Adaptive Local Search Parameters for Real-Coded Memetic Algorithms

Daniel Molina
Dept. of Computer Science and AI.
University of Granada, 18071
Granada, Spain
dmolina@decsai.ugr.es

Francisco Herrera
Dept. of Computer Science and AI.
University of Granada, 18071
Granada, Spain
herrera@decsai.ugr.es

Manuel Lozano
Dept. of Computer Science and AI.
University of Granada, 18071
Granada, Spain
lozano@decsai.ugr.es

Abstract-

This paper presents a real-coded memetic algorithm that combines a high diversity global exploration with an adaptive local search method to the most promising individuals that adjusts the local search probability and the local search depth. In our proposal we use the individual fitness to decide when local search will be applied (local search probability) and how many effort should be applied (the local search depth), focusing the local search effort on the most promising regions. We divide the individuals of the population into three different categories and we assign different values of the above local search parameters to the individual in function of the category to which that individual belongs.

In this study, we analyze the performance of our proposal when tackling the test problems proposed for the Special Session of the IEEE Congress on Evolutionary Computation 2005.

1 Introduction

It is now well established that pure *genetic algorithms* (GAs) [7] can be improved by the hybridization with other techniques. GAs that have been hybridized with local search (LS) techniques are often called *memetic algorithms* (MAs) [15]. MAs are evolutionary algorithms that apply a separate LS process to refine new individuals. An important aspect concerning MAs is the trade-off between the exploration abilities of the GA, and the exploitation abilities of the LS used [11].

Under the initial formulation of GAs, the search space solutions are coded using the binary alphabet, however, other coding types, such as real-coding, have also been taken into account to deal with the representation of the problem. The real-coding approach seems particularly natural when tackling optimization problems of parameters with variables in continuous domains. A chromosome is a vector of floating point numbers whose size is kept the same length of the vector, which is the solution to the problem. GAs based on real number representation are called *real-coded* GAs (RCGAs) [2, 9, 14].

For function optimization problems in continuous search spaces, an important difficulty must be addressed: solutions of high precision must be obtained by the solvers [10]. In order to solve this problem, *real-coding* MAs (RCMAs) have been proposed, which incorporate LS to efficiently refining solutions [8, 13]. One commonly used formulation of MAs applies LS to members of the population after recombination and mutation, with the aim of exploiting the best search regions gathered during the global sampling done by the RCGA.

In order to obtain an adequate trade-off of exploration and exploitation for different functions, different adaptive application of local search have been proposed [1, 13, 17]. In this paper we propose a RCMA that adjusts the local search probability and the local search depth. In our proposal we use the individual fitness to decide when local search will be applied (local search probability) and how many effort should be applied (the local search depth), focusing the local search effort on the most promising regions. In this paper, we will study the performance of our proposal when tackling the test problems proposed for the Special Session of Real-Parameter Optimization [20].

The paper is set up as follow. In section 2 we presents our RCMA. In section 3, we describe the experiments carried out in order to determine the suitability of our approach. In section 4, we show the parameter values used on the experiments. Finally, in section 5, we summarize and draw some conclusions.

2 RCMA Model

In this section we introduce the RCMA proposed. The model is composed of two different components:

- The RCGA applied to promote exploration (global search). Its main objective is to maintain diversity in the population.
- The LS applied to improve new solutions. Its main objective is to exploit the most promising regions of the search space.

Each one of these components are explained in detail in the following sections.

2.1 RCGA component

Population diversity is crucial to a GA's ability to continue the fruitful exploration of the search space. If the lack of population diversity takes place too early, a premature stagnation of the search is caused. Under these circumstances, the search is likely to be trapped in a local optimum before the global optimum is found. This problem, called *premature convergence*, has long been recognized as a serious failure mode to GA [3]. In MAs, the GA component should be more explorative oriented, because a part of the exploitation of solutions may be made by the LS procedure.

In our proposal, we use a Steady-State Genetic Algo-

0-7803-9363-5/05/$20.00 ©2005 IEEE.

rithm (SSGA) [21]. In SSGA, usually only one or two offspring are produced in each generation. Parents are selected to produce offspring, and then a decision is made as to which individuals in the population to select for deleting to make room for the new offspring. SSGAs are overlapping systems, since parents and offspring compete for survival. Although SSGAs are less common than generational GAs, different authors [12, 13] recommended their use for the design of steady-state MAs (SSGA plus LS), because they may be more stable (as the best solutions do not get replaced until the newly generated solutions become superior) and allow the results of LS to be maintained in the population.

The characterists of the SSGA applied are the following:

- It uses the $BLX-0.5$ crossover method [4], that have been proved to give significant diversity to population [16].
- It applies the *nonuniform* mutation [14].
- It performs the *Negative Assortative Mating* (NAM) [5] as our selection method. In NAM a first parent is selected by the roulette wheel method and N_{nam} chromosomes are selected with the same method. Then, the similarity between each one of these chromosomes and the first parent is computed (similarity between two real-coded chromosomes is defined as the Euclidean distance between them). Then, the one with less similarity is chosen to be the second parent. In our experiments the roulette wheel method is not used, all the candidate parents are selected at random. This variation is used because it gives in memetic algorithms greater diversity than the original proposal. Clearly, the NAM mechanism increases genetic diversity in the population by mating dissimilar genomes with higher probability.
- As a replacement strategy the standard *Replace Worst Strategy* (RW) [6] is applied. In RW, offspring replaces the worst individual of population only if the new one is better. This strategy is adequate to be used in combination with a LS, because it is an elitist strategy, and they are recommended for MAs [8, 18]. Futhermore, this replacement strategy offers a high selective pressure, making a good complement with the NAM selection method.

2.2 Adaptive Mechanism for the application of Local Search procedure

LS typically operates over a small portion of the total visited solutions, because the additional function evaluations required for local search can be very expensive. Thus, a parameter, called LS Probability (p_{LS}) is introduced which determines the probability that LS will be invoke to refine a new chromosome. The question naturally arises as to how best to select the solutions which will undergo LS.

In [12] the author introduced the concept of "sniff", that is, individual solutions were subject of a limited amount of local search (i.e. a sniff), moreover, those solutions that were in the proximity of a promising basin of attraction received (at a latter stage) and extended cpu budget. With that budget further iterations of local search were performed. Hart [8] addresses this issue and proposes different mechanism for adaptively calculate the application of LS:

Group	when	p_{LS}	N_{itera}
Most Promising	$f(new) < f(best)$	100%	100
Median	$f(best) \leq f(new)$ $f(new) \leq f(worst)$	6.25%	50
Less Promising	$f(worst) < f(new)$	0%	-

Table 1: Application of LS

- Fitness-based adaptive methods use the fitness information in the population bias the LS toward individuals that have better fitness. They modify the p_{LS} of an individuals based on the relationship of its fitness to the rest of population. These method assume that individuals with better fitness are more likely to be in basins of attraction of good local optima.
- Distribution-based adaptive methods use redundancy in the population to avoid performing unnecessary local searches. In particular, selected solutions will be far away from each other, and ideally span as much of the search space as the population itself. This helps ensure locally optimized solutions cover the search space, and tends to prevent premature convergence.

Since the steady-state GA proposed attempts to maintain a diverse population, we focus our attention on fitness-based adaptive methods.

Our proposal is conceptually simple, to classify each offspring into three categories, and to assign different p_{LS} and number of iterations for local search (N_{itera}) values to the offspring in the different categories. The values of these parameters for each category are shown in the table 1.

These categories are defined as following:

1. Individuals with a fitness value better than best fitness in the current population. These individuals are identified as the most promising current solutions, therefore the LS should be applied to each solution in this group: $P_{LS} = 1.0$ and $N_{itera} = 100$.
2. Individuals with a fitness value between the worst individual and the best individual. We consider that a nascent chomosome being better than the current worst element is a promising element, and thus, it deserves a local tuning, but with less probability than the individuals in the first category. In this category the LS is applied with a $p_{LS} = 0.0625$ (standard value [8]) and with $N_{itera} = 50$.
3. Individuals with a fitness value worse than the worst in the population. These are individuals with very low fitness that could not be introduced in population (unless the LS make a serious improvement), so to apply LS in that case will imply a cost in local evalua-

tions that in rarely cases will produce any profit. The LS is not applied to the individuals of this category ($p_{LS} = 0$).

3 Experiments

We have carried out different optimization experiments with the algorithm proposed, on the tests suite of 25 problems described in the CEC2005 Guidelines [20].

3.1 Results with Dim=10

In this section, we present the empirical results obtained by the proposed algorithm when tackling the 25 problems indicated [20] with Dimension D = 10. The maximum number of fitness evaluations (FEs) for this Dimension is 1e5.

We present in tables 2, 3 and 4 the empirical results obtained by our RCMA with Dimension D=10. In each table the error ($f(x) - f(x^*)$) is shown after 1e3,1e4,1e5 FES for each one of the 25 runs. The values of the 25 runs have been sorted and the tables present the following ones: 1st (Best), 7th, 13th (Median), 19th, 25th (Worst), Mean and Standard Deviation (Std). The character 'T' after the error value indicates that the error obtained is inferior than 1e-8. In this case the algorithm stops before the maximum FES is reached.

In table 8 is presented the number of FEs needed in each run to achieve the fixed accuracy level. For each problem and run(25) the number of FEs are sorted and they are presented as follow: 1^{th} (Best), 7^{th}, 9^{th}, 13^{th} and 25^{th} (Worst), the mean, the Std, the success rate (# successful run)/totals run), and the success performance in seconds, defined as (mean(FEs for successful runs)) *(#total runs)/(# of successful runs).

3.2 Results with Dim=30

We present in tables 5, 6 and 7 the empirical results obtained by our RCMA with Dimension D=30. The tables have the same format that tables 2, 3 and 4. The maximum number of fitness evaluations (FEs) for this Dimension is 3e5.

In table 9 is presented the number of FEs needed in each run to achieve the fixed accuracy level. The table has the same format that the table 8.

3.3 Convergence graphs

The graphs 1-5 show the mean performance of the total runs when Dim=30. Graphs show the evolution of the error ($log10(f(x) - f(x*))$) in five functions.

3.4 Algorithm Complexity

The experiments have been obtained by using a Pentium 4, 2.80GHz with 512MB as RAM and Linux (kernel version 2.6). The used programming language was C++ (using the GCC 3.3.2), using the C implementation of the framework test functions. The Table 10 presents the computational costs of the algorithm, as it is defined in [20].

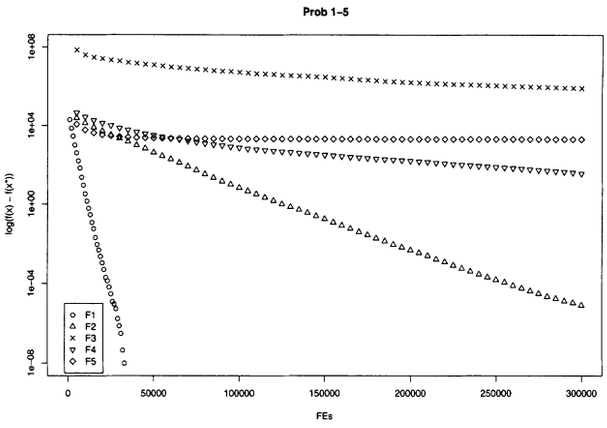

Figure 1: Convergence Graphs for Problems 1-5

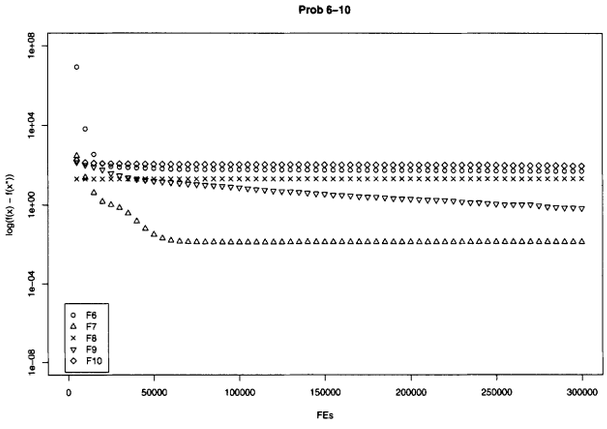

Figure 2: Convergence Graphs for Problems 6-10

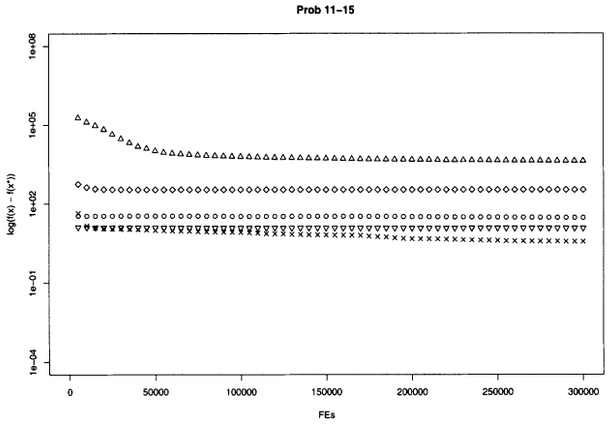

Figure 3: Convergence Graphs for Problems 11-15

T0 is the time of the test program described on [20]. T0 does not depend on the dimension. T1 corresponds to the computing time for 200000 evaluations of the function 3. It is calculated for every dimension. Finally, T2 is the complete computing time for our proposal with 200000 evaluations as stopping criterion when tackling benchmark func-

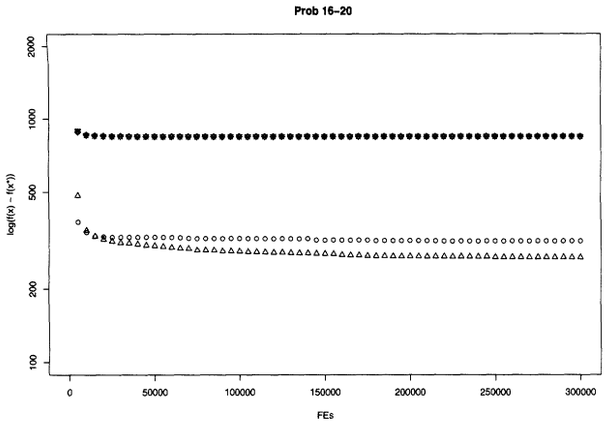

Figure 4: Convergence Graphs for Problems 16-20

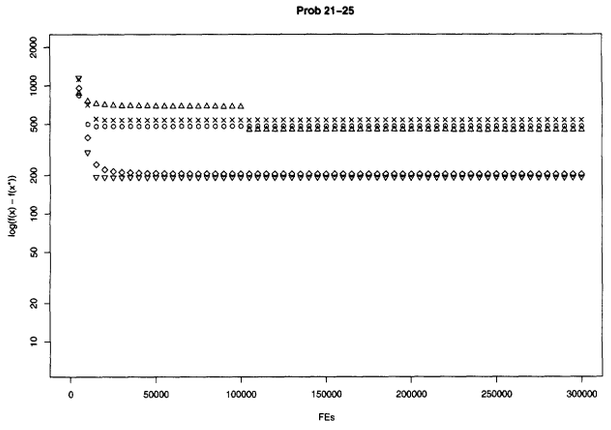

Figure 5: Convergence Graphs for Problems 21-25

tion 3. T2 is recorded five times and the value presented is the mean of the five values.

D	T0	T1	T2	$\frac{T2-T1}{T0}$
10	420ms	1414ms	4440ms	7.20
30	420ms	8630ms	13452ms	11.48

Table 10: Algorithm Complexity

4 Parameters

- A population size of 60 individuals.
- The crossover operator applied is the BLX-α [4] with $\alpha = 0.5$. The $\alpha = 0.5$ gives high diversity that contribute to a better exploration in the global search.
- The mutation applied is the *nonuniform* mutation. It is applied with a probability of 0.125 (12.5% genes).
- The selection method applied is the NAM with $N_{nam} = 3$. Although each value between 1 and the population size is possible, the NAM obtains better results with a low value [5].
- The LS applied is the Solis Wets [19], with the The Local Search Probabilities and the Local Search Depth indicated in the section 2.2. These parameters have a great influence on results, and they can take many different values.
- The other parameters are fixed following the CEC2005 Guidelines [20].

5 Conclusions

These experiments let us to draw some characteristics of the algorithm. With D=10, the algorithm achieves the desired accuracy proposed [20] only for the test fuctions 1, 2 in every execution (and with 4 in 24 of the 25 executions). In test functions 9 and 15 the desired accuracy proposed is achieved only in some executions. When D=30 the algorithm achieves the fixed level only for test function 1, and in some executions in function 9.

Another characteristic in that the algorithm gets trapped in a local optimum very quickly, because of the LS application. The convergence graphs show premature convergence, which indicates that the algorithm could be improved by the addition of a restart strategy.

Finally, we analyze the behaviour of the algorithm when tackling problems with similar characteristics:

- *Different Condition Number* (Functions 1, 2 and 3): The algorithm performs more or less well on function 1 and 2. However, in the highly conditioned function 3 the proposal presents poorer results.
- *Function with Noise Vs Without Noise* (Functions 4 vs 2 and 17 vs 16): In the second pair the difference between them is small, so we will concentrate our comments in the first pair. In the first pair (Function 2 and 4) it is shown that the noise in the test function makes the results worse than without noise. This bad behaviour with a test function with noise was expected, because noise in the test function originates important problems to the LS application.
- *Function With Rotation Vs with Rotation* (Functions 9 vs 10, 15 vs 16): It appears that the rotation affects to the algorithm, specially when they are compared the results in functions 9 and 10. When the dimension number raises the differences raises also. On the other hand, in functions 15 and 16 it is achieved a bit better results with the rotated function.
- *Continuous vs non-continuous* (Functions 21 vs 23): The results with the non-continuous function 23 are worse than with the continuous function 21. In this property the LS method used is very important. In our proposal the LS method used (Solis Wets) obtains better results with a continuous function, but it is quite robust.
- *Global Optimum on Bound vs Narrow Global Optimum Basins* (Functions 18 vs 19): It seems that this property does not affect to the performance of the algorithm.
- *Ortohomal Matrix vs High Condition Number Matrix*

(Functions 21 vs 22): The results for function 21 are quite worse than for function 22. This property affects to the results of the algorithm.

- *Unimodal Functions* (Functions 1-5): In unimodal the results seems good. It achieves the accurary required with D=10 in 1,2 and 4 (in function 4 in 24 of the 25 executions). But when D=30, the performance decreases very quickly.
- *Multimodal Functions* (functions 6-25): We think that in multimodal functions the algorithm performs well, and it is not really affected by the use of higher dimensions.
- *Functions with Global Optimum outside of the Initialization Range* (Function 7 and 25): It seeems that the algorithm is able to go through the fitness landscape looking for the best regions. The algorithm achieves good results in these test functions.
- *Function with Global Optimum on Bounds* (Functions 5, 8 and 20): It appears that having the global optimum on bounds is a important problem for the algorithm proposed.

In short, the proposal obtains acceptable results in the test problems. It is necessary to work in the trade-off between the diversity provided by the GA components and the exploration provided by the LS including a restart strategy for avoiding premature convergence.

Acknowledgment

This research has been supported by proyect TIC-2002-04036-C05-01.

Bibliography

[1] N. K. Bambha, S. S. Bhattacharyya, J. Teich, and E. Zitzler. Systematic integration of parameterized local search into evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2):137–155, 2004.

[2] K . Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley and Sons, 2001.

[3] L. Eshelman and D.J. Schaffer. Preventing premature convergence in genetic algorithms by preventing incest. *Proc. of the Fourth Int. Conf. on Genetic Algorithms*, pages 115–122, 1991.

[4] L. J. Eshelman and J. D. Schaffer. Real-coded genetic algorithms in genetic algorithms by preventing incest. *Foundation of Genetic Algorithms 2*, pages 187–202, 1993.

[5] C. Fernandes and A. Rosa. A study of non-random matching and varying population size in genetic algorithm using a royal road function. *Proc. of the 2001 Congress on Evolutionary Computation*, pages 60–66, 2001.

[6] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, pages 69–93, 1991.

[7] D.E. Goldberg. Genetic algorithms in search, optimization, and machine learning. *Addison-Wesley, Reading, MA*, 1989.

[8] W.E. Hart. *Adaptive Global Optimization With Local Search*. PhD thesis, Univ. California, San Diego, CA., 1994.

[9] F. Herrera, M. Lozano, and J. L. Verdegay. Tackling real-coded genetic algorithms: operators and tools for the behavioral analysis. *Artificial Intelligence Reviews*, 12(4):265–319, 1998.

[10] H. Kita. A comparison study of self-adaptation in evolutionary strategies and real-coded genetic algorithms. *Evolutionary Computation Journal*, 9(2):223–241, 2001.

[11] N. Krasnogor and J. E. Smith. Emergence of profi table search strategies based on a simple inheritance mechanism. *Proceedings of the 2001 International Conference on Genetic and Evolutionary Computation*, pages 432–439, 2001.

[12] M.W. Shannon Land. *Evolutionary Algorithms with Local Search for Combinational Optimization*. PhD thesis, Univ. California, San Diego, CA., 1998.

[13] M. Lozano, F. Herrera, N. Krasnogor, and D. Molina. Real-coded memetic algorithms with crossover hill-climbing. *Evolutionary Computation*, 12(2):273–302, 2004.

[14] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996.

[15] P. A. Moscato. Memetic algorithms: a short introduction. *New Ideas in Optimization*, pages 219–234, 1999.

[16] T. Nomura and K. Shimohara. An analysis of two-parent recombinations for real valued chromosomes in an in nite population. *Evolutionary Computation Journal*, 3(9):283–308, 2001.

[17] Yew Soon Ong and Andy J . Keane. Meta-lamarckian learning in memetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4(2):99–110, 2004.

[18] Abhishek Sinha. *Designing efficient genetic and evolutionary algorithm hybrid*. PhD thesis, Indian Institute of Technology, Kharagpur, 1999.

[19] F. J. Solis and R. J. Wets. Minimization by random search techniques. *Mathematical Operations Research*, 6:19–30, 1981.

[20] P.N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem defi nitions and evaluation criteria for the CEC 2005 Special Session on Real-Parameter Optimization. Technical report, Nanyang Technological University, May 2005.

[21] D. Whitley. The genitor algorithm and selection pressure: why rank-based allocation of reproductive trials is best. *Proc. of the Third Int. Conf. on Genetic Algorithms*, pages 116–121, 1989.

FE	Prob	1	2	3	4	5	6	7	8
1e3	1^{st} (Best)	1.262119e+1	7.658490e+2	1.105278e+6	8.912104e+3	2.624715e+3	3.991610e+3	4.443392e+0	2.010401e+1
	7^{st}	4.019532e+1	1.920095e+3	8.747411e+6	1.167588e+4	3.742181e+3	2.097278e+5	3.098237e+1	2.027422e+1
	13^{st} (Median)	3.114633e+2	2.824513e+3	1.086397e+7	1.553790e+4	5.318624e+3	2.080915e+6	8.530263e+1	2.037782e+1
	19^{st}	1.085511e+3	5.475485e+3	1.848368e+7	1.815881e+4	7.132962e+3	3.595747e+6	2.060035e+2	2.041170e+1
	25^{st} (Worst)	1.844514e+3	1.055641e+4	2.768807e+7	2.216112e+4	1.148651e+4	7.982182e+7	5.466184e+2	2.054972e+1
	Mean	5.706847e+2	3.860661e+3	1.247700e+7	1.532376e+4	5.711970e+3	5.681137e+6	1.577687e+2	2.035422e+1
	Std	1.249861e+2	5.247886e+2	1.382574e+6	7.409367e+2	4.460096e+2	3.092584e+6	3.383493e+1	2.310213e-2
1e4	1^{st} (Best)	1.315698e-8	2.317171e-1	8.142014e+4	1.019428e+2	2.558889e+1	5.530166e+0	8.593690e-2	2.010401e+1
	7^{st}	1.365818e-6	8.910344e-1	1.871005e+5	3.191573e+2	6.523714e+1	6.881649e+0	2.381118e-1	2.018322e+1
	13^{st} (Median)	4.233202e-6	1.781526e+0	3.355640e+5	5.214734e+2	1.101503e+2	7.130548e+0	3.607726e-1	2.024997e+1
	19^{st}	6.208211e-6	3.460448e+0	6.470402e+5	7.409813e+2	1.727256e+2	7.596043e+0	7.780818e-1	2.033574e+1
	25^{st} (Worst)	1.522324e-5	2.764345e+1	1.657183e+6	1.357601e+3	2.531489e+2	8.063997e+0	1.095514e+0	2.041170e+1
	Mean	4.597912e-6	3.659028e+0	5.153548e+5	5.535116e+2	1.253784e+2	7.107893e+0	4.840820e-1	2.024701e+1
	Std	7.715936e-7	1.133016e+0	8.402949e+4	5.966375e+1	1.282582e+1	1.286327e-1	6.478612e-2	1.764862e-2
1e5	1^{st} (Best)	9.869495e-9T	9.936002e-9T	7.213966e+3	8.770940e-9T	2.956301e-8	4.064509e-1	2.215614e-2	2.006735e+1
	7^{st}	9.869495e-9T	9.936002e-9T	2.509596e+4	8.770940e-9T	2.064754e-5	1.405250e+0	1.299165e-1	2.013806e+1
	13^{st} (Median)	9.869495e-9T	9.936002e-9T	3.754519e+4	8.770940e-9T	6.651825e-5	1.533909e+0	1.748641e-1	2.019064e+1
	19^{st}	9.869495e-9T	9.936002e-9T	6.024464e+4	3.421638e-8	1.686571e-4	1.621310e+0	2.668683e-1	2.024046e+1
	25^{st} (Worst)	9.869495e-9T	9.936002e-9T	1.353326e+5	3.421638e-8	4.801763e-1	1.825659e+0	3.922757e-1	2.035160e+1
	Mean	9.869495e-9T	9.936002e-9T	4.770855e+4	1.996693e-8	2.124172e-2	1.489618e+0	1.971385e-1	2.019227e+1
	Std	6.617445e-25	0.000000e+0	6.496540e+3	2.526157e-9	1.879476e-2	5.402242e-2	2.066332e-2	1.376151e-2

Table 2: Error Values Achieved When FES=1e3,FES=1e4,FES=1e5 for Problem 1-8(10D)

FE	Prob	9	10	11	12	13	14	15	16	17
1e3	1^{st} (Best)	1.163174e+1	3.036695e+0	6.367757e+0	8.615226e+2	6.775806e-1	3.917175e+0	3.053222e+2	1.924536e+2	2.660661e+2
	7^{st}	2.755761e+1	3.783668e+1	8.355995e+0	2.180589e+3	5.121136e+0	4.150881e+0	4.743015e+2	2.611655e+2	3.615788e+2
	13^{st} (Median)	3.088117e+1	5.653461e+1	8.722457e+0	3.459871e+3	6.110521e+0	4.392769e+0	5.744346e+2	2.935380e+2	4.156262e+2
	19^{st}	4.170766e+1	6.507879e+1	9.518344e+0	4.578468e+3	8.452387e+0	4.509979e+0	6.223214e+2	3.299604e+2	4.790269e+2
	25^{st} (Worst)	6.757408e+1	1.055160e+2	1.042308e+1	1.336270e+4	1.332129e+1	4.568252e+0	6.961008e+2	4.202376e+2	6.136053e+2
	Mean	3.549666e+1	5.091060e+1	8.718387e+0	3.974608e+3	6.817779e+0	4.331257e+0	5.443637e+2	3.026486e+2	4.239580e+2
	Std	2.841199e+0	4.257620e+0	2.364308e-1	5.384027e+2	6.187049e-1	3.857192e-2	2.064962e+1	1.175869e+1	1.735805e+1
1e4	1^{st} (Best)	2.985914e+0	3.036695e+0	4.969883e+0	1.783408e+1	6.775806e-1	2.913970e+0	4.171912e+1	1.029091e+2	1.309578e+2
	7^{st}	4.421013e+0	6.774883e+0	6.738806e+0	3.972735e+1	1.186024e+0	3.352911e+0	2.046810e+2	1.065643e+2	1.444218e+2
	13^{st} (Median)	5.074841e+0	9.250428e+0	7.503502e+0	1.646180e+2	1.453476e+0	3.531132e+0	3.272458e+2	1.119785e+2	1.578876e+2
	19^{st}	5.716247e+0	1.087035e+1	7.896100e+0	3.827384e+2	1.644548e+0	3.654842e+0	4.203303e+2	1.146951e+2	1.704973e+2
	25^{st} (Worst)	1.026980e+1	1.558639e+1	9.647197e+0	1.176184e+3	2.518001e+0	4.027183e+0	4.366024e+2	1.316701e+2	1.897002e+2
	Mean	5.309446e+0	8.832397e+0	7.334553e+0	2.643701e+2	1.425342e+0	3.513988e+0	3.057060e+2	1.128299e+2	1.566613e+2
	Std	3.268367e-1	6.463855e-1	1.940966e-1	6.065328e+1	7.528632e-2	5.244148e-2	2.508155e+1	1.465986e+0	3.228587e+0
1e5	1^{st} (Best)	8.098311e-9T	3.036695e+0	1.143997e-3	1.184681e-3	3.809260e-1	6.843058e-1	6.451700e-09T	8.963766e+1	1.036632e+2
	7^{st}	8.098311e-9T	4.096764e+0	3.895998e+0	2.782561e+0	5.660125e-1	1.512328e+0	1.582134e+02	9.901256e+1	1.188455e+2
	13^{st} (Median)	4.566927e-4	5.535853e+0	5.836514e+0	1.028080e+1	7.987768e-1	2.174968e+0	3.272458e+2	1.018605e+2	1.260354e+2
	19^{st}	9.949591e-1	6.779161e+0	6.424615e+0	1.924463e+1	9.126824e-1	2.519942e+0	4.065923e+2	1.048056e+2	1.327261e+2
	25^{st} (Worst)	9.949591e-1	9.381669e+0	6.946415e+0	1.110526e+3	1.232675e+0	3.202053e+0	4.203303e+2	1.100834e+2	1.556119e+2
	Mean	4.378553e-1	5.642886e+0	4.557405e+0	7.430320e+1	7.735628e-1	2.030477e+0	2.695663e+02	1.015583e+2	1.269614e+2
	Std	9.876396e-2	3.618307e-1	5.243598e-1	4.431492e+1	4.279666e-2	1.442754e-1	3.254323e+01	9.906657e-1	2.204403e+0

Table 3: Error Values Achieved When FES=1e3,FES=1e4,FES=1e5 for Problem 9-17(10D)

FE	Prob	18	19	20	21	22	23	24	25
1e3	1^{st} (Best)	8.466341e+2	8.117184e+2	8.823276e+2	5.106485e+2	7.027463e+2	1.033592e+3	2.092101e+2	4.464380e+2
	7^{st}	1.018966e+3	9.726784e+2	1.049383e+3	1.184825e+3	9.294043e+2	1.296111e+3	7.626945e+2	4.794847e+2
	13^{st} (Median)	1.075872e+3	1.040544e+3	1.063521e+3	1.273669e+3	9.968203e+2	1.337335e+3	1.254237e+3	5.322984e+2
	19^{st}	1.093713e+3	1.069623e+3	1.084463e+3	1.293268e+3	1.060837e+3	1.354096e+3	1.323839e+3	7.472645e+2
	25^{st} (Worst)	1.151497e+3	1.157198e+3	1.141976e+3	1.336527e+3	1.095247e+3	1.445943e+3	1.435553e+3	1.313318e+3
	Mean	1.052310e+3	1.015431e+3	1.058909e+3	1.179837e+3	9.811002e+2	1.316684e+3	1.070391e+3	6.879292e+2
	Std	1.366224e+1	1.764683e+1	9.645626e+0	4.284998e+1	1.913996e+1	1.602214e+1	7.211232e+1	6.055752e+1
1e4	1^{st} (Best)	8.000159e+2	3.687447e+2	8.000156e+2	5.000000e+2	3.001545e+2	5.594683e+2	2.000000e+2	2.016866e+2
	7^{st}	8.000300e+2	8.000192e+2	8.000310e+2	6.445535e+2	7.623152e+2	9.705031e+2	2.000004e+2	4.069653e+2
	13^{st} (Median)	8.000435e+2	8.000287e+2	8.000453e+2	8.005747e+2	7.692855e+2	9.705031e+2	2.000013e+2	4.077110e+2
	19^{st}	8.001233e+2	8.000526e+2	8.000787e+2	8.079915e+2	7.769706e+2	1.088796e+3	2.000030e+2	4.090925e+2
	25^{st} (Worst)	8.985018e+2	9.227072e+2	8.012572e+2	1.161496e+3	8.000145e+2	1.190426e+3	5.003016e+2	4.126156e+2
	Mean	8.067819e+2	7.720305e+2	8.001165e+2	7.414137e+2	7.210526e+2	9.812801e+2	2.240172e+2	3.999965e+2
	Std	4.616689e+0	2.314691e+1	4.875277e-2	3.130355e+1	2.838575e+1	2.836066e+1	1.628643e+1	8.103884e+0
1e5	1^{st} (Best)	8.000000e+2	3.000000e+2	8.000000e+2	5.000000e+2	3.000000e+2	5.594683e+2	2.000000e+2	2.016866e+2
	7^{st}	8.000000e+2	8.000000e+2	8.000000e+2	6.445535e+2	7.450071e+2	9.705031e+2	2.000000e+2	4.033972e+2
	13^{st} (Median)	8.000000e+2	8.000000e+2	8.000000e+2	8.000000e+2	7.547637e+2	9.705031e+2	2.000000e+2	4.037643e+2
	19^{st}	8.000000e+2	8.000000e+2	8.000000e+2	8.000000e+2	7.605225e+2	9.705031e+2	2.000000e+2	4.040628e+2
	25^{st} (Worst)	8.837135e+2	8.698194e+2	8.000000e+2	8.000000e+2	8.000000e+2	1.052114e+3	5.000000e+2	4.051427e+2
	Mean	8.033485e+2	7.627928e+2	8.000000e+2	7.217970e+2	6.708948e+2	9.267270e+2	2.240000e+2	3.957304e+2
	Std	3.280886e+0	2.743036e+1	0.000000e+0	2.564523e+1	3.724400e+1	2.432270e+1	1.627759e+1	7.922446e+0

Table 4: Error Values Achieved When FES=1e3,FES=1e4,FES=1e5 for Problem 18-25(10D)

FE	Prob	1	2	3	4	5	6	7	8
1e3	1^{st} (Best)	4.785153e3	2.947021e4	9.634190e7	8.126499e4	1.867541e4	5.844203e8	1.103454e3	2.073110e1
	7^{st}	1.389724e+4	5.040182e+4	1.720136e+8	9.660867e+4	2.657141e+4	2.759649e+9	1.834079e+3	2.086607e+1
	13^{st} (Median)	1.948926e+4	5.650342e+4	2.689526e+8	1.111173e+5	2.870082e+4	4.283744e+9	2.704106e+3	2.092294e+1
	19^{st}	2.655956e+4	6.319208e+4	4.262956e+8	1.286191e+5	3.181499e+4	6.996915e+9	4.790137e+3	2.098840e+1
	25^{st} (Worst)	4.375382e+4	7.997801e+4	5.821066e+8	1.935447e+5	3.964038e+4	1.789269e+10	6.417529e+3	2.104980e+1
	Mean	2.075255e+4	5.526144e+4	2.942487e+8	1.148479e+5	2.855830e+4	5.570027e+9	3.157140e+3	2.091883e+1
	Std	2.045551e+3	2.901075e+3	3.044073e+7	5.081595e+3	9.406852e+2	8.459367e+8	3.398499e+2	1.631366e-2
1e4	1^{st} (Best)	6.235774e-1	8.489698e+3	1.806962e+7	2.148009e+4	4.801445e+3	1.149044e+3	8.532645e+0	2.070544e+1
	7^{st}	2.193336e+0	1.162295e+4	3.285096e+7	2.551881e+4	5.664097e+3	3.851728e+3	1.566550e+1	2.080983e+1
	13^{st} (Median)	2.939848e+0	1.410554e+4	4.164689e+7	2.812103e+4	6.050064e+3	6.745423e+3	2.306794e+1	2.085790e+1
	19^{st}	3.831830e+0	1.601493e+4	4.874844e+7	3.059331e+4	6.726951e+3	9.753103e+3	3.457346e+1	2.090054e+1
	25^{st} (Worst)	9.530704e+0	2.109118e+4	8.508354e+7	3.718710e+4	7.385587e+3	1.585932e+4	4.949210e+1	2.097299e+1
	Mean	3.455112e+0	1.408333e+4	4.143831e+7	2.865751e+4	6.190890e+3	6.955493e+3	2.498603e+1	2.084678e+1
	Std	3.895374e-1	6.436832e+2	2.946109e+6	8.218531e+2	1.374379e+2	7.843925e+2	2.278961e+0	1.392533e-2
1e5	1^{st} (Best)	9.532187e-9T	2.129892e+0	1.762251e+6	2.746883e+2	1.462885e+3	2.390262e+1	9.858620e-3	2.060705e+1
	7^{st}	9.532187e-9T	4.860155e+0	3.310522e+6	4.855926e+2	2.005141e+3	2.435520e+1	9.861830e-3	2.075701e+1
	13^{st} (Median)	9.532187e-9T	6.592619e+0	4.911697e+6	6.315892e+2	2.139488e+3	2.442715e+1	9.864193e-3	2.080282e+1
	19^{st}	9.532187e-9T	9.584608e+0	6.796514e+6	1.116642e+3	2.447244e+3	2.475324e+1	9.866108e-3	2.084384e+1
	25^{st} (Worst)	9.532187e-9T	1.633971e+1	1.629367e+7	1.755111e+3	3.028676e+3	6.638548e+2	2.950429e-2	2.090934e+1
	Mean	9.532187e-9T	7.289625e+0	5.508628e+6	7.791663e+2	2.213529e+3	6.142320e+1	1.329727e-2	2.079068e+1
	Std	0.000000e+0	6.725700e-1	6.051306e+5	7.580473e+1	7.749805e+1	2.514519e+1	1.272050e-3	1.493147e-2
3e5	1^{st} (Best)	5.678373e-9T	2.969620e-6	5.550100e+5	2.212154e+0	1.390569e+3	1.861727e+1	9.857285e-3	2.013079e+1
	7^{st}	9.523433e-9T	5.767513e-6	6.582124e+5	1.194080e+1	1.979758e+3	1.878639e+1	9.857285e-3	2.072175e+1
	13^{st} (Median)	9.746145e-9T	8.750054e-6	7.644183e+5	2.557038e+1	2.128982e+3	1.895552e+1	9.857285e-3	2.077060e+1
	19^{st}	9.838004e-9T	1.162407e-5	1.071497e+6	4.425757e+1	2.434446e+3	1.909375e+1	9.857285e-3	2.081075e+1
	25^{st} (Worst)	9.981534e-9T	1.635608e-5	1.564545e+6	1.817200e+2	2.998772e+3	5.062993e+2	2.949824e-2	2.086685e+1
	Mean	9.363891e-9T	8.717185e-6	8.774563e+5	3.966305e+1	2.182733e+3	4.954603e+1	1.329224e-2	2.071023e+1
	Std	1.899177e-10	7.473824e-7	5.810081e+4	8.554025e+0	7.831320e+1	1.932915e+1	1.272062e-3	3.765799e-2

Table 5: Error Values Achieved When FES=1e3,FES=1e4,FES=1e5 for Problem 1-8(30D)

FE	Prob	9	10	11	12	13	14	15	16	17
1e3	1^{st} (Best)	1.821815e+2	3.332530e+2	3.264595e+1	2.738435e+5	4.086900e+2	1.286064e+1	5.456772e+2	5.225558e+2	7.115639e+2
	7^{st}	2.688272e+2	4.638570e+2	3.588622e+1	3.439062e+5	2.070146e+3	1.372522e+1	6.539371e+2	5.788223e+2	9.640334e+2
	13^{st} (Median)	3.000969e+2	5.208638e+2	3.742189e+1	3.879426e+5	3.861813e+3	1.397975e+1	7.486787e+2	6.136520e+2	1.012108e+3
	19^{st}	3.322981e+2	5.739386e+2	3.842954e+1	4.578980e+5	4.917834e+3	1.408815e+1	8.425728e+2	7.007053e+2	1.160487e+3
	25^{st} (Worst)	4.002867e+2	7.387875e+2	4.177213e+1	7.153358e+5	1.062408e+4	1.435979e+1	1.047236e+3	8.399466e+2	1.229681e+3
	Mean	2.993973e+2	5.169249e+2	3.735887e+1	4.155702e+5	3.951074e+3	1.386704e+1	7.615713e+2	6.430473e+2	1.022346e+3
	Std	1.002936e+1	1.964073e+1	4.164883e-1	2.177495e+4	4.620196e+2	7.038779e-2	2.636883e+1	1.821484e+1	2.843139e+1
1e4	1^{st} (Best)	6.274571e+1	9.900682e+1	3.231074e+1	5.623788e+4	9.973102e+0	1.219226e+1	3.716493e+2	1.420165e+2	2.441006e+2
	7^{st}	9.693920e+1	1.312169e+2	3.463882e+1	1.069305e+5	1.348590e+1	1.335492e+1	4.138459e+2	1.917028e+2	2.619001e+2
	13^{st} (Median)	1.042943e+2	1.404792e+2	3.588622e+1	1.253561e+5	1.492979e+1	1.343054e+1	4.297323e+2	4.119702e+2	2.858393e+2
	19^{st}	1.128897e+2	1.466916e+2	3.694296e+1	1.659512e+5	1.738819e+1	1.358508e+1	4.558649e+2	4.202189e+2	4.697987e+2
	25^{st} (Worst)	1.495608e+2	1.704459e+2	3.899740e+1	2.661896e+5	1.957411e+1	1.379379e+1	5.421944e+2	5.200815e+2	6.093750e+2
	Mean	1.051354e+2	1.390140e+2	3.569857e+1	1.355712e+5	1.514617e+1	1.339735e+1	4.405251e+2	3.557056e+2	3.623821e+2
	Std	3.172582e+0	2.931391e+0	3.123145e-1	1.057410e+4	4.493608e-1	6.363425e-2	7.959993e+0	2.510158e+1	2.714366e+1
1e5	1^{st} (Best)	3.981470e+0	9.036515e+1	2.941181e+1	6.870645e+2	2.775858e+0	1.178606e+1	2.000000e+2	1.079148e+2	1.159982e+2
	7^{st}	4.975026e+0	9.963391e+1	3.406252e+1	2.418593e+3	7.754761e+0	1.251039e+1	3.000000e+2	1.546695e+2	1.871662e+2
	13^{st} (Median)	7.959679e+0	1.107665e+2	3.492811e+1	3.919681e+3	9.067975e+0	1.282528e+1	3.000000e+2	4.000000e+2	2.095141e+2
	19^{st}	9.949632e+0	1.196376e+2	3.569029e+1	8.049848e+3	9.982264e+0	1.295339e+1	4.000000e+2	4.000000e+2	4.300095e+2
	25^{st} (Worst)	1.193985e+1	1.353492e+2	3.720955e+1	3.223388e+4	1.277209e+1	1.338427e+1	5.000000e+2	5.000000e+2	5.381986e+2
	Mean	7.550987e+0	1.101970e+2	3.465414e+1	6.432686e+3	8.659078e+0	1.274700e+1	3.561009e+2	3.350033e+2	2.966728e+2
	Std	5.356655e-1	2.241606e+0	3.276335e-1	1.378259e+3	4.420491e-1	7.176444e-2	1.506403e+1	2.727146e+1	2.959748e+1
3e5	1^{st} (Best)	7.780102e-9	2.188909e+1	6.882403e+0	3.877523e+1	1.330183e+0	1.176834e+1	2.000000e+2	7.142320e+1	1.159982e+2
	7^{st}	9.670714e-9	7.960335e+1	3.231074e+1	4.520953e+2	2.133206e+0	1.230814e+1	3.000000e+2	1.174962e+2	1.741028e+2
	13^{st} (Median)	9.949591e-1	9.511121e+1	3.386457e+1	2.155700e+3	2.540157e+0	1.266529e+1	3.000000e+2	4.000000e+2	1.961053e+2
	19^{st}	9.949591e-1	1.057738e+2	3.463882e+1	4.501364e+3	6.536160e+0	1.286064e+1	4.000000e+2	4.000000e+2	4.011480e+2
	25^{st} (Worst)	1.989918e+0	1.353492e+2	3.720955e+1	3.164688e+4	1.034767e+1	1.307548e+1	5.000000e+2	5.000000e+2	5.025218e+2
	Mean	6.806438e-1	9.058336e+1	3.113689e+1	4.390593e+3	3.963304e+0	1.256268e+1	3.561009e+2	3.262488e+2	2.792652e+2
	Std	1.214401e-1	4.892377e+0	1.524952e+0	1.396365e+3	5.381551e-1	7.810714e-2	1.506403e+1	2.996728e+1	2.755057e+1

Table 6: Error Values Achieved When FES=1e3,FES=1e4,FES=1e5 for Problem 9-17(30D)

FE	Prob	18	19	20	21	22	23	24	25
1e3	1^{st} (Best)	9.597849e+2	9.447712e+2	9.218594e+2	1.242414e+3	1.192309e+3	1.288914e+3	1.288599e+3	1.241049e+3
	7^{st}	9.978650e+2	1.022042e+3	1.011930e+3	1.289318e+3	1.263135e+3	1.350472e+3	1.415036e+3	1.410092e+3
	13^{st} (Median)	1.066333e+3	1.066639e+3	1.085904e+3	1.306183e+3	1.308070e+3	1.374978e+3	1.462482e+3	1.470990e+3
	19^{st}	1.151485e+3	1.129956e+3	1.162866e+3	1.351039e+3	1.404055e+3	1.396644e+3	1.478886e+3	1.519836e+3
	25^{st} (Worst)	1.318342e+3	1.265685e+3	1.370987e+3	1.426287e+3	1.556071e+3	1.435514e+3	1.509907e+3	1.685468e+3
	Mean	1.090253e+3	1.075927e+3	1.102111e+3	1.317160e+3	1.333117e+3	1.368447e+3	1.446678e+3	1.462123e+3
	Std	1.983306e+1	1.568537e+1	2.127084e+1	9.209998e+0	1.867059e+1	7.962296e+0	9.989586e+0	1.747020e+1
1e4	1^{st} (Best)	8.285554e+2	8.850832e+2	8.845718e+2	5.010318e+2	9.674052e+2	6.014400e+2	2.294557e+2	2.405694e+2
	7^{st}	8.896541e+2	8.901372e+2	8.910589e+2	5.043308e+2	9.981548e+2	6.479695e+2	2.579867e+2	2.792244e+2
	13^{st} (Median)	8.923320e+2	8.937507e+2	8.959399e+2	5.071932e+2	1.011379e+3	6.834242e+2	2.961388e+2	3.092993e+2
	19^{st}	8.993581e+2	8.984941e+2	9.001715e+2	5.106806e+2	1.033793e+3	8.096682e+2	3.337401e+2	3.538697e+2
	25^{st} (Worst)	9.091884e+2	9.107455e+2	9.081760e+2	7.834519e+2	1.052595e+3	1.183114e+3	5.439208e+2	1.151859e+3
	Mean	8.915002e+2	8.947336e+2	8.960669e+2	5.194592e+2	1.011220e+3	7.342925e+2	3.136706e+2	4.087138e+2
	Std	2.963196e+0	1.300812e+0	1.181517e+0	1.084970e+1	4.557349e+0	2.844990e+1	1.565411e+1	5.082430e+1
1e5	1^{st} (Best)	8.000000e+2	8.742763e+2	8.679048e+2	5.000000e+2	8.931028e+2	5.341641e+2	2.000000e+2	2.122535e+2
	7^{st}	8.788373e+2	8.804444e+2	8.776628e+2	5.000000e+2	9.083425e+2	5.341641e+2	2.000000e+2	2.125282e+2
	13^{st} (Median)	8.813678e+2	8.821644e+2	8.798319e+2	5.000000e+2	9.132455e+2	5.341644e+2	2.000000e+2	2.128298e+2
	19^{st}	8.833262e+2	8.836646e+2	8.828983e+2	5.000000e+2	9.206759e+2	5.341649e+2	2.000000e+2	2.130218e+2
	25^{st} (Worst)	8.890211e+2	8.901372e+2	8.887574e+2	5.000000e+2	9.272226e+2	1.157567e+3	2.000000e+2	2.136164e+2
	Mean	8.779795e+2	8.821409e+2	8.799020e+2	5.000000e+2	9.137631e+2	5.591006e+2	2.000000e+2	2.128011e+2
	Std	3.250481e+0	6.567333e-1	8.556602e-1	0.000000e+0	1.621938e+0	2.443229e+1	0.000000e+0	6.868520e-2
3e5	1^{st} (Best)	8.000000e+2	8.706848e+2	8.679048e+2	5.000000e+2	8.924177e+2	5.341641e+2	2.000000e+2	2.111042e+2
	7^{st}	8.788373e+2	8.786733e+2	8.773893e+2	5.000000e+2	9.030368e+2	5.341641e+2	2.000000e+2	2.113166e+2
	13^{st} (Median)	8.808548e+2	8.804444e+2	8.798036e+2	5.000000e+2	9.108630e+2	5.341644e+2	2.000000e+2	2.113976e+2
	19^{st}	8.822787e+2	8.825085e+2	8.817026e+2	5.000000e+2	9.140471e+2	5.341649e+2	2.000000e+2	2.116689e+2
	25^{st} (Worst)	8.890211e+2	8.862105e+2	8.887574e+2	5.000000e+2	9.227326e+2	1.157567e+3	2.000000e+2	2.118659e+2
	Mean	8.775680e+2	8.802093e+2	8.792608e+2	5.000000e+2	9.082954e+2	5.591006e+2	2.000000e+2	2.114554e+2
	Std	3.231028e+0	7.402571e-1	8.478890e-1	0.000000e+0	1.693326e+0	2.443229e+1	0.000000e+0	4.482150e-2

Table 7: Error Values Achieved When FES=1e3,FES=1e4,FES=1e5 for Problem 18-25(30D)

Problem	1^{th}	7^{th}	13^{th}	19^{th}	25^{th}	Mean	Std	Success Rate	Success Performance
1	9500	11367	11800	12331	13164	1.173792e+04	1.693026e+02	100%	1.173792e+04
2	33003	34855	36817	38034	40031	3.659800e+04	3.731560e+02	100%	3.659800e+04
3	-	-	-	-	-	-	-	0%	-
4	56852	67191	70211	77384	-	7.156388e+04	1.671349e+03	96%	7.454570e+04
5-8	-	-	-	-	-	-	-	0%	-
9	24127	66394	75372	-	-	7.019528e+04	4.230947e+03	72%	9.749344e+04
10-14	-	-	-	-	-	-	-	0%	-
15	24826	-	-	-	-	5.522080e+04	1.132703e+04	20%	2.761040e+05
16-25	-	-	-	-	-	-	-	0%	-

Table 8: Number of FEs to achieve the desired accuracy levels (10D)

Problem	1^{th}	7^{th}	13^{th}	19^{th}	25^{th}	Mean	Std	Success Rate	Success Performance
1	29689	30825	31403	32367	34006	3.167724e+04	2.358559e+02	100%	3.167724e+04
2-8	-	-	-	-	-	-	-	0%	-
9	171376	288428	-	-	-	2.368290e+05	1.507101e+04	36%	6.578583e+05
10-25	-	-	-	-	-	-	-	0%	-

Table 9: Number of FEs to achieve the desired accuracy levels (30D)