.53E+01** |
| $F_{10}$ | 1.26E+01 ± 7.26E+00 | 1.06E+01 ± 4.06E+00 | $F_{10}$ | 4.24E+02 ± 2.98E+01 | 3.35E+02 ± 2.80E+01 |

| | N=100 | | | N=200 | |
|---|---|---|---|---|---|
| | DE | DEahcSPX | | DE | DEahcSPX |
| $F_{sph}$ | 4.28E+03 ± 1.27E+03 | **5.01E+01 ± 8.94E+01** | $F_{sph}$ | 1.26E+05 ± 1.06E+04 | **7.01E+03 ± 1.07E+03** |
| $F_{ros}$ | 3.33E+08 ± 1.67E+08 | **1.45E+05 ± 1.11E+05** | $F_{ros}$ | 2.97E+10 ± 3.81E+09 | **1.11E+08 ± 2.63E+07** |
| $F_{ack}$ | 8.81E+00 ± 8.07E-01 | **1.91E+00 ± 3.44E-01** | $F_{ack}$ | 1.81E+01 ± 2.26E-01 | **8.45E+00 ± 4.13E-01** |
| $F_{grw}$ | 3.94E+01 ± 8.01E+00 | **1.23E+00 ± 2.14E-01** | $F_{grw}$ | 1.15E+03 ± 9.22E+01 | **6.08E+01 ± 9.30E+00** |
| $F_{ras}$ | 8.30E+02 ± 6.51E+01 | **4.75E+02 ± 6.55E+01** | $F_{ras}$ | 2.37E+03 ± 7.24E+01 | 1.53E+03 ± 8.31E+01 |
| $F_{sch}$ | 2.54E+04 ± 2.15E+03 | 2.48E+04 ± 2.17E+03 | $F_{sch}$ | 6.66E+04 ± 1.32E+03 | 6.61E+04 ± 1.44E+03 |
| $F_{sal}$ | 1.02E+01 ± 7.91E-01 | **3.11E+00 ± 5.79E-01** | $F_{sal}$ | 3.69E+01 ± 1.80E+00 | **1.10E+01 ± 4.38E-01** |
| $F_{wht}$ | 5.44E+15 ± 5.07E+15 | **4.06E+10 ± 6.57E+10** | $F_{wht}$ | 3.13E+18 ± 9.48E+17 | **4.21E+13 ± 1.74E+13** |
| $F_{pn1}$ | 6.20E+05 ± 7.38E+05 | **4.34E+00 ± 1.75E+00** | $F_{pn1}$ | 3.49E+08 ± 7.60E+07 | **2.27E+01 ± 5.73E+00** |
| $F_{pn2}$ | 4.34E+06 ± 2.30E+06 | **7.25E+01 ± 2.44E+01** | $F_{pn2}$ | 8.08E+08 ± 1.86E+08 | **6.24E+04 ± 4.77E+04** |

suggested a larger population size (between 5N to 10N) for DE [123], although later studies found that DE performs better with a smaller population [106, 86]. To investigate the sensitivity of the proposed algorithm to variations of population size, experiments were performed with different population sizes at dimension $N = 30$. Results are reported in Table 5.3. A quick look at Table 5.3 reveals how drastically the performance of DE changes with the population size for a given maximum number of evaluations. For some functions, DE converged for all trials using a smaller population size (e.g. P=N) but failed to reach even a single convergence with a larger population (e.g. P = 10N). Since DEahcSPX is just an improvement of basic DE using AHCXLS, it is expected that its sensitivity to variation in population-size is more or less similar to that of the basic algorithm. However, Table 5.3 show that in all experiments the error values achieved by DEahcSPX were always better than those achieved by DE. The graphs of Fig. 5.3 show that AHCXLS scheme has improved the convergence characteristics of the original algorithm, regardless to population-size. Though for some functions ($F_{ras}$, $F_{sch}$, $F_8$, $F_9$, $F10$) the performance of both algorithms were more or less indifferent to population-size, possibly it was because of the incompetence of the learning strategy used. Nevertheless, the results presented in this section confirm that the proposed DEahcSPX algorithm exhibits a higher convergence velocity and greater robustness to the population size

compared to DE.

Table 5.5: **FEs** required to achieve accuracy levels less than $\varepsilon$ (N=10)

| | DE | DEahcSPX | | DE | DEahcSPX |
|---|---|---|---|---|---|
| $F_{sph}$ | 31639.7 ± 1347.0 (50) | **22926.4 ± 1300.3 (50)** | $F_2$ | 49683.4 ± 2184.3 (50) | **34677.8 ± 2203.9 (50)** |
| $F_{ros}$ | 73803.8 ± 12550.9 (43) | **59275.7 ± 14998.0 (46)** | $F_3$ | 94850.3 ± 4906.4 (38) | **89217.0 ± 9466.3 (40)** |
| $F_{ack}$ | 48898.2 ± 1977.7 (50) | **36389.3 ± 1764.4 (50)** | $F_4$ | 58143.8 ± 3372.4 (50) | **39192.2 ± 2673.2 (50)** |
| $F_{ras}$ | 94089.0 ± 12818.3 (13) | **84309.0 ± 22045.5 (18)** | $F_5$ | - | **99328.9 ± 1606.6 (12)** |
| $F_{pn1}$ | 28885.8 ± 2394.4 (50) | **20543.5 ± 1162.8 (50)** | $F_6$ | 61808.5 ± 18899.9 (46) | **50167.7 ± 19785.8 (46)** |
| $F_{pn2}$ | 30812.6 ± 1684.9 (50) | **21633.5 ± 1293.9 (50)** | $F_7$ | - | **97258.7 ± 13794.1 (3)** |
| $F_1$ | 32165.7 ± 1415.3 (50) | **22594.7 ± 1255.7 (50)** | $F_9$ | 97860.2 ± 7475.7 (5) | **89685.7 ± 21519.0 (10)** |

### 5.3.2 Scalability Study

So far, the experiments of these chapter were done in $N = 30$ dimensional problem space. In order to study the effect of problem-dimension on the performance of the DEahcSPX algorithm, a scalability study was carried out comparing with the original DE algorithm. Since the functions $F_1$ to $F_{10}$ are defined up to N=50 dimensions, they were studied at N=10 and 50 dimensions. The other functions were studied at N=10, 50, 100 and 200 dimensions. For N=10 dimensions, population-size was chosen as P=30 and for all other dimensions it was selected as P=N. The accuracy achieved using $N \times 10,000$ fitness evaluations are presented in Table 5.4, and some representative convergence graphs are shown in Fig. 5.4. In order to focus on the comparison between the proposed algorithm DEahcSPX and its parent algorithm DE, Table 5.5 compares the fitness evaluations required by the algorithms to achieve the accuracy level $\varepsilon$ at N=10 dimensions. In general, the same conclusion as in section 5.3 can be drawn about the relative performance of the algorithms, i.e. DEahcSPX outperformed DE at every dimension. Moreover, the results also show that the performance improvement becomes more substantial with the increase in problem dimensionality. So, from the experimental results of this section it can be concluded that the AHCXLS scheme speeds up DE in general, but particularly significant improvements are obtained at higher dimensionality.

## 5.4 Comparison with other XLS

In order to show the superiority of the newly proposed AHCXLS scheme it was also compared with two other XLS strategies applying in DE algorithm. The first one is the FIR strategy proposed in previous chapter, and this memetic version of DE is denoted as DEfirSPX. The other algorithm, denoted as DExhcSPX, was

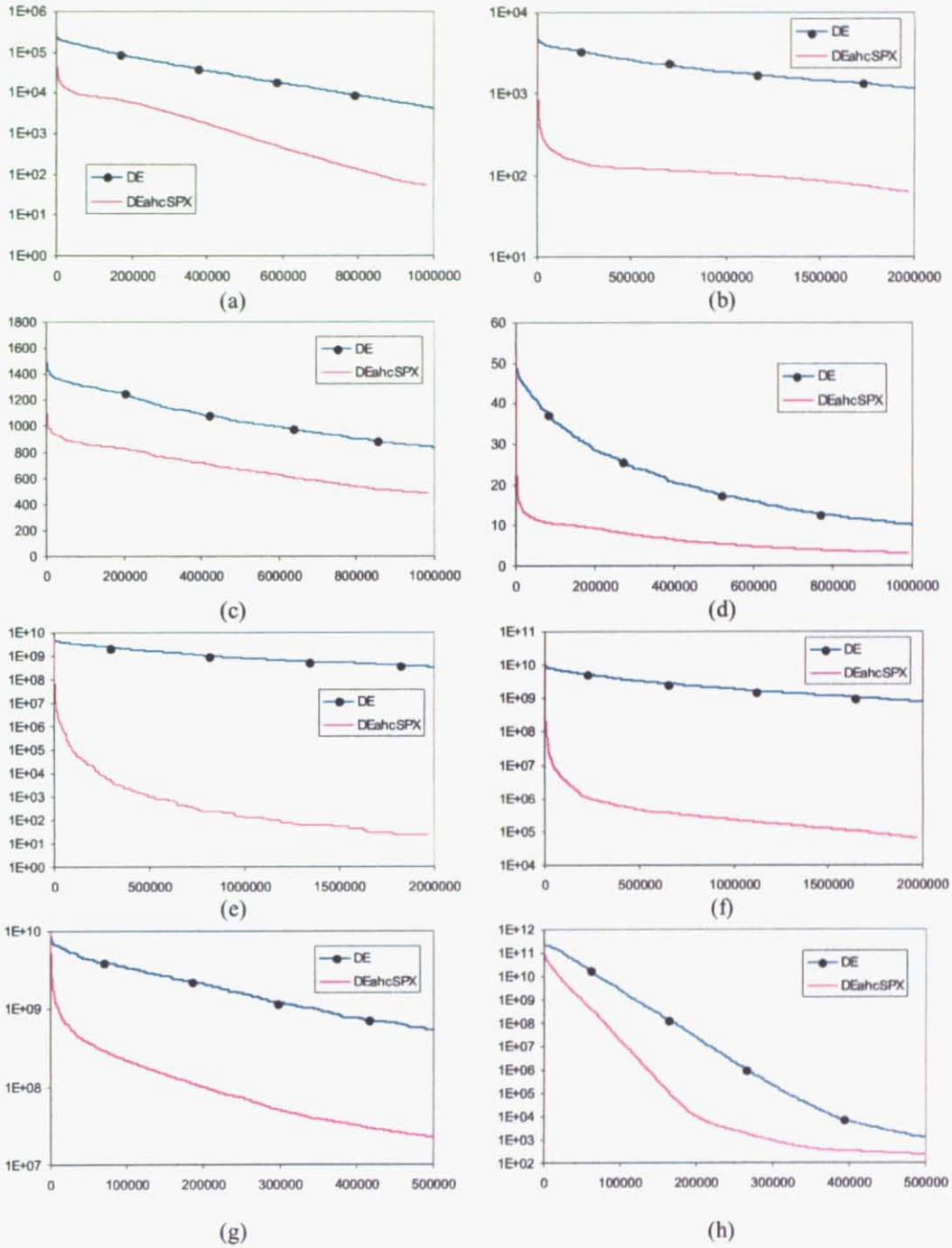


Figure 5.4: Convergence curves to compare the scalability of DE and DEahc-SPX algorithm for selected functions. X-axis represents fitness evaluations (FEs) and Y-axis represents **Error** values. (a) $F_{sph}$(N=100), (b) $F_{grw}$(N=200), (c) $F_{ras}$(N=100), (d) $F_{sal}$(N=100), (e) $F_{pn1}$(N=200), (f) $F_{pn2}$(N=200), (g) $F_3$(N=50) and (h) $F_6$(N=50)

implemented by using the XHC strategy proposed by Lozano *et al.* [69]. Both FIR and XHC belong to the fixed length XLS category and were implemented using SPX crossover operation, in order to have an unbiased comparison. Experiments were performed on the test suite at dimension N=30. Results are presented in Table 5.6 and Table 5.7. The settings for FIR and XHC schemes were chosen as suggested in [86] and [69] respectively. All the other settings are the same as mentioned in section 5.2.2.

Table 5.6: Comparison with other XLS in terms of **Error** values

| | DEahcSPX | DEfirSPX | DExhcSPX |
|---|---|---|---|
| $F_{sph}$ | **1.75E-31 ± 4.99E-31** | 1.22E-27 ± 2.95E-27 | 7.66E-29 ± 1.97E-28 |
| $F_{ros}$ | **4.52E+00 ± 1.55E+01** | 4.84E+00 ± 3.37E+00 | 5.81E+00 ± 4.73E+00 |
| $F_{ack}$ | **2.66E-15 ± 0.00E+00** | 8.35E-15 ± 1.03E-14 | 5.22E-15 ± 2.62E-15 |
| $F_{grw}$ | **2.07E-03 ± 5.89E-03** | 3.54E-03 ± 7.55E-03 | 3.45E-03 ± 7.52E-03 |
| $F_{ras}$ | 2.14E+01 ± 1.23E+01 | 2.27E+01 ± 7.39E+00 | **1.86E+01 ± 7.05E+00** |
| $F_{sch}$ | **4.70E+02 ± 2.96E+02** | 5.23E+02 ± 3.73E+02 | 4.91E+02 ± 4.60E+02 |
| $F_{sal}$ | **1.80E-01 ± 4.08E-02** | 1.84E-01 ± 7.46E-02 | 1.92E-01 ± 4.93E-02 |
| $F_{wht}$ | 3.06E+02 ± 1.10E+02 | 3.11E+02 ± 9.38E+01 | **2.84E+02 ± 1.10E+02** |
| $F_{pn1}$ | **2.07E-02 ± 8.46E-02** | 3.24E-02 ± 3.44E-02 | 2.49E-02 ± 8.61E-02 |
| $F_{pn2}$ | **1.71E-31 ± 5.35E-31** | 1.76E-03 ± 4.11E-03 | 4.39E-04 ± 2.20E-03 |
| $F_1$ | **0.00E+00 ± 0.00E+00** | **0.00E+00 ± 0.00E+00** | **0.00E+00 ± 0.00E+00** |
| $F_2$ | **6.52E-05 ± 4.84E-05** | 1.05E-03 ± 1.29E-03 | 9.40E-04 ± 1.80E-03 |
| $F_3$ | **1.29E+06 ± 9.22E+05** | 1.73E+06 ± 1.22E+06 | 1.54E+06 ± 1.15E+06 |
| $F_4$ | **4.62E+00 ± 8.78E+00** | 1.04E+01 ± 1.75E+01 | 6.69E+00 ± 1.06E+01 |
| $F_5$ | **9.00E+02 ± 4.79E+02** | 1.15E+03 ± 6.68E+02 | 1.01E+03 ± 4.31E+02 |
| $F_6$ | **3.84E+00 ± 3.75E+00** | 1.65E+01 ± 4.72E+01 | 1.41E+01 ± 1.86E+01 |
| $F_7$ | 7.39E-03 ± 6.32E-03 | **4.53E-03 ± 6.92E-03** | 7.98E-03 ± 9.48E-03 |
| $F_8$ | 2.09E+01 ± 1.12E-01 | 2.10E+01 ± 4.61E-02 | 2.09E+01 ± 7.41E-02 |
| $F_9$ | **2.04E+01 ± 8.19E+00** | 2.47E+01 ± 7.72E+00 | 2.80E+01 ± 7.75E+00 |
| $F_{10}$ | **5.27E+01 ± 4.84E+01** | 6.96E+01 ± 5.39E+01 | 6.79E+01 ± 4.80E+01 |

Table 5.7: Comparison with other XLS in terms of **FEs**

| | DEahcSPX | DEfirSPX | DExhcSPX |
|---|---|---|---|
| $F_{sph}$ | **87027.4 ± 3967.3 (50)** | 96588.2 ± 6260.4 (50) | 92111.4 ± 4951.5 (50) |
| $F_{ros}$ | **299913.0 ± 519.5 (2)** | - | - |
| $F_{ack}$ | **129211.6 ± 5168.6 (50)** | 142169.88 ± 8137.9 (50) | 139982.1 ± 7096.2 (50) |
| $F_{grw}$ | **121579.2 ± 79563.4 (43)** | 146999.76 ± 87855.0 (38) | 153119.1 ± 93613.4 (37) |
| $F_{pn1}$ | **96149.0 ± 61787.7 (46)** | 126486.56 ± 66369.7 (44) | 122129.1 ± 68013.8 (44) |
| $F_{pn2}$ | **85360.2 ± 6390.6 (50)** | 135395.48 ± 73557.7 (43) | 106820.1 ± 41154.6 (48) |
| $F_1$ | **89417.8 ± 4117.6 (50)** | 101022.68 ± 7656.8 (50) | 97470.6 ± 7700.0 (50) |
| $F_2$ | **299279.4 ± 3685.9 (3)** | - | - |
| $F_7$ | **148067.7 ± 68996.3 (42)** | 169019.16 ± 84155.5 (36) | 175486.0 ± 80658.6 (37) |

The performance difference among these three XLS methods is not obvious from Table 5.6 because at the end of the search all of them reached similar error values, though DEahcSPX found slightly better error values in almost every case. However, the results presented in Table 5.7 reveals that the newly proposed DEahcSPX algorithm was faster than the other two variants of DE. Statistical analysis of the numbers of FEs needed to reach the given accuracy level (i.e the results of Table 5.7) was performed using two-tailed Student's $t$-test, and it was found that the

differences between the results of DEahcSPX and the other two algorithms are statistically significant at a level of 0.05 for all the functions in which the algorithms found convergences in at least 40 trials (i.e. $F_{sph}$, $F_{ack}$, $F_{pn1}$, $F_{pn2}$ and $F_1$). Besides, the most prominent advantage of the AHCXLS scheme over the other two is that it is free from the lookup for the best length for the local search and thereby does not need any additional parameter. In contrast, for best results, XHC and FIR schemes need to tune two and one parameters respectively, which in turn should be determined experimentally. Moreover, AHCXLS is also useful for lower dimensional problems whereas the FIR scheme is only suitable for high dimensional optimization. At lower dimension (e.g. at N=10) the performance of DEfirSPX was not significantly different from that of DE, and even poor in some cases. Furthermore, in a brief experimentation it was found that the performance difference among the proposed algorithm and the other two variants become more significant at higher dimensions.

## 5.5 Comparison with other EC

Many XLS-oriented evolutionary algorithms for real parameter optimization are now available in the literature. This subsection presents a performance comparison between the proposed algorithm and some other hybrid GAs with LS. Two GA models, Minimal Generation Gap (MGG) [109] and Generalized Generation Gap (G3) [22], have drawn much attention. Both of these models in fact induce an XLS on the neighborhood of the parents by generating multiple offspring using some crossover operation [69]. Over the past few years, substantial research effort has been spent to develope more sophisticated crossover operations for GA and many outstanding schemes have been proposed, such as BLX-$\alpha$ crossover [27], unimodal normal distribution crossover (UNDX) [96], simplex crossover (SPX) [131], and parent centric crossover (PCX) [22]. The respective researches have shown that UNDX and SPX perform best with the MGG and PCX performs best with the G3 generational models. Therefore, in these experiments comparisons were performed using the algorithms MGG+UNDX, MGG+SPX, G3+PCX and G3+SPX and the results are shown in Table 5.8 and 5.9. The performance of G3+SPX was similar to or poorer than that of MGG+SPX. Hence, only the results of MGG+SPX were presented. The MGG model was setup with $P = 300$, $\lambda = 4$ offspring, generated from $\mu$ parents, where $\mu = 6$ was used for UNDX and $\mu = 3$ was used for SPX. For G3 model $P = 100$, $\mu = 3$, and $\lambda = 2$ were used.

Table 5.8: Comparison with other RCMAs in terms of **Error** values (N=30)

| | DEahcSPX | MGG+UNDX | G3+PCX | MGG+SPX |
|---|---|---|---|---|
| $F_{sph}$ | 1.75E-31 ± 4.99E-31 | 1.37E-11 ± 1.94E-11 | **3.58E-81 ± 1.36E-81** ‡ | 8.75E+00 ± 2.87E+00 |
| $F_{ros}$ | 4.52E+00 ± 1.55E+01 | 2.81E+01 ± 1.23E+01 | **4.18E+00 ± 9.68E+01** | 1.38E+03 ± 6.45E+02 |
| $F_{ack}$ | **2.66E-15 ± 0.00E+00** | 8.23E-07 ± 4.64E-07 | 1.48E+01 ± 4.17E+00 ‡ | 1.68E+00 ± 2.99E-01 |
| $F_{grw}$ | 2.07E-03 ± 5.89E-03 | **2.96E-04 ± 1.48E-03** | 1.07E-02 ± 1.30E-02 ‡ | 1.09E+00 ± 2.24E-02 |
| $F_{ras}$ | 2.14E+01 ± 1.23E+01 | **1.35E+00 ± 1.03E+00** | 1.75E+02 ± 3.37E+01 ‡ | 5.78E+00 ± 1.83E+00 |
| $F_{sch}$ | **4.70E+02 ± 2.96E+02** | 4.12E+03 ± 1.72E+03 | 4.04E+03 ± 1.09E+03 ‡ | 8.70E+03 ± 2.41E+02 |
| $F_{sal}$ | 1.80E-01 ± 4.08E-02 | **1.50E-01 ± 4.95E-02** | 4.64E+00 ± 4.74E+00 ‡ | 3.82E-01 ± 4.29E-02 |
| $F_{wht}$ | **3.06E+02 ± 1.10E+02** | 4.28E+02 ± 3.82E+01 | 7.90E+02 ± 1.27E+02 ‡ | 3.28E+03 ± 2.77E+03 |
| $F_{pn1}$ | **2.07E-02 ± 8.46E-02** | 4.93E-02 ± 3.50E-02 | 4.35E+00 ± 6.94E+00 ‡ | 2.57E-01 ± 6.90E-02 |
| $F_{pn2}$ | **1.71E-31 ± 5.35E-31** | 4.39E-04 ± 2.20E-03 | 1.50E+01 ± 1.58E+01 ‡ | 2.29E+00 ± 3.72E-01 |
| $F_1$ | **0.00E+00 ± 0.00E+00** | 2.83E-11 ± 3.33E-11 | 3.52E-13 ± 1.22E-13 ‡ | 4.71E+04 ± 4.21E+03 |
| $F_2$ | 6.52E-05 ± 4.84E-05 | 1.41E+00 ± 7.15E-01 | **4.14E-12 ± 1.21E-12** ‡ | 3.96E+04 ± 3.89E+03 |
| $F_3$ | 1.29E+06 ± 9.22E+05 | 8.76E+05 ± 2.98E+05 | **1.07E+03 ± 1.29E+03** | 7.16E+08 ± 1.34E+08 |
| $F_4$ | **4.62E+00 ± 8.78E+00** | 5.01E+01 ± 3.62E+01 | 9.35E+04 ± 2.66E+04 | 4.45E+04 ± 3.73E+03 |
| $F_5$ | **9.00E+02 ± 4.79E+02** | 1.67E+03 ± 6.01E+02 | 8.13E+03 ± 2.65E+03 ‡ | 3.34E+04 ± 2.11E+03 |
| $F_6$ | **3.84E+00 ± 3.75E+00** | 1.79E+02 ± 2.38E+02 | 1.34E+02 ± 2.48E+02 | 1.56E+10 ± 1.47E+09 |
| $F_7$ | 7.39E-03 ± 6.32E-03 | **7.26E-03 ± 8.19E-03** | 2.01E-02 ± 1.85E-02 ‡ | 1.02E+04 ± 4.71E+02 |
| $F_8$ | 2.09E+01 ± 1.12E-01 | 2.09E+01 ± 5.62E-02 | 2.11E+01 ± 6.67E-12 ‡ | 2.10E+01 ± 4.06E-02 |
| $F_9$ | **2.04E+01 ± 8.19E+00** | 4.65E+01 ± 5.41E+01 | 2.44E+02 ± 3.98E+01 ‡ | 3.15E+02 ± 1.04E+01 |
| $F_{10}$ | 5.27E+01 ± 4.84E+01 | **4.76E+01 ± 5.03E+01** | 3.89E+02 ± 9.96E+01 ‡ | 5.31E+02 ± 2.85E+01 |

‡ Algorithm converged before using the maximum allowed fitness evaluations

Table 5.9: Comparison with other RCMAs in terms of **FEs** (N=30)

| | DEahcSPX | MGG+UNDX | G3+PCX | MGG+SPX |
|---|---|---|---|---|
| $F_{sph}$ | 87027.4 ± 3967.3 (50) | 200515.5 ± 6743.2 (50) | **2640.1 ± 104.9 (50)** ‡ | - |
| $F_{ros}$ | 299913.0 ± 519.5 (2) | - | **177783.9 ± 71145.1 (34)** ‡ | - |
| $F_{ack}$ | **129211.6 ± 5168.6 (50)** | 294226.7 ± 4359.5 (40) | - | - |
| $F_{grw}$ | **121579.2 ± 79563.4 (43)** | 238310.7 ± 13968.8 (42) | 14560.6 ± 12576.4 (20) ‡ | - |
| $F_{ras}$ | - | 299583.6 ± 2102.0 (2) | - | - |
| $F_{pn1}$ | **96149.0 ± 61787.7 (46)** | 185258.4 ± 7436.3 (44) | - | - |
| $F_{pn2}$ | **85360.2 ± 6390.6 (50)** | 209272.8 ± 19680.5 (48) | - | - |
| $F_1$ | 89417.8 ± 4117.6 (50) | 206630.6 ± 5771.4 (50) | **2649.2 ± 121.3 (50)** ‡ | - |
| $F_2$ | 299279.4 ± 3685.9 (3) | - | **13290.7 ± 569.1 (50)** ‡ | - |
| $F_6$ | - | - | **177617.4 ± 110147.4 (24)** | - |
| $F_7$ | **148067.7 ± 68996.3 (42)** | 257533.8 ± 34064.9 (35) | 9314.4 ± 4281.8 (20) ‡ | - |

‡ Algorithm converged before using the maximum allowed fitness evaluations

In these experiments, the MGG+SPX algorithm could not achieve the target accuracy levels for any function of the test suite. The MGG+UNDX algorithm achieved a slightly better error average for some functions ($F_{grw}$, $F_{ras}$, $F_{sal}$, $F_3$, $F_7$ and $F_{10}$) but was outperformed by DEahcSPX for the other functions. Moreover, according to Table 5.9 the average fitness evaluations used by DEahcSPX were fewer than that used by MGG+UNDX to achieve the target accuracy levels $\varepsilon$. The performance of G3+PCX was really outstanding for unimodal functions like $F_{sph}$, $F_{ros}$, $F_1$ and $F_2$. But its performance was really poor for the multimodal functions. In most of the cases, the algorithm converged quickly without reaching the error accuracy level and without exhausting the maximum fitness evaluations as indicated in Table 5.8 and 5.9. So, in general it can be concluded from Table 5.8 and 5.9 that the proposed DEahcSPX exhibits overall better performance than the other algorithms of the tables. These results also establish it as a competitive alternative for real parameter optimization problems.

Table 5.10: Comparison with MA-S2 [94]

| | Range | MA-S2 | DE | DEahcSPX |
|---|---|---|---|---|
| $F_{Sphere}$ | $[-5.12, 5.12]^{30}$ | [Global min in 100% ] 7198 Evals | 4.27E-05 ± 1.18E-05 | 2.56E-06 ± 1.06E-06 |
| $F_{Griewank}$ | $[-600, 600]^{10}$ | 2.80E-04 ± 8.28E-04 | 3.44E-02 ± 2.05E-02 | 2.77E-02 ± 1.24E-02 |
| $F_{Bump}$ | $[0, 10]^{20}$ | 7.34E-01 ± 2.22E-02 | 0.7999 ± 0.0024 | **8.02E-01 ± 3.87E-03** |
| $F_{ShekelsFoxHole}$ | $[-65.536, 65.536]^2$ | [Global min in 100% ] 9634 Evals | 0.998 ± 5.09E-16 | 0.998 ± 7.20E-17 |
| $F_{Schwefel}$ | $[-500, 500]^{10}$ | [Global min in 80% ] | 1.27E-04 ± 3.73E-13 | 1.27E-04 ± 0.00E+00 |
| $F_{Rosenbrock}$ | $[-2.048, 2.048]^{30}$ | 2.57E+04 ± 6.00E+01 | 2.44E+01 ± 7.32E-01 | **2.15E+01 ± 7.35E-01** |
| $F_{Step}$ | $[-5.12, 5.12]^{30}$ | 1.63E-01 ± 3.10E-02 | [13967.7 ± 738.07]† | **[13846.6 ± 621.95]†** |
| $F_{Rastrigin}$ | $[-5.12, 5.12]^{20}$ | 1.55E-01 ± 7.10E-02 | 2.14E-01 ± 1.53E-01 | **6.02E-02 ± 1.00E-01** |

† Global optimum 0.0 found using these FEs counts

The proposed DEahcSPX algorithm was also compared with other MAs with binary coding and real coding using the published results. To show that the proposed AHCXLS is equally suitable for the exponential crossover scheme, in these comparisons exponential crossover was used in DE and DEahcSPX instead of binary crossover. First, comparison was performed with self-adaptive MA scheme MA-S2, which is the best of the two adaptive MAs proposed in [94], and also exhibited overall superior performance compared to nine other traditional MAs. The comparative results are presented in Table 5.10 in terms of 8 benchmark functions used in [94], among which the Bump function ($F_{Bump}$) is a constrained maximization problem whether all the others are unconstrained minimization problems. The maximum FEs allowed to solve each function was 40,000 except for $F_{Bump}$ where it was 100,000. The results presented are average of 20 repeated runs as in [94]. From Table 5.10 it can be found that for $F_{Bump}$, $F_{Rosenbrock}$, $F_{Step}$ and $F_{Rastrigin}$ functions

the DEahcSPX algorithm clearly outperformed the MA-S2 algorithm. And for the other 4 functions apparently it seems that MA-S2 exhibited superior performance. But, if we consider the accuracy level that can be achieved using 10 bit binary coding for the respective search ranges in MA-S2 then the performance of DEahcSPX using real coding was very competitive or even better.

Table 5.11: Comparison with RCMA [69]

| | Range | RCMA-XHC | | DE | | DEahcSPX | |
|---|---|---|---|---|---|---|---|
| | | A | B | A | B | A | B |
| $F_{Sph}$ | $[-5.12, 5.12]^{25}$ | **6.50E-101** | **1.10E-105** | 6.91E-18 | 8.63E-19 | 2.58E-20 | 6.16824E-21 |
| $F_{Ros}$ | $[-5.12, 5.12]^{25}$ | 2.20E+00 | **6.00E-04** | 1.32E-01 | 2.48E-02 | **1.20E-02** | 1.13E-03 |
| $F_{Sch}$ | $[-65.536, 65.536]^{25}$ | **3.80E-07** | **4.50E-09** | 1.47E+01 | 5.53E+00 | 2.48E-01 | 4.98E-02 |
| $F_{Ras}$ | $[-5.12, 5.12]^{25}$ | 1.40E+00 | **32.00%** | 6.59E-11 | 6.09E-12 | **3.48404E-12** | 6.75016E-14 |
| $F_{Gri}$ | $[-600, 600]^{25}$ | 1.30E-02 | **30.00%** | 1.48E-04 | 1.54E-14 | **3.75566E-14** | 18% |
| $P_{sle}$ | $[-127, 127]^{10}$ | 5.50E+01 | 7.90E-01 | 8.25E-04 | 1.98E-05 | **6.17E-04** | **8.76E-08** |
| $P_{Cheb}$ | $[-6.4, 6.35]^{6}$ | 1.40E+02 | 9.20E+00 | **0.00E+00** | **100%** | **0.00E+00** | **100%** |
| $P_{fms}$ | $[-512, 512]^{9}$ | 7.70E+00 | **40%** | 2.68E+00 | 1.10E-23 | **2.22E+00** | 1.43E-04 |

A: Average of the minimum fitness found; B: best of all minimum fitness or percentage of run that found global optimum

Then, the proposed algorithm was compared with the results of the RCMA presented in [69]. Comparing with other 21 variants of real coded memetic algorithms Lozano *et. al* showed that, in general, their proposed RCMA outperforms all other algorithms [69]. Table 5.11 shows comparative results for 5 benchmark functions and 3 real-world problems as used in [69]. The same performance measure criteria as used in [69] was used; **A**: average of minimum fitness found in 50 repeated runs; **B**: best of all minimum fitness in 50 runs or the percentage of run in which the global optimum was found (if some runs located the global minimum). The maximum FEs allowed in each run was 100,000. From Table 5.11 it can be found that the performance of RCMA was better than DEahcSPX only for $F_{Sph}$ and $F_{Sch}$, and in all other case the average performance of the proposed algorithm was better than that of RCMA. Hence, in an average the DEahcSPX algorithm outperformed the RCMA on the studied benchmark and on the real world problems.

Finally, the proposed algorithm was compared with the Dynamic Multi-Swarm Particle Swarm Optimizer with local search (DMS-PSO) algorithm using the results reported in [65]. Table 5.12 compares DMS-PSO, DE and DEahcSPX for the 10 benchmark functions used in the test suite ($F_1$ to $F_{10}$). The results are the average of 25 runs under the same experimental conditions. As shown in Table 5.12, for $F_1$, $F_2$, $F_3$ and $F_{10}$ DMS-PSO outperformed DEahcSPX. In particular, the performance of DMS-PSO was extra-ordinary for the first three unimodal functions. In contrast, for $F_4$, $F_6$, $F_7$ and $F_9$, DEahcSPX outperformed DMS-PSO considerably. For the other

Table 5.12: Comparison with DMS-PSO [65] at N=30

| | DMS-PSO | DE | DEahcSPX |
|---|---|---|---|
| $F_1$ | [**5026.3 ± 72.463**] † | [64546.85 ± 1097.25] † | [54959.52 ± 1184.39] † |
| $F_2$ | [**125520 ± 17371**] † | 1.33E-02 ± 6.72E-03 | 4.87E-05 ± 2.74E-05 |
| $F_3$ | **1.63E-06 ± 3.92E-06 (84%)** | 1.85E+06 ± 1.05E+06 | 9.80E+05 ± 5.56E+05 |
| $F_4$ | 2.55E+03 ± 3.06E+02 | 9.16E+01 ± 4.53E+01 | **1.14E+00 ± 6.16E-01** |
| $F_5$ | 2.19E+03 ± 8.26E+02 | 3.27E+03 ± 8.75E+02 | 2.15E+03 ± 7.04E+02 |
| $F_6$ | 4.78E-01 ± 1.32E+00 (98%) | [161762.1 ± 3765.823] † | [**146963.32 ± 5613.87**] † |
| $F_7$ | 7.00E-03 ± 4.54E-03 (96%) | 7.96E-02 ± 5.99E-02 (16%) | **5.64E-04 ± 2.03E-03 (98%)** |
| $F_8$ | 2.00E+01 ± 2.30E-04 | 2.09E+01 ± 5.35E-02 | 2.10E+01 ± 5.52E-02 |
| $F_9$ | 1.76E+01 ± 3.02E+00 | [60441.55 ± 2015.97] † | [**57163.12 ± 1879.38**] † |
| $F_{10}$ | **3.74E+01 ± 5.29E+00** | 1.20E+02 ± 1.42E+01 | 9.51E+01 ± 1.45E+01 |

† Target accuracy level achieved using these FEs counts

two functions $F_5$ and $F_8$ no performance difference was observed. The results of Table 5.12 suggest that DMS-PSO is exceptional in solving unimodal problems, and can also handle multimodal problems competitively. On the other hand DEahcSPX exhibited superior performance in solving multimodal functions compared to DMS-PSO.

In all of the above comparisons in Table 5.10, 5.11 and 5.12 DEahcSPX consistently exhibited superior performance compared to original DE which establishes that AHCXLS scheme is equally suitable for the exponential crossover scheme.

Table 5.13: Study on the suitability of AHCXLS for DESP

| | DESP | DESPahcSPX |
|---|---|---|
| $F_{sph}$ | [50808.08 ± 1178.07] † | [**40956.12 ± 967.89**] † |
| $F_{ros}$ | 1.42E+01 ± 1.61E+01 | **1.30E+01 ± 1.72E+01** |
| $F_{ack}$ | [71779.96 ± 1507.12] † | [**58920.68 ± 1316.83**] † |
| $F_{grw}$ | [53679.6 ± 3113.15] † | [**43412.68 ± 2199.17**] † |
| $F_{ras}$ | [108502.52 ± 4027.63] † | [**102456.6 ± 4775.19**] † |
| $F_{sch}$ | 3.82E-04 ± 0.00E+00 | 3.82E-04 ± 0.00E+00 |
| $F_{sal}$ | 1.92E-01 ± 2.77E-02 | **1.36E-01 ± 4.90E-02** |
| $F_{wht}$ | 3.47E+01 ± 5.88E+01 | **2.83E+01 ± 5.60E+01 (4%)** |
| $F_{pn1}$ | [44659.36 ± 1361.99] † | [**36064.04 ± 1181.63**] † |
| $F_{pn2}$ | [49224.72 ± 1224.12] † | [**39508.88 ± 1035.98**] † |
| $F_1$ | [50269.24 ± 1183.61] † | [**41245.92 ± 1167.10**] † |
| $F_2$ | 6.86E-07 ± 5.81E-07 (80%) | **2.83E-07 ± 3.94E-07 (92%)** |
| $F_3$ | 1.89E+05 ± 1.04E+05 | **1.74E+05 ± 1.04E+05** |
| $F_4$ | 3.74E-02 ± 8.86E-02 | **2.11E-02 ± 3.17E-02** |
| $F_5$ | 4.25E+02 ± 4.07E+02 | **3.78E+02 ± 4.37E+02** |
| $F_6$ | 2.60E+01 ± 2.79E+01 | **2.20E+01 ± 2.55E+01** |
| $F_7$ | 1.60E-02 ± 1.17E-02 (52%) | **1.28E-02 ± 8.51E-03 (76%)** |
| $F_8$ | 2.09E+01 ± 3.66E-02 | 2.09E+01 ± 5.90E-02 |
| $F_9$ | [83861.84 ± 2510.90] † | [**81091.8 ± 3372.61**] † |
| $F_{10}$ | 5.71E+01 ± 9.83E+00 | **5.35E+01 ± 8.13E+00** |

† Target accuracy level achieved using these FEs counts

## 5.6 Other Studies of AHCXLS scheme

In all experiments $F = 0.9$ and $C_r = 0.9$ were used as the parameter setting for all algorithms. As mentioned earlier, because of the sensitivity of DE to its control parameter, some variants with adaptive control parameter have been proposed [15, 68, 150]. In order to show that the proposed AHCXLS scheme can also accelerate such adaptive DE variants, it was incorporated in a recent DE variant with self-adaptive control parameters (DESP) proposed in [15]. The new variant was named as DESPahcSPX. The comparative results (average of 25 runs) with the same settings as in [15] (N=30 and P=100) are reported in Table 5.13. The results of Table 5.13 suggest that integration of AHCXLS in DESP has certainly accelerated the algorithm. These results also indicate that the acceleration of DE by AHCXLS scheme is not influenced by the parameter settings. Hence, the AHCXLS scheme can be similarly useful for performance enhancement of other self-adaptive DE-variants.

The only parameter AHCXLS scheme includes is $n_p$, the number of parents participating in the crossover operation. The authors of SPX operation suggested that the number of parents should be 3 or 4 [131], and hence $n_p = 3$ was used in this study. However, the effect of $n_p$ on the performance was studied using $n_p = 4$, 5, 6, 8, 10, 12 and 15. Table 5.14 compares the performance for some of the choices. From Table 5.14 it seems that in general a higher number of parents, $n_p$, can slightly improve the performance of the algorithm. However, the effect should be studied in more detail by varying population size and problem dimension which is out of the scope of this research.

Another issue in the AHCXLS scheme is the selection of parents other than the best individual of the generation. In this chapter, they were chosen randomly. However, incorporating the knowledge obtained during the search in selecting parents (other than the best) for SPX operation can further improve the performance. The effect of positive assortative mating (PAM) and negative assortative mating (NAM) on the algorithm performance were briefly studied. After selecting the first parent, PAM (NAM) selects other individuals with most (least) phenotype similarity. Here Euclidean distance between chromosomes was used as a measure of similarity between them. The results shown in Table 5.15 suggest that NAM can be useful to improve the performance of the algorithm, mostly for unimodal functions. However, considering the performance improvement achieved and the additional computational cost incurred in NAM, the random mating used in this work can be used as a computationally less expensive approach. Many other sophisticated mechanisms

Table 5.14: Study of $n_p$ for AHCXSL operation (N=30)

| | $n_p = 5$ | $n_p = 8$ |
|---|---|---|
| $F_{sph}$ | [83198.68 ± 4951.30] † | [82987.64 ± 3546.02] † |
| $F_{ros}$ | 2.23E+00 ± 2.27E+00 | 1.56E+00 ± 2.44E+00 |
| $F_{ack}$ | [124043.36 ± 5526.48] † | 4.62E-02 ± 2.31E-01 (96%) |
| $F_{grw}$ | 2.76E-03 ± 4.73E-03 (72%) | **1.58E-03 ± 4.91E-03 (88%)** |
| $F_{ras}$ | 2.09E+01 ± 6.72E+00 | 1.82E+01 ± 5.42E+00 |
| $F_{sch}$ | 5.50E+02 ± 3.11E+02 | 5.26E+02 ± 3.98E+02 |
| $F_{sal}$ | 1.72E-01 ± 8.91E-02 | 1.61E-01 ± 4.90E-02 |
| $F_{wht}$ | 3.03E+02 ± 1.00E+02 | 3.13E+02 ± 1.02E+02 |
| $F_{pn1}$ | 1.66E-02 ± 3.88E-02 (84%) | **8.29E-03 ± 4.15E-02 (96%)** |
| $F_{pn2}$ | 6.54E-02 ± 3.18E-01 (80%) | 1.45E-01 ± 7.19E-01 (88%) |
| $F_1$ | [84284 ± 3930.25] † | [81488.92 ± 3881.04] † |
| $F_2$ | 2.36E-05 ± 3.78E-05 | **1.84E-05 ± 2.73E-05 (4%)** |
| $F_3$ | 3.91E+05 ± 1.93E+05 | **3.65E+05 ± 2.03E+05** |
| $F_4$ | 5.36E-01 ± 5.40E-01 | **5.01E-01 ± 7.05E-01** |
| $F_5$ | 8.20E+01 ± 1.31E+02 | 1.19E+02 ± 1.76E+02 |
| $F_6$ | 1.55E+00 ± 2.49E+00 (20%) | 9.37E-01 ± 1.56E+00 (12%) |
| $F_7$ | 6.89E-03 ± 9.63E-03 (76%) | 4.63E-03 ± 6.45E-03 (84%) |
| $F_8$ | 2.10E+01 ± 4.69E-02 | 2.09E+01 ± 4.26E-02 |
| $F_9$ | 1.98E+01 ± 6.45E+00 | **1.86E+01 ± 4.91E+00** |
| $F_{10}$ | 5.22E+01 ± 6.11E+01 | 5.96E+01 ± 6.69E+01 |
| | $n_p = 10$ | $n_p = 15$ |
| $F_{sph}$ | [80547.52 ± 3112.62] † | **[77132.88 ± 4603.41]** † |
| $F_{ros}$ | 1.87E+00 ± 2.11E+00 | **1.49E+00 ± 2.15E+00** |
| $F_{ack}$ | [121583.2 ± 5885.61] † | **[116291.16 ± 4177.22]** † |
| $F_{grw}$ | 3.44E-03 ± 7.99E-03 (76%) | 2.86E-03 ± 4.48E-03 (68%) |
| $F_{ras}$ | 1.80E+01 ± 5.47E+00 | **1.73E+01 ± 6.32E+00** |
| $F_{sch}$ | **4.91E+02 ± 2.20E+02** | 5.73E+02 ± 2.97E+02 |
| $F_{sal}$ | **1.50E-01 ± 5.00E-02** | 1.54E-01 ± 5.72E-02 |
| $F_{wht}$ | 3.17E+02 ± 9.11E+01 | 2.81E+02 ± 1.07E+02 |
| $F_{pn1}$ | 2.49E-02 ± 1.05E-01 (92%) | 2.49E-02 ± 8.61E-02 (88%) |
| $F_{pn2}$ | 1.44E-01 ± 7.19E-01 (92%) | **8.79E-04 ± 3.04E-03 (92%)** |
| $F_1$ | [79089.92 ± 3227.048] † | **[78427.72 ± 4491.97]** † |
| $F_2$ | 3.62E-05 ± 3.17E-05 | 2.35E-05 ± 2.88E-05 |
| $F_3$ | 4.17E+05 ± 2.44E+05 | 4.05E+05 ± 1.80E+05 |
| $F_4$ | 5.98E-01 ± 7.76E-01 | 1.38E+00 ± 3.60E+00 |
| $F_5$ | 7.41E+01 ± 8.41E+01 | **6.98E+01 ± 7.52E+01** |
| $F_6$ | **1.65E+00 ± 3.22E+00 (24%)** | 7.18E-01 ± 1.16E+00 (16%) |
| $F_7$ | **3.16E-03 ± 4.90E-03 (92%)** | 4.83E-03 ± 7.22E-03 (92%) |
| $F_8$ | 2.09E+01 ± 4.18E-02 | 2.09E+01 ± 4.29E-02 |
| $F_9$ | 1.98E+01 ± 6.95E+00 | 1.99E+01 ± 8.55E+00 |
| $F_{10}$ | 5.79E+01 ± 6.67E+01 | **4.55E+01 ± 5.69E+01** |

† Target accuracy level achieved using these FEs counts

Table 5.15: Comparison with different mating selection mechanisms for the SPX operation in DEahcSPX

| | PAM | NAM |
|---|---|---|
| $F_{sph}$ | [71637.24 ± 3602.63] † | **[57435.96 ± 3656.022]** † |
| $F_{ros}$ | 1.16E+00 ± 1.53E+00 | **2.55E-01 ± 7.90E-01** |
| $F_{ack}$ | [139184.34 ± 6239.022] † | **[118600.28 ± 5720.94]** † |
| $F_{grw}$ | 4.87E-03 ± 9.30E-03 (66%) | **1.72E-03 ± 5.19E-03 (88%)** |
| $F_{ras}$ | 2.38E+01 ± 7.81E+00 | **1.88E+01 ± 6.08E+00** |
| $F_{sch}$ | 6.28E+02 ± 3.82E+02 | **5.10E+02 ± 2.94E+02** |
| $F_{sal}$ | 2.45E-01 ± 5.71E-02 | **1.90E-01 ± 3.64E-02** |
| $F_{wht}$ | **2.80E+02 ± 1.11E+02** | 3.39E+02 ± 6.35E+01 |
| $F_{pn1}$ | 5.62E-02 ± 2.00E-01 (86%) | **6.22E-03 ± 2.49E-02 (94%)** |
| $F_{pn2}$ | 8.79E-04 ± 3.01E-03 (92%) | **4.39E-04 ± 2.17E-03 (96%)** |
| $F_1$ | [93801.62 ± 4809.75] † | **[78479.86 ± 4113.097]** † |
| $F_2$ | 2.27E-04 ± 2.59E-04 | **9.34E-05 ± 1.53E-04** |
| $F_3$ | **4.38E+05 ± 2.52E+05** | 5.16E+05 ± 2.27E+05 |
| $F_4$ | 6.51E+00 ± 3.38E+01 | **7.74E-01 ± 1.05E+00** |
| $F_5$ | 1.52E+02 ± 1.73E+02 | **5.14E+01 ± 5.60E+01** |
| $F_6$ | 4.78E+00 ± 1.06E+01 (10%) | **2.07E+00 ± 2.39E+00 (10%)** |
| $F_7$ | 9.50E-03 ± 9.31E-03 (33%) | **5.42E-03 ± 7.71E-03 (78%)** |
| $F_8$ | 2.09E+01 ± 5.37E-02 | 2.09E+01 ± 5.80E-02 |
| $F_9$ | 2.11E+01 ± 6.01E+00 | **1.97E+01 ± 5.62E+00** |
| $F_{10}$ | 8.65E+01 ± 7.53E+01 | **7.62E+01 ± 7.42E+01** |

† Target accuracy level achieved using these FEs counts

available can be used for getting online feed-back from the search that can help to improve the quality of the local tuning at the expense of some computational effort [95, 11, 60].

# Chapter 6

# An Algorithm for Reconstructing Genetic Networks

This chapter is devoted to the study of gene regulatory network reconstruction in simulation. First the evolutionary framework that was developed for reconstructing genetic networks from time series data using the decoupled S-system as the model of regulatory interaction among the genes is presented. Then the capability of the reconstruction methodology to predict the network topology and the regulatory parameters is investigated varying network dimension, amount of gene expression data for inference and the noise level present in the expression profile.

From the study of Chapter 4 and from many other studies it was evident that any evolutionary algorithm in its basic form is inadequate for estimating the exact network structure and the regulatory parameter values of the network. Therefore, we need to develop more sophisticated and advanced methodology for inferring the network topology as well as kinetic parameters. And incorporating some intelligent heuristics in the reconstruction mechanism often significantly improves the capability of the algorithm for such complex problems.

The developed system for inferring the transcriptional regulations in a biochemical network is an enhanced memetic algorithm that takes advantage of the embedded local search mechanism. The proposed memetic algorithm was designed targeting several issues such as identifying robust transcriptional regulations, estimating precise kinetic parameters, attaining skeletal network architecture and above all computational efficiency. The optimization engine, the core of the algorithm, was implemented using a reliable and robust optimization algorithm DEfirSPX developed and analyzed in chapter 4. Besides, the exploitation capability of a hill climbing local search heuristic for efficient identification of sparse network structure was used.

Moreover, double optimization was employed for detecting robust regulatory interactions and special means was taken for maintaining the population diversity for global convergence.

## 6.1 Reconstruction Algorithm

In the developed methodology for estimating the S-system parameters, the objective function (3.17) or (3.18) is optimized for each sub-problem $i$ ($i = 1 \cdots N$) using a modified DEfirSPX algorithm with a local search procedure (described later). For explaining the inference algorithm the case of gene $i$ is taken as an example. Same is used for other sub-problems to obtain a complete set of parameters for the full network.

The genetic network reconstruction problem is strongly non-linear and highly multimodal that causes the search procedure to stick in a local optimum. Therefore, to avoid premature convergence to some local minima optimization was performed using a two step method commonly known as double optimization. In double optimization a second phase of optimization is performed on different local solutions obtained in first phase. Double optimization is useful for identifying essential parameters automatically and hence was found useful for detecting robust regulatory interactions in genetic networks [54, 6, 87]. In each phase, the parameter values of the gene are represented as an individual of DEfirSPX (as mentioned earlier). Two phases of the proposed algorithm are explained taking the sub-problem corresponding to gene $i$ as an example.

**Phase 1:** At first, $\Gamma$ repeated trails of optimization of the fitness function of (3.18) were performed starting from different random initial solutions. In each of these trials, the optimization was performed using DEfirSPX algorithm with a hill climbing local search procedure (explained later). Each of these trial runs gives a solution of the sub-problem i.e. a set of parameters for the target gene-$i$. However, some optimization trials may converge to some local optimum and may fail to infer the actual parameter set.

**Phase 2:** Since we assume some solutions in *Phase 1* are possibly local solutions, they may not identify all the target regulations and the parameter values may be significantly different in different solutions. Therefore, in order to obtain more robust network structure and accurate parameter values another optimization was performed on the elite individuals from different trails of *Phase 1*. The best individuals from each of the $\Gamma$ trials were selected and some random individual were

added to form the initial population and then the optimization was performed using the same fitness function and algorithm.

If the solutions obtained from different trials of *Phase 1* are local solutions they retain some essential regulations. So applying another optimization on these solutions we can expect to identify all the correct regulations with accurate strengths and avoid the loss of any necessary interaction. The schematic diagram of the complete algorithm is illustrated in Fig. 6.1.

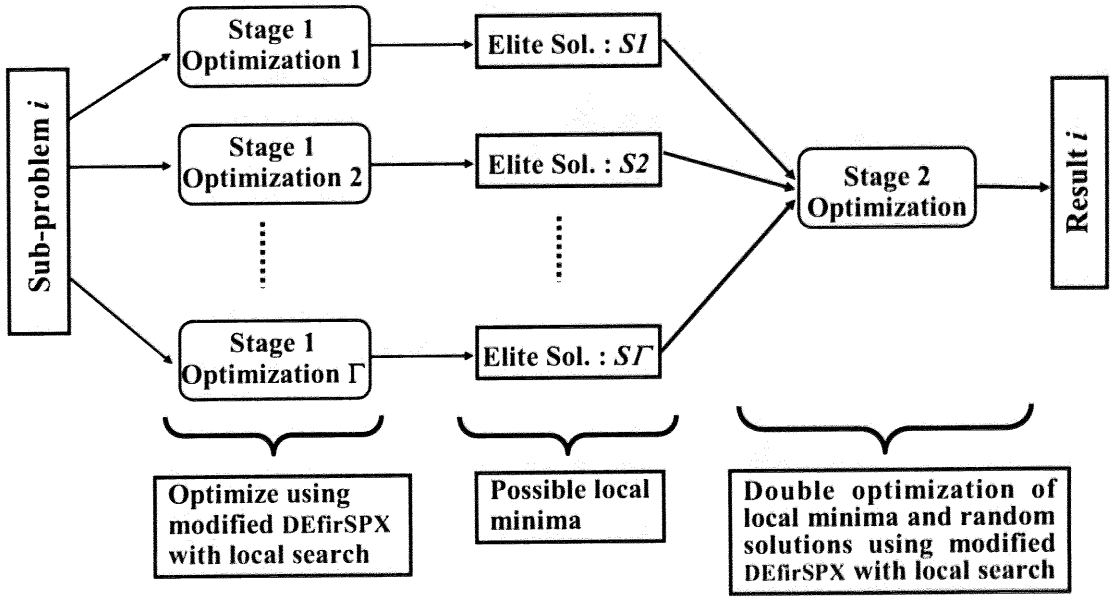


Figure 6.1: Optimization procedure for subproblem $i$

### Mutation Phase

In EAs the mutation operator serves to create random diversity in the population. Traditionally, mutation has always been used as a secondary mechanism in comparison to crossover. Since DE/DEfirSPX does not have a direct mutation operation in this reconstruction algorithm a mutation operation is periodically applied to introduce new traits in the population and to escape local minima thereby. The mutation process works as follows: If the fitness of the elite individual does not improve for $G_m$ generations then all the other individuals in current generation are passed through the mutation phase. Here, the Gaussian mutation was applied with mutation probability $p_m$. Gaussian mutation realizes the mutation operation by adding a random value from the Gaussian distribution. For mutating the *rate constants* of an individual the random numbers are drawn from a Gaussian distribution with mean $\mu_r = 0$

and standard deviation $\sigma_r$ and for mutating the *kinetic orders* the random numbers are drawn from a distribution with mean $\mu_k = 0$ and standard deviation $\sigma_k$.

In each stage of the algorithm the overall optimization procedure for estimating the model parameters for each subproblem can be summarized as follows:

1. Prepare initial population $P_G$ with candidate solutions
2. Create the new generation $P_{G+1}$ of candidate solution applying recombination/selection operation of DEfirSPX
3. Apply local search to the best individual and a randomly selected individual of the new generation $P_{G+1}$
4. If fitness of the elite individual does not improve for $G_m$ generations then apply mutation to non-elite individuals
5. Stop if the termination criteria satisfied. Otherwise Set G=G+1 and go to Step 2

In phase 1 the initial population $P_G$ is created randomly. And $\Gamma$ trails of this phase 1 is repeated. In Phase 2 the elite individuals from different trials of Phase 1 together with some random individuals are used for initialization.

### 6.1.1 Local Search Procedure

In order to provide an effective global optimization method some metaheuristics or local searches are often embedded inside the evolutionary algorithm. Therefore, a local search method was introduced inside the DEfirSPX algorithm. This local refinement procedure performs a greedy search operation around the best individual and a random individual of each generation. The local search around the best individual and a random individual will accelerate the optimization process as well as maintain the diversity of the population. The local searching is performed as follows: All the *kinetic orders* are sorted in ascending order of their absolute values. Then the *kinetic order* of the lowest absolute value is set to zero. And the fitness of this new individual is evaluated. If this modification improves the fitness of the individual then new solution is accepted otherwise it's parent solution is restored. And this process repeated for all the kinetic order in increasing order of their absolute values. This local search process allows to identify the zero valued parameters by mutating them in the increasing order of their strength and thus helps to identify the skeletal network structure. And the restore capability of the greedy search also

allows to recover from wrong elimination of any essential regulation. Hybridizing this greedy local search procedure with the DEfirSPX algorithm we can identify the sparse network structure efficiently and the strength of regulations more accurately.

## 6.2 Reconstruction Experiments and Results

In order to investigate how successfully the proposed method can reconstruct the network topology and estimate the kinetic parameters first it was evaluated by simulation. Three artificial networks of different dimensions and characteristics were used for this purpose. First, these artificial networks were simulated to obtain synthetic microarray data sets and then the reconstruction method was applied for reverse engineering the networks from these data sets. The details of the experiments and the outcomes follow in the subsequent sections.

If an insufficient amount of time series data is used for estimating the parameters for S-system model many candidate solutions will evolve due to the high-degree of freedom of the model [77]. This is because, it is only one path in a phase diagram and from such a single path no general conclusions about the overall behavior of the dynamic system can be drawn [124]. Use of multiple set of observed response under different environmental condition can be helpful in identifying the behavior of a dynamics system. Generally, a gene expression profile consisting of $T \times D$ expression measurements, $T$ in each experiment under $D$ different environmental conditions, will contain more information than an expression profile of $(T \times D)$ data points collected from a single phenomenon [25]. In addition, earlier experiments have shown a single set of gene expression data is not sufficient to reconstruct the network accurately [54, 56, 124] and use of multiple different expression profiles can enhance accuracy of the reconstruction methodology. Therefore, in these simulations multiple set of gene expression data were used for identifying the regulatory interactions in the target genomic networks.

### 6.2.1 Small Scale Network Inference

As a first study, the proposed approach was tested using a well studied small scale network model NET1. The system, consisting of five genes, adequately demonstrates different types of positive and negative mode of regulatory controls among the reactants. The network is shown in Fig. 6.2 and the target S-system parameters for the network are listed in Table 6.1.

In this typical regulatory system the gene interaction takes place centering two

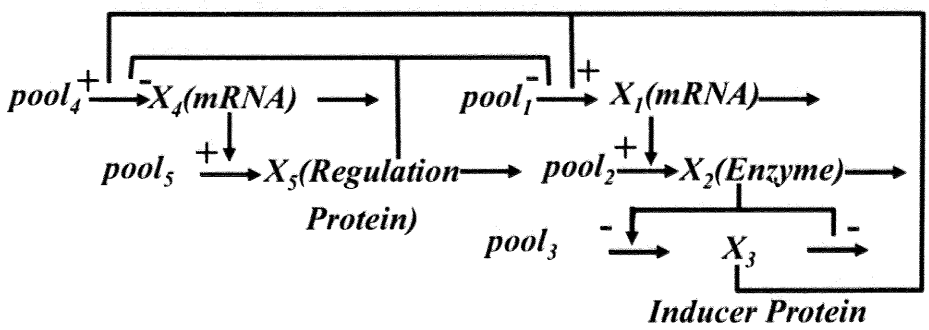


Figure 6.2: Small scale genetic network NET1

Table 6.1: S-system parameters for network model NET1

| $i$ | $\alpha_i$ | $g_{i1}$ | $g_{i2}$ | $g_{i3}$ | $g_{i4}$ | $g_{i5}$ | $\beta_i$ | $h_{i1}$ | $h_{i2}$ | $h_{i3}$ | $h_{i4}$ | $h_{i5}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 5.0 | 0.0 | 0.0 | 1.0 | 0.0 | -1.0 | 10.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2 | 10.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 | 10.0 | 0.0 | 2.0 | 0.0 | 0.0 | 0.0 |
| 3 | 10.0 | 0.0 | -1.0 | 0.0 | 0.0 | 0.0 | 10.0 | 0.0 | -1.0 | 2.0 | 0.0 | 0.0 |
| 4 | 8.0 | 0.0 | 0.0 | 2.0 | 0.0 | -1.0 | 10.0 | 0.0 | 0.0 | 0.0 | 2.0 | 0.0 |
| 5 | 10.0 | 0.0 | 0.0 | 0.0 | 2.0 | 0.0 | 10.0 | 0.0 | 0.0 | 0.0 | 0.0 | 2.0 |

genes (genes 1 and 4). X1 is the mRNA produced from gene 1, X2 is an enzyme protein gene 2 produces, and X3 is an inducer protein catalyzed by X2. X4 is an mRNA produced from gene 4 and X5 is a regulator protein produced by gene 5. Positive feedback from the inducer protein X3 and negative feedback from the regulator protein X5 are assumed in the mRNA production processes of genes 1 and 4. This model has been developed to analyze the interaction of regulator and effector genes. In this study, this model was used as an example that is well-studied and has feedback loops. Using the same set of parameters as found in many other studies [129, 54, 56, 88] the results can be easily compared with early approaches.

As mentioned earlier, many candidate solutions evolve if the model parameters are estimated using insufficient amount of time series data. Therefore, $M = 10$ sets of time series data were used for ensuring sufficient amount of observed gene expression levels. The sets of time-series were obtained by solving (3.5) on the model of Table 6.1. Initial concentration level for each time series was generated randomly in $[0.0, 1.0]$. Sampling 11 points from each time-course $10 \times 11 = 110$ gene expression samples were used for each gene. Then 5% Gaussian noise was added to the time series data in order to simulate the measurement error between the true expression and observed expression.

Table 6.2: Inferred parameters for network model NET1 from 5% noisy data

| $i$ | $\alpha_i$ | $g_{i1}$ | $g_{i2}$ | $g_{i3}$ | $g_{i4}$ | $g_{i5}$ | $\beta_i$ | $h_{i1}$ | $h_{i2}$ | $h_{i3}$ | $h_{i4}$ | $h_{i5}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4.804 | 0.000 | 0.000 | 1.209 | 0.000 | -1.613 | 7.595 | 1.996 | -0.150 | 0.000 | 0.000 | -0.960 |
| 2 | 10.827 | 2.358 | 0.000 | 0.000 | -0.109 | 0.000 | 11.189 | 0.000 | 2.609 | 0.000 | 0.000 | -0.390 |
| 3 | 9.883 | 0.000 | -0.913 | 0.000 | 0.000 | -0.103 | 10.730 | 0.000 | -0.889 | 2.248 | 0.000 | 0.000 |
| 4 | 7.687 | 0.000 | 0.000 | 2.234 | 0.000 | -0.992 | 10.550 | 0.505 | 0.000 | 0.000 | 2.113 | 0.000 |
| 5 | 12.289 | 0.000 | 0.000 | 0.413 | 1.471 | 0.000 | 10.800 | 0.000 | -0.138 | 0.000 | 0.000 | 1.495 |

### Experimental Setup

The experiment was carried out under the following setup. The search regions of the parameters were $[0.0, 20.0]$ for $\alpha_i$ and $\beta_i$, and $[-3.0, 3.0]$ for $g_{ij}$ and $h_{ij}$. The maximum cardinality $I$ was chosen 5, and the penalty coefficient $c$ was 1000.0. The parameter values for memDE algorithm were $F = 0.5$, $CF = 0.8$ and $L = 10$, population size was 60 and the maximum number of generation in each trial of *Phase 1* and in *Phase 2* was 850. In *Phase 1*, 5 ($\Gamma = 1, \cdots, 5$) independent trial solutions were evolved from which elite individuals were selected for optimization in *Phase 2*. The parameter values for the mutation phase were $p_m = 0.01$, $\sigma_r = 3.0$ and $\sigma_k = 1.2$. In *Phase 1* of the optimization, $G_m = 100$ and in *Phase 2*, $G_m = 200$ were used. The algorithm was implemented in Java language and the time required for solving each subproblem was approximately 12 minutes using a PC with 1700 MHz Intel Pentium processor and 512 MB of RAM.

In order to reduce the computational burden a structure skeletalizing was applied in a similar fashion used by Tominaga *et al.* in [129]. If the absolute value of a parameter is less than a threshold value $\delta$ then structure skeletalizing resets it to zero. This process reduces the computational cost as well as helps to identify the nonexistent regulations. In different experiments $\delta = 0.001$ was used. For assuring the soundness of the stochastic search algorithm 10 repetitions for each experiment were performed.

### Result

Table 6.2 shows the parameters estimated by the developed algorithm in a typical run. From the 5% noise corrupted data the method method extracted all the correct regulations and a few false interactions. The number of false-negative interactions was zero. Many of the zero valued parameters were identified correctly but in some cases estimated *kinetic constants* were not very precise and some false predicted parameter values were too large to ignore. Nevertheless, all the core interactions were identified correctly and parameter values were estimated with reasonable accuracy.

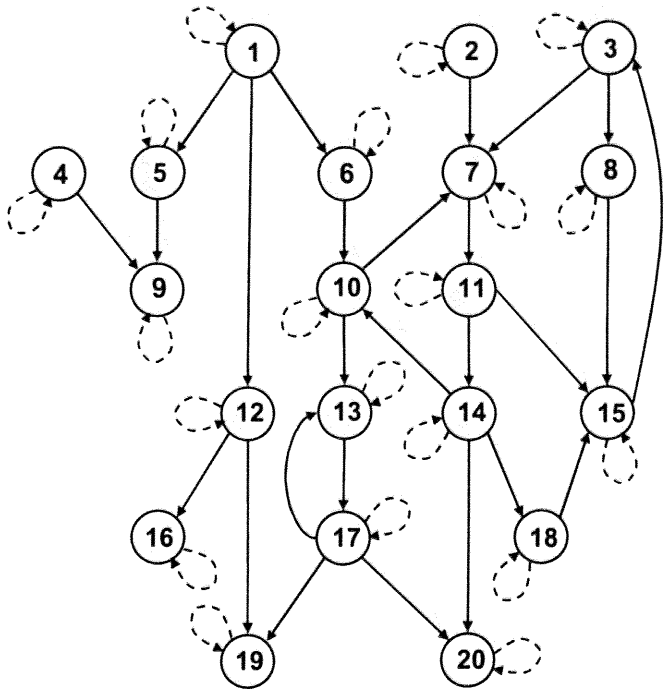


Figure 6.3: Structure of the artificial gene regulatory network NET2. Solid and dashed lines show synthetic and degradative influences, respectively

### 6.2.2 Medium Scale Network Inference

In this experiment, the performance of the proposed algorithm was investigated using a 20 gene network NET2. The topology of the network was created randomly with a maximum indegree limit and then formulated the network in S-system formalism. Fig. 6.3 shows the network structure and Table 6.3 contains the parameters which were chosen arbitrarily. This artificial network models several types of regulations (e.g. auto-regulation, cyclic- regulation, feed-back regulation) commonly found in bio-chemical networks, which makes it a standard simulation model. The network model NET2 was simulated with random initial concentrations chosen from $[0.0, 1.0]$ and 20 sets of synthetic microarray data were generated. From each time course data 11 samples were used for inferring the model parameters. Experiment were performed both in noise free ideal and noisy conditions. This time 10% Gaussian noise was added to the time series to simulate a more noisy environment.

#### Experimental Setup

Since the developed methodology is a stochastic search process its reconstruction capability was verified in ten repeated runs under following conditions. The popu-

Table 6.3: Target S-system parameters for NET2

| | |
|---|---|
| $\alpha_i$ $\beta_i$ | 10.0 |
| $g_{i,j}$ | $g_{3,15} = -0.7$, $g_{5,1} = 1.0$, $g_{6,1} = 2.0$, $g_{7,2} = 1.2$, $g_{7,3} = -0.8$, $g_{7,10} = 1.6$, $g_{8,3} = -0.6$, $g_{9,4} = 0.5$, $g_{9,5} = 0.7$, $g_{10,6} = -0.3$, $g_{10,14} = 0.9$, $g_{11,7} = 0.5$, $g_{12,1} = 1.0$, $g_{13,10} = -0.4$, $g_{13,17} = 1.3$, $g_{14,11} = -0.4$, $g_{15,8} = 0.5$, $g_{15,11} = -1.0$, $g_{15,18} = -0.9$,$g_{16,12} = 2.0$, $g_{17,13} = -0.5$,$g_{18,14} = 1.2$, $g_{19,12} = 1.4$, $g_{19,17} = 0.6$ $g_{20,14} = 1.0$, $g_{20,17} = 1.5$, other $g_{i,j} = 0.0$ |
| $h_{ij}$ | 1.0 if $(i = j)$, 0.0, otherwise |

lation size was 210 and the maximum number of generation in each trial of *Phase 1* and in *Phase 2* was 2400. Other conditions were same as in Sec. 6.2.1. The average time for solving each sub-problem was approximately 15.5 hours using a PC with 1700 MHz Intel Pentium processor and 512 MB of RAM.

### Results

Under noise free condition, the proposed method successfully predicted the exact network architecture and also determined the type of regulation (activation/inhibition) correctly in each of the 10 experimental runs. The process also estimated the kinetic parameters with high accuracy under noise free condition which are presented in Table 6.4. And the typical regulations estimated for NET2 from 10% noise corrupted time series are shown in Table 6.5. In 10% noisy environment the proposed method failed to predict a few regulations in the target network and predicted some false interactions. Some of these false regulations can be ignored because of the strength of the affectivity whereas some are too strong to ignore. From 46 true regulations of the target network, the algorithm inferred $44.8 \pm 0.421$ true-positive, $53.8 \pm 0.421$ false-positive and $1.2 \pm 0.421$ false negative interactions, on an average in 10 experimental runs. The possible reason for such failure is the noise in the experimental data and the accuracy can be increased using additional time-series data.

In order to further investigate the capability of the method in identifying different types of transcriptional regulations in a genetic network, the NET2 was modified adding some degradative regulations and self-regulatory synthetic regulations. This modified network is denoted as NET3 and the parameters are shown in Table 6.6. Then reconstruction experiments were performed under the same conditions as used for NET2 (Section 6.2.2), and the method was successful to identify almost all regulations in the network with pretty accurate parameter values. In 10 repeated trials of experiment, the algorithm predicted $53.6 \pm 0.516$ true-positive, $45.2 \pm 0.788$ false-positive and $1.4 \pm 0.516$ false-negative regulations out of 55 true regulations,

Table 6.4: Inferred parameters for NET2 using proposed fitness function of (9) and noise free data

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Gene 1 | $\alpha_1 = 9.875$ | $g_{1,8} = -0.002$ | $g_{1,11} = 0.002$ | $\beta_1 = 9.856$ | $h_{1,1} = 1.017$ | | |
| Gene 2 | $\alpha_2 = 10.252$ | $g_{2,2} = 0.004$ | $g_{2,3} = -0.004$ | $g_{2,18} = 0.002$ | $\beta_2 = 10.279$ | $h_{2,2} = 0.994$ | $h_{2,8} = -0.015$ |
| Gene 3 | $\alpha_3 = 9.546$ | $g_{3,3} = -0.042$ | $g_{3,14} = -0.010$ | $g_{3,15} = -0.700$ | $\beta_3 = 9.478$ | $h_{3,3} = 1.019$ | $h_{3,15} = 0.036$ |
| Gene 4 | $\alpha_4 = 10.257$ | $g_{4,13} = 0.004$ | $\beta_4 = 10.282$ | $h_{4,4} = 0.973$ | $h_{4,16} = 0.001$ | | |
| Gene 5 | $\alpha_5 = 9.586$ | $g_{5,1} = 1.009$ | $g_{5,13} = -0.027$ | $g_{5,18} = -0.010$ | $\beta_5 = 9.509$ | $h_{5,1} = -0.031$ | $h_{5,5} = 1.028$ |
| Gene 6 | $\alpha_6 = 9.809$ | $g_{6,1} = 2.045$ | $g_{6,7} = 0.002$ | $g_{6,13} = -0.008$ | $\beta_6 = 9.845$ | $h_{6,6} = 1.020$ | $h_{6,10} = 0.003$ |
| Gene 7 | $\alpha_7 = 10.094$ | $g_{7,2} = 1.254$ | $g_{7,3} = -0.798$ | $g_{7,10} = 1.528$ | $\beta_7 = 10.156$ | $h_{7,1} = -0.011$ | $h_{7,7} = 0.986$ |
| Gene 8 | $\alpha_8 = 10.079$ | $g_{8,3} = -0.582$ | $\beta_8 = 10.088$ | $h_{8,3} = 0.013$ | $h_{8,8} = 0.990$ | | |
| Gene 9 | $\alpha_9 = 9.311$ | $g_{9,4} = 0.553$ | $g_{9,5} = 0.733$ | $g_{9,9} = -0.052$ | $\beta_9 = 9.380$ | $h_{9,1} = -0.021$ | $h_{9,9} = 1.049$ |
| Gene 10 | $\alpha_{10} = 9.352$ | $g_{10,6} = -0.320$ | $g_{10,10} = -0.008$ | $g_{10,14} = 0.923$ | $\beta_{10} = 9.278$ | $h_{10,1} = -0.012$ | $h_{10,10} = 1.079$ |
| Gene 11 | $\alpha_{11} = 11.682$ | $g_{11,3} = 0.016$ | $g_{11,7} = 0.447$ | $\beta_{11} = 11.761$ | $h_{11,7} = 0.024$ | $h_{11,11} = 0.861$ | $h_{11,17} = -0.011$ |
| Gene 12 | $\alpha_{12} = 9.986$ | $g_{12,1} = 0.995$ | $g_{12,4} = -0.004$ | $g_{12,6} = 0.006$ | $\beta_{12} = 9.969$ | $h_{12,12} = 1.007$ | $h_{12,13} = -0.002$ |
| Gene 13 | $\alpha_{13} = 8.356$ | $g_{13,10} = -0.466$ | $g_{13,13} = -0.037$ | $g_{13,17} = 1.432$ | $\beta_{13} = 8.325$ | $h_{13,13} = 1.125$ | $h_{13,17} = -0.049$ |
| Gene 14 | $\alpha_{14} = 9.799$ | $g_{14,9} = -0.004$ | $g_{14,11} = -0.408$ | $g_{14,14} = 0.009$ | $\beta_{14} = 9.785$ | $h_{14,5} = -0.004$ | $h_{14,14} = 1.051$ |
| Gene 15 | $\alpha_{15} = 11.022$ | $g_{15,8} = 0.380$ | $g_{15,11} = -0.922$ | $g_{15,18} = -0.894$ | $\beta_{15} = 11.294$ | $h_{15,8} = -0.156$ | $h_{15,15} = 0.913$ |
| Gene 16 | $\alpha_{16} = 8.199$ | $g_{16,4} = -0.015$ | $g_{16,12} = 2.229$ | $g_{16,16} = -0.066$ | $\beta_{16} = 8.177$ | $h_{16,12} = -0.015$ | $h_{16,12} = 1.127$ |
| Gene 17 | $\alpha_{17} = 10.305$ | $g_{17,5} = -0.005$ | $g_{17,13} = -0.488$ | $\beta_{17} = 10.341$ | $h_{17,11} = 0.009$ | $h_{17,13} = 0.006$ | $h_{17,17} = 0.944$ |
| Gene 18 | $\alpha_{18} = 8.735$ | $g_{18,14} = 1.265$ | $g_{18,18} = -0.056$ | $\beta_{18} = 8.701$ | $h_{18,14} = -0.084$ | $h_{18,15} = 0.005$ | $h_{18,18} = 1.078$ |
| Gene 19 | $\alpha_{19} = 10.708$ | $g_{19,12} = 1.335$ | $g_{19,17} = 0.572$ | $\beta_{19} = 10.726$ | $h_{19,5} = 0.018$ | $h_{19,12} = 0.045$ | $h_{19,19} = 0.904$ |
| Gene 20 | $\alpha_{20} = 7.883$ | $g_{20,14} = 1.024$ | $g_{20,17} = 1.770$ | $g_{20,20} = -0.136$ | $\beta_{20} = 7.814$ | $h_{20,14} = -0.199$ | $h_{20,20} = 1.068$ |
| | | | | | | | other $g_{i,j} = 0$ and $h_{i,j} = 0$ |

Table 6.5: Inferred parameters for NET2 using proposed fitness function of (9) and 10% noisy data

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Gene 1 | $\alpha_1 = 8.726$ | $g_{1,10} = -0.110$ | $\beta_1 = 8.676$ | $h_{1,1} = 1.132$ | $h_{1,14} = -0.144$ | $h_{1,17} = -0.163$ | |
| Gene 2 | $\alpha_2 = 9.143$ | $g_{2,10} = 0.081$ | $g_{2,16} = -0.049$ | $g_{2,18} = -0.095$ | $\beta_2 = 9.044$ | $h_{2,2} = 1.007$ | |
| Gene 3 | $\alpha_3 = 10.255$ | $g_{3,9} = -0.084$ | $g_{3,15} = -0.708$ | $\beta_3 = 10.842$ | $h_{3,1} = 0.095$ | $h_{3,3} = 1.112$ | $h_{3,5} = -0.086$ |
| Gene 4 | $\alpha_4 = 9.181$ | $g_{4,17} = 0.090$ | $\beta_4 = 8.239$ | $h_{4,1} = -0.148$ | $h_{4,3} = -0.110$ | $h_{4,4} = 1.443$ | $h_{4,11} = -0.102$ |
| Gene 5 | $\alpha_5 = 7.968$ | $g_{5,1} = 0.958$ | $\beta_5 = 7.513$ | $h_{5,1} = -0.384$ | $h_{5,5} = 1.260$ | $h_{5,11} = 0.203$ | |
| Gene 6 | $\alpha_6 = 7.271$ | $g_{6,1} = 2.177$ | $\beta_6 = 6.005$ | $h_{6,1} = -1.039$ | $h_{6,6} = 2.078$ | $h_{6,7} = -0.202$ | $h_{6,8} = 0.330$ |
| Gene 7 | $\alpha_7 = 9.303$ | $g_{7,2} = 1.124$ | $g_{7,3} = -0.843$ | $g_{7,10} = 1.370$ | $\beta_7 = 9.406$ | $h_{7,7} = 0.933$ | $h_{7,17} = -0.131$ |
| Gene 8 | $\alpha_8 = 8.880$ | $g_{8,3} = -0.593$ | $g_{8,20} = -0.054$ | $\beta_8 = 8.773$ | $h_{8,2} = -0.289$ | $h_{8,6} = 0.130$ | $h_{8,8} = 1.419$ |
| Gene 9 | $\alpha_9 = 8.382$ | $g_{9,3} = 0.212$ | $g_{9,5} = 0.849$ | $g_{9,9} = 0.173$ | $\beta_9 = 8.650$ | $h_{9,4} = -0.387$ | $h_{9,9} = 1.469$ |
| Gene 10 | $\alpha_{10} = 9.334$ | $g_{10,6} = -0.352$ | $g_{10,14} = 1.058$ | $\beta_{10} = 9.017$ | $h_{10,1} = -0.177$ | $h_{10,2} = 0.133$ | $h_{10,10} = 1.280$ |
| Gene 11 | $\alpha_{11} = 7.906$ | $g_{11,7} = 1.153$ | $g_{11,20} = 0.582$ | $\beta_{11} = 5.674$ | $h_{11,1} = -0.221$ | $h_{11,11} = 2.630$ | $h_{11,15} = 0.117$ |
| Gene 12 | $\alpha_{12} = 8.132$ | $g_{12,1} = 0.968$ | $\beta_{12} = 7.944$ | $h_{12,1} = -0.146$ | $h_{12,2} = 0.057$ | $h_{12,12} = 1.058$ | $h_{12,15} = -0.054$ |
| Gene 13 | $\alpha_{13} = 10.872$ | $g_{13,5} = -0.064$ | $g_{13,10} = -0.274$ | $g_{13,17} = 0.686$ | $\beta_{13} = 10.909$ | $h_{13,13} = 0.832$ | $h_{13,17} = -0.226$ |
| Gene 14 | $\alpha_{14} = 8.516$ | $g_{14,9} = 0.060$ | $g_{14,11} = -0.444$ | $\beta_{14} = 8.457$ | $h_{14,4} = 0.190$ | $h_{14,14} = 1.363$ | $h_{14,18} = 0.115$ |
| Gene 15 | $\alpha_{15} = 9.659$ | $g_{15,8} = 0.558$ | $g_{15,11} = -0.914$ | $g_{15,18} = -0.953$ | $\beta_{15} = 9.903$ | $h_{15,11} = 0.286$ | $h_{15,15} = 1.071$ |
| Gene 16 | $\alpha_{16} = 8.663$ | $g_{16,12} = 1.963$ | $\beta_{16} = 7.125$ | $h_{16,1} = -0.242$ | $h_{16,11} = -0.301$ | $h_{16,14} = -0.376$ | $h_{16,16} = 1.227$ |
| Gene 17 | $\alpha_{17} = 10.917$ | $g_{17,13} = -0.456$ | $\beta_{17} = 10.900$ | $h_{17,1} = 0.406$ | $h_{17,6} = -0.199$ | $h_{17,17} = 0.648$ | |
| Gene 18 | $\alpha_{18} = 7.851$ | $g_{18,4} = -0.107$ | $g_{18,14} = 1.472$ | $\beta_{18} = 7.515$ | $h_{18,4} = -0.307$ | $h_{18,12} = -0.180$ | $h_{18,18} = 1.407$ |
| Gene 19 | $\alpha_{19} = 8.929$ | $g_{19,8} = -0.070$ | $g_{19,12} = 1.476$ | $g_{19,17} = 0.693$ | $\beta_{19} = 8.793$ | $h_{19,16} = -0.261$ | $h_{19,19} = 1.310$ |
| Gene 20 | $\alpha_{20} = 10.144$ | $g_{20,2} = 0.274$ | $g_{20,14} = 0.880$ | $g_{20,17} = 1.203$ | $g_{20,18} = -0.065$ | $\beta_{20} = 10.583$ | $h_{20,20} = 0.976$ |
| | | | | | | | other $g_{i,j} = 0$ and $h_{i,j} = 0$ |

Table 6.6: Target S-system parameters for network model NET3

| | |
|---|---|
| $\alpha_i$, $\beta_i$ | 10.0 |
| $g_{i,j}$ | $g_{5,5} = -0.8$, $g_{19,19} = 0.5$ other $g_{i,j}$ same as in NET2 |
| $h_{i,j}$ | $h_{2,3} = 1.0$, $h_{5,4} = 1.0$, $h_{8,2} = 1.0$, $h_{11,10} = 1.0$, $h_{12,9} = 1.0$, $h_{17,14} = 1.0$,$h_{20,19} = 1.0$, other $h_{i,j}$ same as in NET2 |

which shows the competence of the algorithm once again. Typical estimations for the kinetic parameters from 10% noisy time series are shown in Table 6.7.

## 6.3 Effect of Noise Level in Gene Expression Data

The task of reconstructing a genetic network from the time-series data is made more complicated by the noise that is incurred during measuring the mRNA levels in a microarray experiment. If the mRNA levels could be measured precisely then the system could be estimated with reasonable accuracy by learning the model parameters using some regression method. The existence of the significant level of noise in the expression profile together with the flexibility of the S-system model makes the learning task most difficult for an inference algorithm.

In order to study the effect of noise-level, present in gene expression data, on the proposed fitness evaluation criteria and on the reverse engineering algorithm, the reconstruction experiments of this section were performed. Since microarray experiment can incur a wide rage of noise levels depending on the technology, environment and the study, experiments were performed under various noise-intensities present in the data. To mimic the real noisy environment $\eta$% Gaussian noise were added to the time-series data for simulating the measurement error. Reconstructions were performed from expression profiles with noise level $\eta = 0$% (noise free ideal condition), $\eta = 5$%, $\eta = 10$% and $\eta = 25$% for investigating the effect of noise on the reconstruction method. For validating the superiority of the proposed fitness functions over the conventional evaluation criteria, the same set of reconstruction experiments were performed using the same algorithm, same setup but different fitness evaluation functions. The performance comparison was done using the fitness function of (3.17), (3.13) and the following

$$f_i = -2\Lambda_i + c \sum_{j=1}^{2N-I} (|K_{ij}|) \tag{6.1}$$

This fitness function of (6.1) was designed by removing the original AIC *penalty*

Table 6.7: Inferred parameters for NET3 from 10% noisy time sereis

| | | | | | | |
|---|---|---|---|---|---|---|
| Gene 1 | $\alpha_1 = 7.743$ | $g_{1,1} = -0.168$ | $\beta_1 = 7.128$ | $h_{1,1} = 2.029$ | $h_{1,3} = -0.163$ | $h_{1,4} = 0.116$ | $h_{1,10} = -0.434$ |
| Gene 2 | $\alpha_2 = 8.739$ | $g_{2,2} = -0.105$ | $g_{2,14} = -0.163$ | $\beta_2 = 8.136$ | $h_{2,2} = 1.393$ | $h_{2,3} = 1.379$ | $h_{2,4} = -0.185$ |
| Gene 3 | $\alpha_3 = 10.876$ | $g_{3,4} = -0.062$ | $g_{3,15} = -0.688$ | $g_{3,18} = -0.068$ | $\beta_3 = 11.047$ | $h_{3,3} = 0.817$ | $h_{3,17} = -0.131$ |
| Gene 4 | $\alpha_4 = 10.548$ | $g_{4,3} = 0.099$ | $\beta_4 = 10.502$ | $h_{4,1} = -0.150$ | $h_{4,4} = 0.942$ | $h_{4,15} = 0.081$ | |
| Gene 5 | $\alpha_5 = 11.140$ | $g_{5,1} = 0.789$ | $g_{5,5} = -0.608$ | $\beta_5 = 11.071$ | $h_{5,4} = 0.708$ | $h_{5,5} = 0.959$ | $h_{5,15} = -0.184$ |
| Gene 6 | $\alpha_6 = 9.709$ | $g_{6,1} = 1.989$ | $\beta_6 = 10.959$ | $h_{6,3} = 0.175$ | $h_{6,6} = 1.067$ | $h_{6,11} = -0.160$ | $h_{6,17} = 0.137$ |
| Gene 7 | $\alpha_7 = 9.642$ | $g_{7,2} = 1.195$ | $g_{7,3} = -0.943$ | $g_{7,10} = 1.775$ | $g_{7,13} = 0.153$ | $\beta_7 = 9.923$ | $h_{7,7} = 0.994$ |
| Gene 8 | $\alpha_8 = 6.254$ | $g_{8,2} = -0.299$ | $g_{8,3} = -0.868$ | $\beta_8 = 6.059$ | $h_{8,2} = 0.904$ | $h_{8,8} = 2.374$ | $h_{8,10} = 0.382$ |
| Gene 9 | $\alpha_9 = 9.148$ | $g_{9,4} = 0.658$ | $g_{9,5} = 0.832$ | $\beta_9 = 9.526$ | $h_{9,2} = 0.133$ | $h_{9,9} = 1.153$ | $h_{9,20} = -0.085$ |
| Gene 10 | $\alpha_{10} = 8.410$ | $g_{10,6} = -0.365$ | $g_{10,14} = 0.976$ | $\beta_{10} = 8.305$ | $h_{10,7} = -0.093$ | $h_{10,10} = 1.370$ | $h_{10,15} = -0.166$ |
| Gene 11 | $\alpha_{11} = 7.850$ | $g_{11,7} = 0.609$ | $g_{11,10} = -0.151$ | $\beta_{11} = 7.405$ | $h_{11,10} = 1.219$ | $h_{11,11} = 1.798$ | $h_{11,20} = -0.252$ |
| Gene 12 | $\alpha_{12} = 9.889$ | $g_{12,1} = 1.040$ | $g_{12,16} = 0.124$ | $\beta_{12} = 10.047$ | $h_{12,5} = 0.601$ | $h_{12,9} = 0.931$ | $h_{12,12} = 1.221$ |
| Gene 13 | $\alpha_{13} = 8.049$ | $g_{13,8} = 0.367$ | $g_{13,10} = -0.402$ | $g_{13,17} = 1.546$ | $\beta_{13} = 7.235$ | $h_{13,4} = -0.232$ | $h_{13,13} = 1.307$ |
| Gene 14 | $\alpha_{14} = 8.010$ | $g_{14,11} = -0.397$ | $g_{14,18} = -0.109$ | $\beta_{14} = 9.385$ | $h_{14,1} = 0.200$ | $h_{14,8} = 0.236$ | $h_{14,14} = 1.492$ |
| Gene 15 | $\alpha_{15} = 9.035$ | $g_{15,8} = 0.518$ | $g_{15,11} = -1.001$ | $g_{15,18} = -0.979$ | $\beta_{15} = 8.734$ | $h_{15,15} = 1.101$ | $h_{15,20} = -0.075$ |
| Gene 16 | $\alpha_{16} = 8.397$ | $g_{16,1} = 0.284$ | $g_{16,12} = 2.316$ | $g_{16,16} = -0.365$ | $\beta_{16} = 7.621$ | $h_{16,16} = 0.948$ | $h_{16,20} = -0.138$ |
| Gene 17 | $\alpha_{17} = 10.101$ | $g_{17,13} = -0.507$ | $g_{17,15} = 0.079$ | $\beta_{17} = 9.419$ | $h_{17,5} = -0.193$ | $h_{17,14} = 1.095$ | $h_{17,17} = 0.844$ |
| Gene 18 | $\alpha_{18} = 8.053$ | $g_{18,14} = 1.122$ | $\beta_{18} = 7.968$ | $h_{18,12} = -0.150$ | $h_{18,14} = -0.150$ | $h_{18,15} = -0.145$ | $h_{18,18} = 1.389$ |
| Gene 19 | $\alpha_{19} = 6.164$ | $g_{19,12} = 1.726$ | $g_{19,17} = 0.724$ | $\beta_{19} = 5.268$ | $h_{19,3} = -0.296$ | $h_{19,10} = -0.476$ | $h_{19,19} = 0.649$ |
| Gene 20 | $\alpha_{20} = 9.233$ | $g_{20,6} = -0.130$ | $g_{20,14} = 1.581$ | $g_{20,17} = 1.773$ | $\beta_{20} = 9.709$ | $h_{20,19} = 0.994$ | $h_{20,20} = 1.484$ |
| | | | | | | | other $g_{i,j} = 0$ and $h_{i,j} = 0$ |

Table 6.8: Inferred parameters for NET1 using proposed fitness function of (3.18)

| | $i$ | $\alpha_i$ | $g_{i1}$ | $g_{i2}$ | $g_{i3}$ | $g_{i4}$ | $g_{i5}$ | $\beta_i$ | $h_{i1}$ | $h_{i2}$ | $h_{i3}$ | $h_{i4}$ | $h_{i5}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NoiseFree | 1 | 5.006 | 0.000 | -0.008 | 0.978 | -0.004 | -0.996 | 10.010 | 1.975 | 0.000 | 0.000 | 0.000 | 0.000 |
| | 2 | 10.064 | 1.994 | 0.005 | 0.008 | 0.002 | -0.001 | 10.070 | 0.000 | 1.996 | 0.010 | 0.000 | 0.004 |
| | 3 | 9.942 | 0.000 | -1.001 | -0.002 | 0.000 | -0.002 | 9.944 | 0.001 | -1.001 | 2.008 | 0.000 | -0.001 |
| | 4 | 7.970 | 0.000 | -0.017 | 1.946 | 0.000 | -0.996 | 10.007 | -0.029 | 0.000 | 0.000 | 1.990 | 0.000 |
| | 5 | 10.012 | 0.000 | 0.003 | 0.023 | 2.002 | -0.009 | 9.993 | 0.005 | 0.000 | 0.003 | 0.000 | 1.991 |
| 5% Noisy | 1 | 4.804 | 0.000 | 0.000 | 1.209 | 0.000 | -1.613 | 7.595 | 1.996 | -0.150 | 0.000 | 0.000 | -0.960 |
| | 2 | 10.827 | 2.358 | 0.000 | 0.000 | -0.109 | 0.000 | 11.189 | 0.000 | 2.609 | 0.000 | 0.000 | -0.390 |
| | 3 | 9.883 | 0.000 | -0.913 | 0.000 | 0.000 | -0.103 | 10.730 | 0.000 | -0.889 | 2.248 | 0.000 | 0.000 |
| | 4 | 7.687 | 0.000 | 0.000 | 2.234 | 0.000 | -0.992 | 10.550 | 0.505 | 0.000 | 0.000 | 2.113 | 0.000 |
| | 5 | 12.289 | 0.000 | 0.000 | 0.413 | 1.471 | 0.000 | 10.800 | 0.000 | -0.138 | 0.000 | 0.000 | 1.495 |
| 10% Noisy | 1 | 3.081 | 0.000 | 0.000 | 0.000 | 0.000 | -1.112 | 7.997 | 3.000 | 0.000 | -1.615 | 0.426 | 0.000 |
| | 2 | 10.590 | 1.182 | 0.327 | 0.000 | 0.207 | 0.000 | 18.390 | 0.000 | 3.000 | 0.000 | 0.000 | 0.661 |
| | 3 | 11.605 | 0.000 | -0.755 | 0.000 | 0.000 | 0.000 | 12.929 | -0.328 | -0.509 | 2.248 | 0.000 | 0.000 |
| | 4 | 7.753 | 0.000 | 0.000 | 1.384 | 0.000 | -0.993 | 12.507 | 0.000 | 0.000 | 0.000 | 3.000 | 0.000 |
| | 5 | 7.782 | 0.000 | 0.000 | 0.000 | 1.387 | 0.000 | 7.427 | 0.000 | 0.000 | 0.000 | -0.458 | 2.319 |
| 25% Noisy | 1 | 10.992 | 0.000 | 0.000 | 0.000 | 0.000 | -0.566 | 20.000 | 1.631 | 0.000 | 0.000 | 0.000 | 0.000 |
| | 2 | 20.000 | 3.000 | 0.672 | 0.000 | 0.000 | 0.000 | 19.720 | 0.000 | 2.507 | -0.966 | 0.965 | 0.000 |
| | 3 | 15.283 | 0.000 | -0.783 | 0.000 | 0.000 | 0.000 | 17.969 | 0.773 | -0.632 | 3.000 | 0.000 | 0.000 |
| | 4 | 5.618 | 0.000 | 0.000 | 1.409 | 0.000 | -0.990 | 5.146 | 1.779 | -0.737 | 0.000 | 2.099 | 0.000 |
| | 5 | 16.109 | 0.000 | 0.393 | 0.000 | 1.223 | 0.000 | 18.456 | 0.000 | 0.000 | 0.000 | 0.000 | 3.177 |

*term* from the proposed fitness function of (3.18) to investigate whether the proposed penalty term alone is sufficient to identify the skeletal network structure. Other setup of the environment was same as in section 6.2.1 and 10 repetitions of each experiments were performed. The network parameter values estimated in typical experimental runs with different noise patterns and using the fitness functions of (3.18) , (3.17), (3.13) and (6.1) are reported in Table 6.8, 6.9, 6.10 and 6.11 respectively.

The fact, that the performance of a reverse-engineering algorithm is influenced by the noise-level in the system dynamics, was relearnt from the experiments with the proposed fitness function of (3.18) (Table 6.8). At first, in the absence of the noise, the exact network topology was identified and the estimated parameters were almost accurate. But as the level of noise increases the performance of the method starts to degrade both in terms of topology inference and parameter estimation. For example, when the noise level was 5% all the regulatory interactions in the network were identified correctly and the estimated parameter values were very close the target values. But when the noise level was 10% or 25% the proposed algorithm was not good enough for identifying the exact network architecture. And the accuracy of the inferred parameter values also degraded with the increase of noise.

The similar scenario had been observed when inference was done using the fitness function of (3.17), (3.13) and (6.1) (as shown in Table 6.9, 6.10 and 6.11 respectively). Still, comparing the results of the experiments with the same noise level, it was found that the network topology and the parameter values were more accurately

Table 6.9: Inferred parameters for NET1 using MSE based fitness function of (3.17)

| | $i$ | $\alpha_i$ | $g_{i1}$ | $g_{i2}$ | $g_{i3}$ | $g_{i4}$ | $g_{i5}$ | $\beta_i$ | $h_{i1}$ | $h_{i2}$ | $h_{i3}$ | $h_{i4}$ | $h_{i5}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NoiseFree | 1 | 4.631 | -0.094 | 0.000 | 1.035 | 0.000 | -1.042 | 9.522 | 1.978 | 0.000 | 0.000 | 0.000 | 0.003 |
| | 2 | 10.207 | 1.989 | 0.013 | 0.000 | 0.000 | 0.000 | 10.154 | 0.030 | 1.963 | 0.000 | -0.007 | 0.000 |
| | 3 | 9.685 | 0.000 | -1.003 | -0.017 | 0.000 | 0.000 | 9.686 | 0.000 | -1.003 | 2.058 | 0.000 | 0.000 |
| | 4 | 8.136 | 0.000 | -0.017 | 1.927 | 0.013 | -0.988 | 10.269 | 0.000 | 0.000 | 0.000 | 1.961 | 0.000 |
| | 5 | 9.728 | 0.000 | 0.000 | 0.000 | 2.041 | -0.027 | 9.710 | -0.005 | 0.000 | 0.000 | -0.021 | 2.029 |
| 5% Noisy | 1 | 4.816 | 0.000 | 0.105 | 1.420 | 0.000 | -1.392 | 9.900 | 2.640 | 0.000 | 0.000 | 0.000 | -0.445 |
| | 2 | 12.092 | 2.335 | 0.000 | 0.396 | -0.145 | 0.161 | 12.777 | 0.000 | 2.188 | 0.000 | 0.000 | 0.000 |
| | 3 | 6.418 | 0.000 | -0.890 | -0.372 | 0.000 | -0.172 | 7.493 | 0.000 | -0.840 | 3.000 | 0.000 | 0.000 |
| | 4 | 7.071 | 0.000 | 0.000 | 2.370 | -0.086 | -1.027 | 10.209 | 0.743 | 0.000 | 0.000 | 1.987 | 0.000 |
| | 5 | 14.404 | 0.000 | 0.000 | 0.422 | 1.254 | 0.000 | 12.651 | 0.410 | -0.170 | 0.000 | 0.000 | 1.030 |
| 10% Noisy | 1 | 2.669 | 0.695 | 0.000 | 0.000 | 0.000 | -2.310 | 5.045 | 3.000 | 0.000 | -1.406 | 0.000 | -1.678 |
| | 2 | 7.853 | 1.511 | 0.000 | 0.000 | 0.138 | 0.192 | 20.000 | 0.000 | 3.005 | 0.000 | 0.000 | 1.990 |
| | 3 | 9.284 | 0.457 | -0.793 | -0.250 | 0.000 | 0.000 | 11.854 | 0.000 | -0.434 | 3.000 | 0.000 | 0.000 |
| | 4 | 6.894 | 0.000 | 0.000 | 2.028 | -0.123 | -1.116 | 9.025 | 0.000 | -0.347 | 0.000 | 3.000 | 0.000 |
| | 5 | 7.712 | 0.000 | 0.000 | 0.456 | 1.636 | 0.000 | 7.683 | 0.000 | -0.246 | 0.000 | -0.403 | 3.000 |
| 25% Noisy | 1 | 0.198 | -2.435 | 0.000 | 0.000 | -1.548 | 0.000 | 15.992 | 3.000 | 0.000 | -1.668 | 0.000 | 2.701 |
| | 2 | 19.108 | 2.470 | 0.548 | 0.000 | 0.000 | 0.000 | 15.340 | 0.000 | 1.443 | -0.914 | 0.397 | 0.000 |
| | 3 | 13.971 | 0.000 | -0.807 | 0.000 | 0.000 | 0.000 | 20.000 | 1.252 | -0.711 | 3.000 | -0.416 | 0.000 |
| | 4 | 2.335 | -0.016 | 0.202 | 2.939 | -1.127 | -0.889 | 5.062 | 0.076 | 2.913 | 0.024 | -0.837 | -0.005 |
| | 5 | 18.096 | 1.101 | 0.642 | 0.000 | 2.337 | -0.297 | 20.000 | 0.000 | 0.000 | 0.000 | 0.000 | 3.096 |

Table 6.10: Inferred parameters for NET1 using AIC of (3.13)

| | $i$ | $\alpha_i$ | $g_{i1}$ | $g_{i2}$ | $g_{i3}$ | $g_{i4}$ | $g_{i5}$ | $\beta_i$ | $h_{i1}$ | $h_{i2}$ | $h_{i3}$ | $h_{i4}$ | $h_{i5}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NoiseFree | 1 | 2.428 | -0.563 | -0.013 | 1.295 | 0.015 | -1.302 | 7.742 | 2.745 | 0.000 | -0.299 | 0.016 | 0.366 |
| | 2 | 9.429 | 2.018 | -0.029 | 0.013 | 0.005 | -0.010 | 9.430 | -0.132 | 2.120 | 0.016 | 0.004 | -0.011 |
| | 3 | 7.167 | -0.038 | -0.988 | -0.227 | -0.008 | -0.009 | 7.187 | -0.047 | -0.983 | 2.668 | -0.009 | -0.012 |
| | 4 | 6.474 | -0.023 | -0.016 | 2.106 | -0.119 | -1.059 | 8.599 | -0.102 | 0.000 | -0.339 | 2.438 | 0.205 |
| | 5 | 9.452 | -0.017 | 0.000 | 0.030 | 2.067 | -0.057 | 9.432 | -0.016 | 0.002 | 0.002 | -0.042 | 2.043 |
| 5% Noisy | 1 | 2.495 | 0.000 | -0.240 | 1.479 | 0.196 | -2.100 | 4.377 | 2.707 | -0.461 | 0.000 | 0.000 | -1.249 |
| | 2 | 8.331 | 2.809 | -0.307 | 0.304 | -0.200 | 0.186 | 9.514 | 0.000 | 3.000 | -0.882 | 0.000 | 0.000 |
| | 3 | 7.099 | 0.000 | -0.979 | 0.000 | -0.232 | -0.084 | 7.460 | 0.312 | -1.037 | 2.737 | -0.347 | 0.000 |
| | 4 | 5.876 | 0.847 | -0.420 | 2.370 | -0.178 | -0.842 | 6.986 | 1.219 | -0.674 | 0.000 | 3.000 | 0.344 |
| | 5 | 5.752 | -0.608 | 0.189 | 1.071 | 2.111 | -0.866 | 3.921 | 0.000 | 0.000 | -0.453 | -0.698 | 1.304 |
| 10% Noisy | 1 | 5.829 | 0.000 | 0.417 | 0.000 | 0.000 | -0.978 | 15.788 | 2.168 | 0.753 | -1.548 | 0.614 | 0.000 |
| | 2 | 7.312 | 1.287 | 0.000 | -0.817 | 0.449 | 0.000 | 19.144 | 0.000 | 3.000 | -1.916 | 0.788 | 1.646 |
| | 3 | 16.161 | 1.235 | -0.797 | 0.000 | 0.000 | 0.000 | 20.000 | 1.193 | -0.506 | 2.439 | 0.000 | 0.000 |
| | 4 | 6.775 | 1.232 | -0.750 | 1.020 | 0.000 | -0.629 | 6.044 | 0.716 | -1.141 | 0.000 | 3.105 | 0.000 |
| | 5 | 3.722 | -1.083 | 0.000 | 2.155 | 2.517 | -0.995 | 2.734 | -1.645 | 0.000 | 1.165 | -1.158 | 3.000 |
| 25% Noisy | 1 | 10.992 | 0.000 | 0.000 | 0.000 | 0.000 | -0.566 | 20.000 | 1.631 | 0.000 | 0.000 | 0.000 | 0.000 |
| | 2 | 16.451 | 3.000 | 0.547 | 0.000 | 0.000 | 0.000 | 20.000 | 0.000 | 2.407 | -1.714 | 1.130 | 0.905 |
| | 3 | 17.869 | 0.000 | -0.724 | 0.000 | 0.000 | 0.000 | 18.786 | 0.806 | -0.588 | 3.000 | -0.352 | 0.000 |
| | 4 | 4.593 | 0.000 | -0.764 | 0.791 | 0.000 | -0.579 | 2.170 | 0.000 | -1.562 | 0.000 | 1.255 | 0.000 |
| | 5 | 16.138 | 0.000 | 0.394 | 0.000 | 1.222 | 0.000 | 18.455 | 0.000 | 0.000 | 0.000 | 0.000 | 3.186 |

Table 6.11: Inferred parameters for NET1 using fitness function of (6.1)

| | $i$ | $\alpha_i$ | $g_{i1}$ | $g_{i2}$ | $g_{i3}$ | $g_{i4}$ | $g_{i5}$ | $\beta_i$ | $h_{i1}$ | $h_{i2}$ | $h_{i3}$ | $h_{i4}$ | $h_{i5}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NoiseFree | 1 | 5.013 | -0.002 | -0.008 | 0.975 | -0.004 | -0.995 | 10.006 | 1.968 | 0.000 | 0.000 | 0.000 | 0.000 |
| | 2 | 9.987 | 1.997 | 0.001 | 0.009 | 0.003 | -0.002 | 10.013 | -0.015 | 2.013 | 0.011 | 0.003 | 0.004 |
| | 3 | 9.672 | 0.000 | -1.000 | -0.020 | 0.000 | -0.002 | 9.676 | 0.002 | -1.001 | 2.053 | 0.000 | -0.002 |
| | 4 | 7.966 | 0.000 | -0.017 | 1.947 | 0.000 | -0.996 | 10.003 | -0.029 | 0.000 | 0.000 | 1.992 | 0.000 |
| | 5 | 9.903 | -0.004 | 0.002 | 0.025 | 2.015 | -0.018 | 9.884 | 0.000 | 0.000 | 0.003 | -0.008 | 2.003 |
| 5% Noisy | 1 | 4.323 | 0.000 | 0.000 | 1.065 | 0.000 | -1.141 | 9.538 | 3.000 | -0.174 | -0.284 | 0.000 | 0.000 |
| | 2 | 10.965 | 2.315 | 0.000 | 0.000 | 0.000 | 0.134 | 12.309 | 0.000 | 2.658 | -1.036 | 0.227 | 0.000 |
| | 3 | 6.826 | 0.000 | -0.902 | -0.307 | 0.000 | -0.141 | 7.708 | 0.000 | -0.858 | 3.000 | 0.000 | 0.000 |
| | 4 | 8.131 | 0.000 | 0.000 | 2.489 | 0.000 | -0.987 | 11.215 | 0.395 | 0.000 | 0.605 | 1.995 | 0.000 |
| | 5 | 15.100 | 0.000 | 0.000 | 0.468 | 1.097 | 0.000 | 13.269 | 0.317 | -0.148 | 0.000 | 0.000 | 0.966 |
| 10% Noisy | 1 | 2.945 | 0.000 | 0.000 | 0.000 | 0.000 | -1.240 | 7.072 | 2.906 | 0.000 | -1.624 | 0.377 | -0.258 |
| | 2 | 13.039 | 1.085 | 0.501 | 0.000 | 0.218 | 0.000 | 19.985 | 0.000 | 2.883 | 0.000 | 0.000 | 0.418 |
| | 3 | 18.733 | 0.000 | -0.730 | 0.290 | 0.000 | 0.000 | 20.000 | -0.208 | -0.578 | 1.684 | 0.000 | 0.000 |
| | 4 | 7.643 | 0.000 | 0.000 | 2.008 | 0.000 | -1.204 | 11.151 | 0.000 | 0.000 | 1.511 | 3.000 | -0.573 |
| | 5 | 9.618 | 0.000 | 0.000 | 1.178 | 1.414 | 0.000 | 9.438 | 0.000 | 0.000 | 0.900 | -0.342 | 2.121 |
| 25% Noisy | 1 | 2.338 | -1.809 | 0.000 | 0.379 | 0.000 | -1.201 | 6.893 | 1.543 | -0.318 | 0.000 | 0.000 | 0.000 |
| | 2 | 6.005 | 2.847 | 0.000 | 0.000 | 0.000 | -0.482 | 8.280 | 0.000 | 3.000 | -2.702 | 2.725 | 0.000 |
| | 3 | 18.021 | 0.000 | -0.717 | 0.000 | 0.000 | 0.000 | 19.424 | 0.804 | -0.574 | 3.000 | -0.300 | 0.000 |
| | 4 | 6.771 | 0.000 | 0.000 | 0.000 | 0.000 | -0.576 | 13.075 | 3.000 | 0.000 | -2.394 | 2.046 | 0.942 |
| | 5 | 17.515 | 0.592 | 0.372 | 0.000 | 1.231 | 0.000 | 16.530 | 0.000 | 0.000 | -0.490 | 0.000 | 3.139 |

estimated when inference was done using the proposed fitness function.

To perform a direct comparison among the effectiveness of the four fitness functions the *average error* in estimated parameter values in different runs were calculated. The error $e_p$ in an estimated parameter was defined as $e_p = |p - p'|$ where $p$ and $p'$ are target and estimated parameter values respectively. The *average error* of estimated parameters in $M$ different runs is given by $E = \frac{1}{M\wp}\sum_1^M \sum_1^\wp e_p$, where $\wp$ is the number of parameters estimated in a single run; in this study the values were $\wp = 60$ and $M = 10$. The *average errors* using different fitness functions and different error levels are compared in graph of Fig. 6.4. Error bars indicate the standard deviations for multiple runs. For every fitness function the average error increased with the increase in noise level but among these four fitness functions, in each case, the minimum *average error* was found for the proposed fitness function of (3.18).

For further validation the sensitivity $S_n$ and specificity $S_p$ of network models were calculated as follows

$$S_n = \frac{TP}{TP + FN} \qquad S_p = \frac{TN}{TN + FP} \tag{6.2}$$

where TP, FN, TN, FP represent the number of *True positive*, *False Negative*, *True Negative* and *False Positive* prediction of parameters. The average $S_n$ and $S_p$ of ten runs for different noise levels obtained using the fitness functions of (3.18), (3.17), (3.13) and (6.1) are compared in Table 6.12. Consulting Table 6.12 it can be found that only for the noise free condition $S_p$ obtained using (3.18) was worse than that

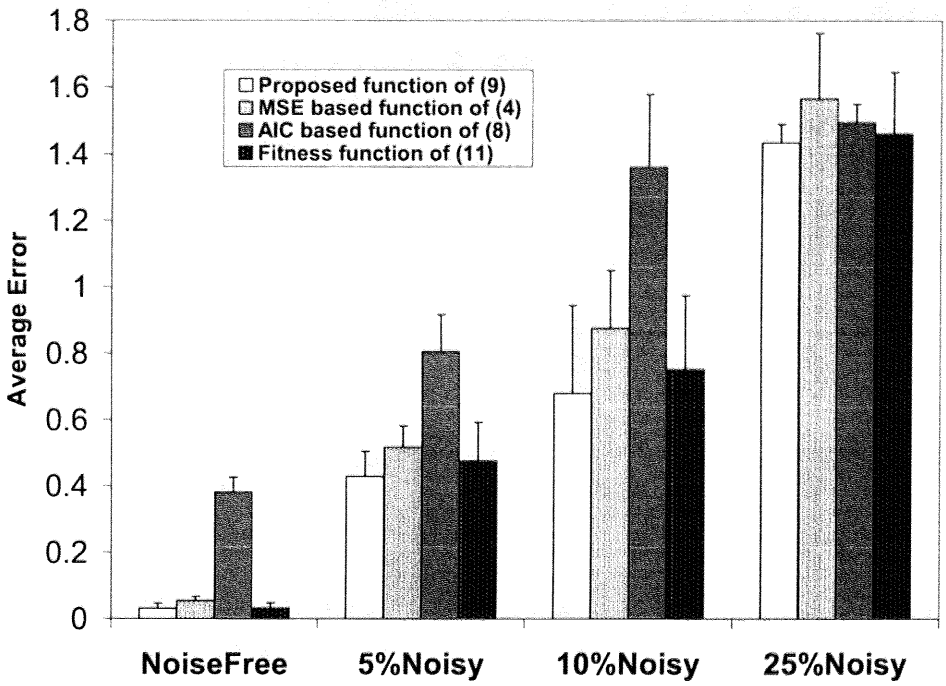


Figure 6.4: Average error in estimated parameters of NET1 for different noise levels in expression data

obtained using (3.17) and in 25% noisy condition $S_n$ of (3.18) was slightly inferior to that of (3.13) and (6.1). In all other cases $S_n$ and $S_p$ of the proposed fitness function of (3.18) was better or same compared to that of other fitness functions. Though the $S_p$ was worse in noise free condition, the estimated network parameters were more accurate using this fitness function, which can be verified comparing the results of Table 6.8 and Table 6.9 and looking at the graph of Fig. 6.4. The reason of poor $S_p$ (under noise free condition) was many parameter values were very close to skeletalizing threshold $\delta = 0.001$ but was not absolutely zero so the number of FP increased and TN decreased and $S_p$ went down. For further verification, $S_n$ and $S_p$ were recalculated for all fitness functions (under noise free condition) setting the skeletalizing threshold $\delta = 0.01$ and $S_n/S_p$ changed to 1.000/0.822, 1.000/0.805, 1.000/0.321 and 1.000/0.757 for (3.18), (3.17), (3.13) and (6.1) respectively. Nevertheless, the cases in which the proposed fitness criteria of (3.18) performed poor was two extreme cases (no-noise and very high noise). Still, comparing the overall performance it can be stated that the proposed fitness function (3.18) is most suitable for evaluating the candidate network models. And the fitness function of (6.1) has also performed very good with the proposed penalty term alone but when compared to the proposed fitness function of (3.18) the later one was found superior.

Table 6.12: Comparison of sensitivity/specificity for NET1 with different noise levels

| | NoiseFree | 5% Noisy | 10% Noisy | 25% Noisy |
|---|---|---|---|---|
| Fitness of (3.18) | 1.000 / 0.514 | 1.000 / 0.730 | 0.923 / 0.757 | 0.877 / 0.795 |
| Fitness of (3.17) | 1.000 / 0.703 | 1.000 / 0.676 | 0.908 / 0.643 | 0.864 / 0.524 |
| Fitness of (3.13) | 1.000 / 0.135 | 1.000 / 0.335 | 0.908 / 0.438 | 0.923 / 0.746 |
| Fitness of (6.1) | 1.000 / 0.384 | 1.000 / 0.676 | 0.923 / 0.649 | 0.908 / 0.643 |

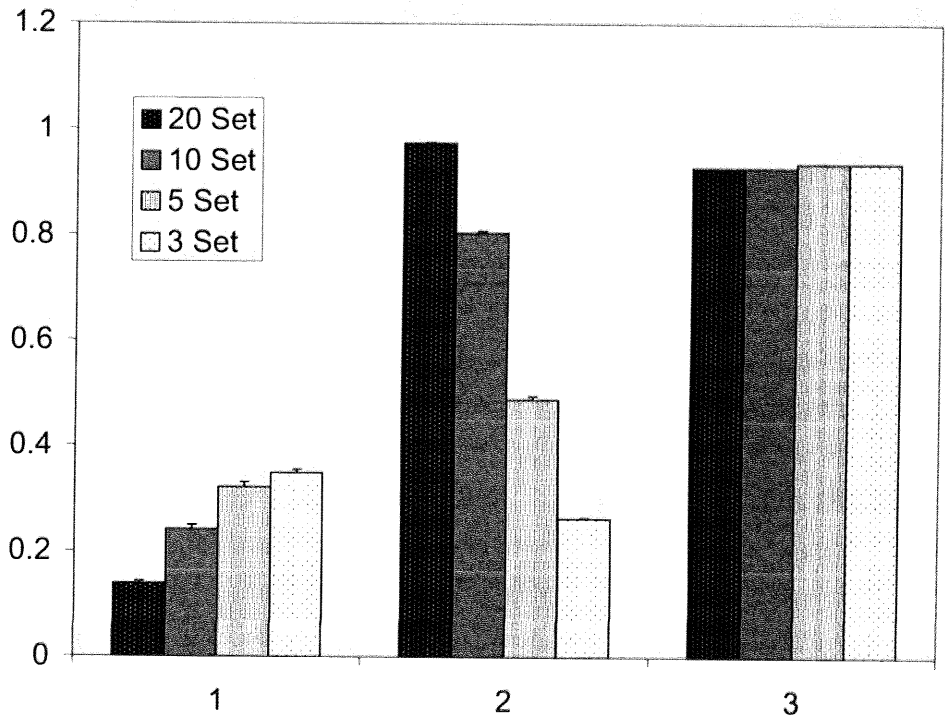


Figure 6.5: Effect of supplied data sets on algorithm's performance

## 6.4 Effect of Available Gene Expression Data

In Machine Learning, it is well known that the more parameters are involved in a model, the harder the prediction task becomes. Moreover, because of the nonlinear and dynamic nature of the problem, the gene network inference using S-system and other differential equation based models tend to be data-hungry [25]. The data requirement for inferring an N gene network is known for some simpler models e.g. Boolean network. Unfortunately no concrete research has addressed this issue for more complex models. Generally, the amount of data needed to uniquely identify the correct regulatory structure depends on the number of network components, nature of their interactions, the characteristics of the data (e.g. the level of noise present) and the capability of the inference algorithm [87].

In this section how the supplied gene expression data affect the inference al-

gorithm's prediction power was investigated. The medium scale network NET2 of section 6.2.2 was inferred using different number of gene expression data sets in the same inference environment. All the experimental conditions were same as in section 6.2.2, except the number of gene expression data sets were varied to 20, 10, 5 and 3 where each data set was corrupted with 10% Gaussian Noise. Then the average error $E$, sensitivity $S_n$ and specificity $S_p$ as defined in previous section were measured for each case. The average of 10 runs of these measurements are compared in the graph of Fig. 6.5. Error bars indicate the standard deviations for multiple runs.

From the graph of Fig. 6.5 it is apparent that the amount of the supplied data has a drastic effect on the performance of the algorithm and the prediction power decreases with the decrease in amount of gene expression data. Particularly, the sensitivity decreases exponentially with the decrease in number of data set. The average error is inversely proportional with the amount of gene expression data. Since, the number of inferred interactions is limited by the maximum in-degree $I$, the effect on sensitivity due to change in data sets is not visible in Fig. 6.5. However, based on the results presented above, it can be stated that the proposed method can identify the underlying biomolecular interactions in a gene circuit if sufficient amount of experimental dynamics is given. Moreover, the capacity of the algorithm to infer some essential regulations from an inadequate noisy expression profile indicates its robustness against small sample size and noise.

## 6.5 Effect of Random Number Generator

The proposed inference algorithm for gene network reconstruction is a stochastic algorithm that makes frequent use of pseudorandom number generators (PRNG). Generally, it is known that the performance of evolutionary algorithms are not terribly sensitive to the quality of PRNG used to drive it [73, 74]. But this is not a general case for all class of stochastic algorithms. Therefore, in this section, it is examined to what degree the quality of the PRNG employed affects the performance of the proposed memetic algorithm in gene network reconstruction.

All of the experiments presented in this dissertation use the Random class of Java that uses a linear congruential algorithm [100] with a 48-bit seed. Lets name it as RANLCG. Linear congruential generators (LCGs) represent one of the oldest and best-known pseudorandom number generator algorithms. The theory behind them is easy to understand, and they are easily implemented and fast. It is, however, well

Table 6.13: Inferred parameters for NET1 using different PRNGs

| | $i$ | $\alpha_i$ | $g_{i1}$ | $g_{i2}$ | $g_{i3}$ | $g_{i4}$ | $g_{i5}$ | $\beta_i$ | $h_{i1}$ | $h_{i2}$ | $h_{i3}$ | $h_{i4}$ | $h_{i5}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RANLCG | 1 | 5.007 | 0.000 | -0.008 | 0.977 | -0.004 | -0.996 | 10.010 | 1.975 | 0.000 | 0.000 | 0.000 | 0.000 |
| | 2 | 10.049 | 1.995 | 0.004 | 0.005 | 0.004 | -0.001 | 10.089 | 0.000 | 2.002 | 0.000 | 0.005 | 0.008 |
| | 3 | 9.667 | -0.006 | -1.000 | -0.020 | 0.000 | -0.002 | 9.665 | -0.006 | -1.001 | 2.048 | 0.000 | -0.002 |
| | 4 | 7.970 | 0.000 | -0.017 | 1.946 | 0.000 | -0.996 | 10.007 | -0.029 | 0.000 | 0.000 | 1.990 | 0.000 |
| | 5 | 10.008 | 0.000 | 0.002 | 0.021 | 2.001 | -0.010 | 9.986 | 0.008 | 0.000 | 0.000 | 0.000 | 1.986 |
| RANECU | 1 | 5.007 | 0.000 | -0.008 | 0.978 | -0.004 | -0.996 | 10.009 | 1.975 | 0.000 | 0.000 | 0.000 | 0.000 |
| | 2 | 10.066 | 1.994 | 0.005 | 0.010 | 0.002 | -0.002 | 10.076 | 0.000 | 1.996 | 0.013 | 0.000 | 0.003 |
| | 3 | 9.803 | -0.001 | -1.002 | -0.010 | 0.000 | 0.000 | 9.808 | 0.000 | -1.002 | 2.031 | 0.000 | 0.002 |
| | 4 | 7.970 | 0.000 | -0.017 | 1.946 | 0.000 | -0.996 | 10.007 | -0.029 | 0.000 | 0.000 | 1.990 | 0.000 |
| | 5 | 9.899 | -0.004 | 0.002 | 0.025 | 2.015 | -0.018 | 9.882 | 0.000 | 0.000 | 0.003 | -0.008 | 2.004 |
| RANLUX | 1 | 4.986 | 0.000 | -0.009 | 0.983 | -0.004 | -1.000 | 9.994 | 1.984 | 0.000 | 0.000 | 0.000 | 0.000 |
| | 2 | 10.059 | 1.995 | 0.005 | 0.012 | 0.000 | -0.002 | 10.051 | 0.000 | 1.995 | 0.019 | -0.004 | 0.000 |
| | 3 | 9.771 | -0.001 | -1.002 | -0.012 | 0.000 | -0.001 | 9.773 | 0.000 | -1.002 | 2.034 | 0.000 | 0.000 |
| | 4 | 7.970 | 0.000 | -0.017 | 1.946 | 0.000 | -0.996 | 10.007 | -0.029 | 0.000 | 0.000 | 1.990 | 0.000 |
| | 5 | 9.829 | 0.000 | 0.000 | 0.015 | 2.021 | -0.022 | 9.818 | 0.000 | 0.000 | 0.000 | -0.013 | 2.009 |
| RANMAR | 1 | 4.978 | 0.000 | -0.012 | 0.978 | -0.003 | -0.999 | 9.965 | 1.981 | -0.004 | 0.000 | 0.000 | 0.000 |
| | 2 | 10.056 | 1.994 | 0.005 | 0.000 | 0.006 | 0.000 | 10.110 | 0.000 | 2.001 | -0.010 | 0.009 | 0.012 |
| | 3 | 9.883 | -0.003 | -1.001 | -0.005 | 0.000 | -0.001 | 9.884 | -0.002 | -1.002 | 2.014 | 0.000 | 0.000 |
| | 4 | 7.969 | 0.000 | -0.017 | 1.946 | 0.000 | -0.996 | 10.006 | -0.029 | 0.000 | 0.000 | 1.991 | 0.000 |
| | 5 | 9.887 | -0.004 | 0.002 | 0.025 | 2.017 | -0.019 | 9.869 | 0.000 | 0.000 | 0.003 | -0.009 | 2.005 |
| RANMT | 1 | 5.003 | 0.000 | -0.008 | 0.978 | -0.004 | -0.997 | 10.006 | 1.976 | 0.000 | 0.000 | 0.000 | 0.000 |
| | 2 | 10.066 | 1.994 | 0.005 | 0.005 | 0.003 | 0.000 | 10.098 | 0.000 | 1.998 | 0.002 | 0.004 | 0.009 |
| | 3 | 10.339 | 0.000 | -1.001 | 0.023 | 0.000 | 0.000 | 10.338 | 0.000 | -1.001 | 1.952 | 0.000 | 0.000 |
| | 4 | 7.970 | 0.000 | -0.017 | 1.946 | 0.000 | -0.996 | 10.007 | -0.029 | 0.000 | 0.000 | 1.990 | 0.000 |
| | 5 | 10.009 | 0.000 | 0.003 | 0.021 | 2.001 | -0.010 | 9.987 | 0.008 | 0.000 | 0.000 | 0.000 | 1.986 |

known that the properties of this class of generator are far from ideal.

The sensitivity of the the proposed reconstruction algorithm to PRNG was verified using four other random number generators those are available form RngPack 1.1 a pseudorandom number generator package in Java [41]. This package offers four PRNGs namely RANECU, RANLUX, RANMAR and Mersenne Twister. RANECU is an advanced multiplicative linear congruential random number generator with a period of aproximately $10^{18}$. RANLUX is an advanced pseudo-random number generator based on the RCARRY algorithm proposed in 1991 by Marsaglia and Zaman. RANMAR is a lagged Fibonacci generator proposed by Marsaglia and Zaman and is a good research grade generator. Mersenne Twister is a PRNG developed in 1997 by Makoto Matsumoto and Takuji Nishimura that provides for fast generation of very high quality pseudorandom numbers.

The proposed reconstruction algorithm was driven by all five PRNGs for estimating the parameters of the network model NET1 under the noise-free condition. The setup for all experiments was same as that of section 6.2.1 and for each PRNG the reconstruction experiment was repeated five times. Typical estimated parameters for all PRNGs are presented in Table 6.13.

Comparing the results obtained by driving the algorithm with different PRNGs, it was found that there is almost no significant difference in the estimated model parameter sets. In other words, the chosen PRNG has almost no effect on the

performance of the proposed algorithm which is the general case for evolutionary algorithms. Although the study is completely empirical, but the results definitely suggest that the accuracy of the reconstructed network is not correlated with the quality of the PRNG used to drive the algorithm.