Multiple Trajectory Search for Large Scale Global Optimization

Lin-Yu Tseng and Chun Chen

Abstract—**In this paper, the multiple trajectory search (MTS) is presented for large scale global optimization. The MTS uses multiple agents to search the solution space concurrently. Each agent does an iterated local search using one of three candidate local search methods. By choosing a local search method that best fits the landscape of a solution's neighborhood, an agent may find its way to a local optimum or the global optimum. We applied the MTS to the seven benchmark problems designed for the CEC 2008 Special Session and Competition on Large Scale Global Optimization.**

I. Introduction

Global numerical optimization is an important research issue because many real-life problems can be formulated as global numerical optimization. Many of numerical optimization problems cannot be solved analytically, and consequently, numerical algorithms were proposed to solve these problems. Many of these algorithms are evolutionary algorithms. In CEC 2005, a special session of real-parameter optimization had been organized. Several algorithms [1][4][5][6][11] were presented in this special session and all these algorithms were tested on a suite of 25 benchmark functions with dimensions 10 and 30. Besides these algorithms, some of evolutionary algorithms proposed recently are briefly surveyed in the following. Kazarlis et al. [2] combined a standard GA with a microgenetic algorithm (MGA) [GA with small population and short evolution] to solve the numerical optimization problem. In their method, the MGA operator performs genetic local search. Leung and Wang [3] proposed an orthogonal genetic algorithm with quantization for this problem. They applied the quantization technique and orthogonal design to implement a new crossover operator, such that the crossover operator can generate a small but representative sample of solutions as the potential offspring. Tsai et al. [8] hybridized the genetic algorithm and the Taguchi method. The Taguchi method is incorporated into the crossover operator and the mutation operator. Tu and Lu [10] presented the stochastic genetic algorithm (StGA) for numerical optimization. They employed a stochastic coding method to code chromosomes. Each chromosome is coded as a representative of a stochastic region described by a multivariate Gaussian distribution. Zhong et al. [12] integrated multiagent systems and genetic algorithms to form a new algorithm called the multiagent genetic algorithm (MAGA) for numerical optimization. All above mentioned algorithms except the last one were tested on functions with dimensions less than or equal to 100. The last algorithm (MAGA) was tested on functions with dimensions from 20 to 10,000.

In this paper, the multiple trajectory search (MTS) was presented to solve the large scale global optimization problem. The MTS had been used to solve the multi-objective optimization problems and obtained satisfactory results [9]. We used the same framework but improved the local search methods for numerical optimization. When applying the MTS to the seven benchmark problems designed for the CES 2008 Special Session and Competition on Large Scale Global Optimization [7], the experimental results reveal that the MTS is effective and efficient in solving these problems.

The remainder of this paper is organized as follows. Section II gives some definitions. Section III describes the MTS algorithm. Section VI presents the experimental results and Section IV draws the conclusion.

Lin-Yu Tseng is with the Institute of Networking and Multimedia and the Department of Computer Science and Engineering, National Chung Hsing University, 250 Kuo Kuang Road, Taichung, Taiwan 402, ROC (corresponding author; phone: 886-4-22874020; e-mail: lytseng@cs.nchu.edu.tw).

Chun Chen is a PhD student with the Department of Computer Science and Engineering, National Chung Hsing University, 250 Kuo Kuang Road, Taichung, Taiwan 402, ROC (e-mail:phd9401@cs.nchu.edu.tw).

II. Preliminaries

A. Problem Definition

The global numerical optimization problem is defined as follows.

Minimize $\mathbf{F}(\mathbf{X})$

Subject to $\mathbf{l} \leqq \mathbf{X} \leqq \mathbf{u}$

where $\mathbf{X}=(x_1, x_2, \ldots, x_N)$ is a variable vector in R^N space and $\mathbf{F}(\mathbf{X})$ is the objective function. $\mathbf{l}=(l_1, l_2, \ldots, l_N)$ and $\mathbf{u}=(u_1, u_2, \ldots, u_N)$ define the feasible solution space, that is, a feasible solution $\mathbf{X}=(x_1, x_2, \ldots, x_N)$ must satisfies $li \leqq xi \leqq ui$ for i=1, 2, …, N. The feasible solution space is denote by $[\mathbf{l}, \mathbf{u}]$.

B. Orthogonal Array and Simulated Orthogonal Array

We briefly introduce the concept of orthogonal arrays which are used in experimental design methods. Suppose in an experiment, there are k factors and each factor has q levels. In order to find the best setting of each factor's level, qk experiments must be done. Very often, it is not possible or cost effective to test all qk combinations. It is desirable to sample a small but representative sample of combinations for testing. The orthogonal arrays were developed for this purpose. In an experiment that has k factors and each factor has q levels, an orthogonal array OA(n,k,q,t) is an array with n rows and k columns which is a representative sample of n testing

978-1-4244-1823-7/08/$25.00©2008 IEEE

experiments that satisfies the following three conditions. (1) For the factor in any column, every level occurs the same number of times. (2) For the t factors in any t columns, every combination of q levels occurs the same number of times. (3) The selected combinations are uniformly distributed over the whole space of all the possible combinations. In the notation OA(n,k,q,t), n is the number of experiments, k is the number of factors, q is the number of levels of each factor and t is called the strength.

The orthogonal arrays exist for only some specific n's and k's. So it is not appropriate to use the OA in some applications. We proposed the simulated OA (SOA) in this study. The SOA satisfies only the first of the above mentioned three conditions, but it is easy to construct an SOA of almost any size.

Suppose there are k factors and each factor has q levels, an m×k simulated orthogonal array $SOA_{m\times k}$ with m being a multiple of q can be generated as follows.

For each column of $SOA_{m\times k}$, a random permutation of 0, 1, …, q-1 is generated and denoted as sequence C. Then the elements in C are picked one by one sequentially and filled in a randomly chosen empty entry of the column. If all elements in C were picked, the process picks elements again from the beginning of C. So in every column of $SOA_{m\times k}$, each of q elements will appear the same number of times (condition 1).

III. Multiple Trajectory Search

In this section, we present the multiple trajectory search (MTS) for the large scale global optimization problem.

In the beginning, the MTS generates M initial solutions by utilizing the simulated orthogonal array $SOA_{M\times N}$, where the number of factors corresponds to the dimension N and the number of levels of each factor is taken to be M. So each of 0, 1, …, M-1 will appear once in every column. Using SOA tends to make these M initial solutions uniformly distributed over the feasible solution space. The initial search range for local search methods is set to half of the difference between the upper bound and the lower bound. Afterwards, local search methods will change the search range.

The MTS consists of iterations of local searches until the maximum number of function evaluations is reached. In the first iteration, the MTS conducts local searches on all of M initial solutions. But in the following iterations, only some better solutions are chosen as foreground solutions and the MTS conducts local searches on these solutions. Three local search methods are provided for the MTS. The MTS will first test the performance of three local search methods and then choose the one that performs best, that is, the one that best fits the landscape of the neighborhood of the solution, to do the search. After conducting the search on foreground solutions, the MTS applies Local Search 1 to the current best solution trying to improve the current best solution. Before the end of an iteration, some better solutions are chosen as the foreground solutions for the next iteration. The multiple trajectory search algorithm is described in the following

```
Multiple Trajectory Search
/*Generate M initial solutions */
Build simulated orthogonal array $SOA_{M\times N}$
For i = 1 to M
  For j = 1 to N
    $Xi[j]=l_i+(u_i-l_i)$*SOA[i, j]/(M-1)
  End For
End For
Evaluate function values of Xi's
For i=1 to M
  Enable[i]←TRUE
  Improve[i] ←TRUE
  SearchRangeXi = (UPPER_BOUND-LOWER_BOUND)/2
End For
While ( #ofEvaluation ≦predefined_max_evaluation)
  For i=1 to M
    If Enable[i]=TRUE
    Then  GradeXi←0
          LS1_TestGrade←0
          LS2_TestGrade←0
          LS3_TestGrade←0
          For j =1 to #ofLocalSearchTest
            LS1_TestGrade←LS1_TestGrade+
                LocalSearch1(Xi, SearchRangeXi)
            LS2_TestGrade←LS2_TestGrade+
                LocalSearch2(Xi, SearchRangeXi)
            LS3_TestGrade←LS3_TestGrade+
                LocalSearch3(Xi, SearchRangeXi)
          End For
          Choose the one with the best TestGrade and
          let it be LocalSearchK /* K may be 1, 2, or 3 */
          For j =1 to #ofLocalSearch
            GradeXi←GradeXi+
                LocalSearchK(Xi, SearchRangeXi)
          End For
    End If
  End For
  For i= 1 to #ofLocalSearchBest
        LocalSearch1(BestSolution,
                     SearchRangeBestSolution)
  End For
  For i =1 to M
     Enable[i]←FALSE
  End For
  Choose #ofForeground Xi's whose GradeXi are best
  among the M solutions and set their corresponding
  Enable[i] to TRUE
End While
```

In the MTS, three local search methods are used for searching different landscape of the neighborhood of a solution.

Local Search 1 searches along one dimension from the first dimension to the last dimension. Local Search 2 is similar to Local Search 1 except that it searches along about one-fourth of dimensions. In both local search methods, the search range (SR) will be cut to one-half until it is less than 1×10^{-15} if the previous local search does not make improvement. In Local Search 1, on the dimension concerning the search, the solution's coordinate of this dimension is first subtracted by SR to see if the objective function value is improved. If it is, the search proceeds to consider the next dimension. If it is not, the solution is restored and then the solution's coordinate of this dimension is added by 0.5*SR, again to see if the objective function value is improved. If it is, the search proceeds to consider the next dimension. If it is not, the solution is restored and the search proceeds to consider the next dimension. Local Search 1 and Local Search 2 are listed in the following

```
Function  LocalSearch1(Xk, SR)
If Improve[k]=FALSE
  Then SR = SR /2
       If SR < 1e-15
         Then SR←
              (UPPER_BOUND-LOWER_BOUND) *0.4
       End If
End If
Improve[k] ←FALSE
For  i = 1 to N
  Xk[i] ← Xk[i]- SR
  If  Xk is better than current best solution
    Then grade ← grade  + BONUS1
         Update current best solution
  End If
  If  function  value of Xk is the same
    Then restore Xk to its original value
  Else
    If  function value of Xk degenerates
       Then  restore Xk to its original value
             Xk[i] ← Xk[i] +  0.5* SR
             If   Xk is better than current best solution
                    Then   grade ← grade  + BONUS1
                           Update current best solution
               End If
               If  function value of Xk
                  has not been improved
                  Then restore Xk to its original value
               Else
                    grade ← grade  + BONUS2
                    Improve[k] ←TRUE
               End If
    Else
      grade ← grade  + BONUS2
      Improve[k] ←TRUE
    End If
  End If
End For
return grade

Function  LocalSearch2(Xk, SR)
If Improve[k]=FALSE
  Then SR=SR/2
       If SR < 1e-15
         Then SR ←
              (UPPER_BOUND-LOWER_BOUND) *0.4
       End If
End If
Improve[k] ←FALSE
For  l = 1 to N
    For i = 1 to N
      r[i] ←Random{0,1,2,3}
      D[i] ←Random{-1,1}
    End For
    For i =1 to N
      If r[i]=0
        Then  Xk[i] ← Xk[i]-SR*D[i]
      End If
    End For
  If  Xk is better than current best solution
    Then grade ← grade  + BONUS1
         Update current best solution
    End If
  If  function  value of Xk is the same
      Then restore Xk to its original value
  Else
      if  function value of Xk degenerates
        Then   restore Xk to its original value
               For i =0 to N
                 If  r[i]=0
                   Then  Xk[i] ← Xk[i]+0.5*SR*D[i]
                 End If
               End For
             If  Xk is better than current best solution
                 Then   grade ← grade  + BONUS1
                        Update current best solution
               End If
               If  function value of Xk
                 has not been improved
                 Then restore Xk to its original value
               Else
                   grade ← grade  + BONUS2
                   Improve[k] ←TRUE
               End If
      Else
        grade ← grade + BONUS2
        Improve[k] ←TRUE
      End If
    End If
  End For
  return grade
```

Local Search 3 is different from Local Search 1 and Local Search 2. Local Search 3 considers three small movements along each dimension and heuristically determines the movement of the solution along each dimension. In Local Search 3, although the search is along each dimension from the first dimension to the last dimension, the evaluation of the objective function value is done after searching all the dimensions, and the solution will be moved to the new position only if the objective function has been improved at this evaluation. Local Search 3 is described in the following.

```
Function  LocalSearch3(X,SR)
```

```
For $i = 1$ to $N$
  $X_1 \leftarrow X$'s $i$th coordinate is increased by 0.1
  $Y_1 \leftarrow X$'s $i$th coordinate is decreased by 0.1
  $X_2 \leftarrow X$'s $i$th coordinate is increased by 0.2
  If $X_1$ is better than current best solution
    Then $grade \leftarrow grade + BONUS1$
        Update current best solution
  End If
  If $Y_1$ is better than current best solution
    Then $grade \leftarrow grade + BONUS1$
        Update current best solution
  End If
  If $X_2$ is better than current best solution
    Then $grade \leftarrow grade + BONUS1$
        Update current best solution
  End If
  $D_1 = F(X)-F(X_1)$
  If $D_1>0$ /*$X_1$ is better than $X$*/
    Then $grade \leftarrow grade + BONUS2$
  End If
  $D_2 = F(X)-F(Y_1)$
  If $D_2>0$ /*$Y_1$ is better than $X$*/
    Then $grade \leftarrow grade + BONUS2$
  End If
  $D_3 = F(X)-F(X_2)$
  If $D_3>0$ /*$X_2$ is better than $X$*/
    Then $grade \leftarrow grade + BONUS2$
  End If
  a $\leftarrow$ Random[0.4, 0.5]
  b $\leftarrow$ Random[0.1, 0.3]
  c $\leftarrow$ Random[0, 1]
  $X[i] = X[i] + a(D_1 - D_2) + b(D_3-2D_1) + c$
End For
If function value of $X$ has not been improved
  Then restore $X$ to its original value
Else
    $grade \leftarrow grade + BONUS2$
End If
return $grade$
```

In our experiments, M is set to 5 and #ofForeground is set to 3. Hence three best solutions are in the foreground and the other two solutions are in the background. The MTS applies the local search method that best fits the landscape of a solution's neighborhood to this solution. From iteration to iteration, a solution in the foreground may be put in the background and vice verse.

IV. Experimental Results

The proposed MTS was applied to the seven benchmark functions for CEC2008 Special Session and Competition on Large Scale Global Optimization. The PC configuration and the parameter setting are stated in the following.

A. PC Configuration

For Problems 1-6
System: Windows XP RAM: 1GB
CPU: Intel Pentium D 2.66GHz
For Problem 7
System: Linux RAM: 2GB
CPU: Intel Xeon E5310 1.6GHz
Language: C++

B. Parameter Setting

M=5	#$ofForeground$=3
#$ofLocalSearchTest$=3	#$ofLocalSearch$=100
#$ofLocalSearchBest$=150	$BONUS1$=10
$BONUS2$=1	a=Random[0.4, 0.5],
b=Random[0.1, 0.3]	c=Random[0, 1],

C. Experimental Results

The function error value (f(x)-f(x*)) after FES/100, FES/10, and Max_FES are listed in Table 1, 2 and 3 respectively for seven problems with dimensions 100, 500 and 1000. Seven benchmark problems are listed in Table 4. The details can be found in [7]. The convergence graphs for problems 1-6 and problem 7 with dimension D=1000 are depicted in Figure 1 and Figure 2 respectively.

V. Conclusions

The multiple trajectory search (MTS) was presented and applied to solve the seven benchmark problems provided for the purpose of competition in CEC 2008. The seven problems are with dimensions 100, 500 and 1000, so the scalability of algorithms can also be tested. When MTS searches the neighborhood of a solution, it tests the performance of the three predefined local search methods first and then chooses the best one to conduct the local search. Therefore, the MTS performs its search automatically adapting to the landscape of the solution's neighborhood. For problems 1-6 whose optimal solutions are already known, the MTS can find the optimal solutions of four problems with FES/10 when dimension is 100. Also, it can find the optimal solutions of two and three problems with FES/10 when dimension is 500 and 1000 respectively.

Acknowledgments

The authors gratefully acknowledge the support of National Science Council of ROC under the contract NSC 96-2628-E-005-074-MY3.

References

[1] P. J. Ballester, J. Stephonson, J. N. Carter and K.Gallagher, "Real-parameter optimization performance study on the CEC-2005 benchmark with SPC-PNX," Proceedings of 2005 IEEE Congress on Evol. Comput., .pp. 498-505, 2005.

[2] S. A. Kazarlis, S. E. Papadakis, J. B. Theocharis, and V. Petridis, "Microgenetic algorithms as generalized hill-climbing operators for GA optimization," IEEE Trans. Evol. Comput., vol. 5, pp. 204–217, Jun. 2001.

[3] Y. W. Leungand Y. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," IEEE Trans. Evol. Comput., vol. 5, pp. 41–53, Feb. 2001.

[4] P. Posik, "Real-parameter optimization using the mutation step co-evolution," Proceedings of 2005 IEEE Congress on Evol. Comput., .pp. 872-879, 2005.

[5] J. Ronkkonen, S. Kukkonen and K.V. Price, "Real-parameter optimization with Differential Evolution," Proceedings of 2005 IEEE Congress on Evol. Comput., .pp.506-513, 2005.
[6] A. Sinha, S. Tiwari and K. Deb, "A population-base, state procedure for real-parameter optimization," Proceeding of 2005 IEEE Congress on Evol. Comput., pp. 514-521, 2005.
[7] K. Tang, X. Yao, P. N. Suganthan, C.MacNish, Y. P. Chen, C. M. Chen and Z. Yang, "Benchmark function for the CEC'2008 Special Session and Competition on Large Scale Global Optimization," Technical Report, http://www.ntu.edu.sg/home/EPNSugan, 2007.
[8] J. T. Tsai, T. K. Liu, and J. H. Chou, "Hybrid Taguchi-genetic algorithm for global numerical optimization," IEEE Trans. Evol. Comput., vol. 8, pp. 365-377, Aug. 2004.
[9] L. Y. Tseng and C. Chen, "Multiple trajectory search for multiobjective optimization," Proceedings of 2007 IEEE Congress on Evol. Comput., .pp. 3609-3616, 2007.
[10] Z. Tu and Y. Lu, "A robust stochastic genetic algorithm for global numerical optimization," IEEE Trans. Evol. Comput., vol. 8, pp. 456-470, Oct. 2004.
[11] B. Yuan, M. Gallagher, "Experimental result for the Special Session on real-parameter optimization at CEC 2005: a simple, continuous EDA," Proceedings of 2005 IEEE Congress on Evol. Comput., .pp. 1792-1799, 2005.
[12] W. Zhong, J. Liu, M. Xue, and L. Jiao, "A multiagent genetic algorithm for global numerical optimization," IEEE Trans. on system, man ,and cybernetics Part B, pp. 1128-1141, Apr. 2004.

Table 1 Error values achieved for problems 1-6 and function values for problem 7, with D=100

FES	Prob	1	2	3	4	5	6	7
5.00e+3	1th(best)	2.7647E+03	4.5091E+01	1.4427E-01	3.0150E+02	6.2602E+01	7.7166E+00	-1.3010E+03
	7th	1.1054E+04	4.9549E+01	1.4704E+01	3.6804E+02	1.5153E+02	1.2654E+01	-1.2865E+03
	13th(median)	1.5884E+04	5.0089E+01	1.7724E+01	4.0088E+02	1.7169E+02	1.3316E+01	-1.2730E+03
	19th	1.8170E+04	5.0787E+01	4.8272E+02	4.6015E+02	1.8816E+02	1.3792E+01	-1.2681E+03
	25th(worst)	2.4348E+04	6.1540E+01	1.1203E+03	5.2764E+02	2.2808E+02	1.4581E+01	-1.2598E+03
	Mean	1.4326E+04	5.0646E+01	3.1953E+02	4.1221E+02	1.6219E+02	1.2776E+01	-1.2772E+03
	Std	5.6652E+03	3.1603E+00	4.5086E+02	6.2943E+01	4.3027E+01	1.6268E+00	7.3752E+00
5.00e+4	1th(best)	0.0000E+00	2.1902E-02	1.1369E-13	0.0000E+00	0.0000E+00	0.0000E+00	-1.4866E+03
	7th	0.0000E+00	2.6218E-02	1.6596E-06	0.0000E+00	0.0000E+00	0.0000E+00	-1.4739E+03
	13th(median)	0.0000E+00	4.0085E-02	1.9553E-06	0.0000E+00	0.0000E+00	0.0000E+00	-1.4686E+03
	19th	0.0000E+00	5.6363E-02	2.3313E-06	0.0000E+00	0.0000E+00	0.0000E+00	-1.4646E+03
	25th(worst)	0.0000E+00	8.9314E-02	1.4180E-04	5.5161E-10	0.0000E+00	0.0000E+00	-1.4567E+03
	Mean	0.0000E+00	4.3509E-02	9.7317E-06	2.2064E-11	0.0000E+00	0.0000E+00	-1.4695E+03
	Std	0.0000E+00	1.8510E-02	2.8397E-05	1.1032E-10	0.0000E+00	0.0000E+00	4.5071E+00
5.00e+5	1th(best)	0.0000E+00	3.9790E-13	5.6843E-14	0.0000E+00	0.0000E+00	0.0000E+00	-1.4992E+03
	7th	0.0000E+00	1.9327E-12	5.8043E-09	0.0000E+00	0.0000E+00	0.0000E+00	-1.4914E+03
	13th(median)	0.0000E+00	5.5707E-12	9.0498E-09	0.0000E+00	0.0000E+00	0.0000E+00	-1.4872E+03
	19th	0.0000E+00	1.7167E-11	1.5000E-08	0.0000E+00	0.0000E+00	0.0000E+00	-1.4801E+03
	25th(worst)	0.0000E+00	4.7976E-11	7.8752E-07	0.0000E+00	0.0000E+00	0.0000E+00	-1.4743E+03
	Mean	0.0000E+00	1.4406E-11	5.1707E-08	0.0000E+00	0.0000E+00	0.0000E+00	-1.4861E+03
	Std	0.0000E+00	1.8685E-11	1.6085E-07	0.0000E+00	0.0000E+00	0.0000E+00	5.7916E+00

Table 2 Error values achieved for problems 1-6 and function values for problem 7, with D=500

FES \ Prob		1	2	3	4	5	6	7
2.50e+4	1th(best)	8.6438E+04	8.9075E+01	2.8251E-01	2.2379E+03	1.3709E+01	1.3532E+01	-6.0937E+03
	7th	9.2725E+04	8.9382E+01	1.2535E+01	2.3095E+03	1.8958E+03	1.3861E+01	-6.0228E+03
	13th(median)	9.5394E+04	9.0030E+01	1.8252E+01	2.3758E+03	1.9609E+03	1.4062E+01	-5.9951E+03
	19th	1.0495E+05	9.0030E+01	1.8253E+01	2.4129E+03	2.0956E+03	1.4205E+01	-5.9760E+03
	25th(worst)	1.2322E+05	1.0698E+02	1.9294E+02	2.4678E+03	2.1973E+03	1.4400E+01	-5.9623E+03
	Mean	9.9562E+04	9.0456E+01	2.3310E+01	2.3658E+03	1.9103E+03	1.4027E+01	-6.0029E+03
	Std	1.0638E+04	3.4693E+00	3.7387E+01	6.6189E+01	4.1064E+02	2.1631E-01	1.7398E+01
2.50e+5	1th(best)	0.0000E+00	1.6376E+01	1.5776E-05	6.5370E-12	0.0000E+00	5.6843E-12	-7.1223E+03
	7th	0.0000E+00	2.1972E+01	9.3813E-04	9.9496E-01	0.0000E+00	6.1675E-12	-7.0687E+03
	13th(median)	0.0000E+00	2.2794E+01	9.5042E-04	1.9899E+00	5.6843E-14	6.3096E-12	-7.0394E+03
	19th	0.0000E+00	2.3351E+01	9.5646E-04	1.9899E+00	1.6753E-08	6.4517E-12	-7.0334E+03
	25th(worst)	0.0000E+00	2.3868E+01	6.7785E-01	7.1945E+02	1.2139E-06	2.6347E-11	-7.0206E+03
	Mean	0.0000E+00	2.2490E+01	2.7993E-02	3.0333E+01	9.4814E-08	7.0816E-12	-7.0520E+03
	Std	0.0000E+00	1.5201E+00	1.3539E-01	1.4357E+02	2.6099E-07	4.0223E-12	1.4290E+01
2.50e+6	1th(best)	0.0000E+00	3.7258E-06	1.4161E-07	0.0000E+00	0.0000E+00	5.6843E-12	-7.1479E+03
	7th	0.0000E+00	5.8826E-06	8.0616E-06	0.0000E+00	0.0000E+00	6.0538E-12	-7.0961E+03
	13th(median)	0.0000E+00	7.1825E-06	8.8171E-06	0.0000E+00	0.0000E+00	6.1675E-12	-7.0715E+03
	19th	0.0000E+00	8.2984E-06	1.0829E-05	0.0000E+00	0.0000E+00	6.2812E-12	-7.0622E+03
	25th(worst)	0.0000E+00	1.2922E-05	1.2569E-01	0.0000E+00	0.0000E+00	6.7075E-12	-7.0427E+03
	Mean	0.0000E+00	7.3194E-06	5.0366E-03	0.0000E+00	0.0000E+00	6.1812E-12	-7.0810E+03
	Std	0.0000E+00	2.2418E-06	2.5136E-02	0.0000E+00	0.0000E+00	2.7982E-13	1.5485E+01

Table 3 Error values achieved for problems 1-6 and function values for problem 7, with D=1000

FES \ Prob		1	2	3	4	5	6	7
5.00e+4	1th(best)	1.8199E+05	9.4802E+01	5.0906E-01	1.8285E+03	1.3709E+01	1.3528E+01	-1.1939E+04
	7th	1.9277E+05	9.4926E+01	1.4508E+00	4.6820E+03	1.8958E+03	1.3615E+01	-1.1842E+04
	13th(median)	2.0102E+05	9.4926E+01	1.4508E+00	4.7300E+03	1.9609E+03	1.3675E+01	-1.1824E+04
	19th	2.2698E+05	9.5091E+01	1.4508E+00	4.7770E+03	2.0956E+03	1.3807E+01	-1.1802E+04
	25th(worst)	2.5055E+05	1.1998E+02	5.4980E+01	4.9186E+03	2.1973E+03	1.4252E+01	-1.1790E+04
	Mean	2.0957E+05	9.6005E+01	3.6401E+00	4.5145E+03	1.9103E+03	1.3747E+01	-1.1830E+04
	Std	2.1285E+04	4.9998E+00	1.0718E+01	7.9034E+02	4.1064E+02	1.9018E-01	1.5079E+01
5.00e+5	1th(best)	0.0000E+00	4.1371E+01	2.7003E-04	0.0000E+00	0.0000E+00	2.6944E-11	-1.4001E+04
	7th	0.0000E+00	4.6166E+01	2.3133E-02	1.9916E+00	0.0000E+00	2.2681E-10	-1.3967E+04
	13th(median)	0.0000E+00	4.6824E+01	2.3133E-02	9.9162E+00	5.6843E-14	1.9151E-09	-1.3941E+04
	19th	1.0687E-11	4.6895E+01	2.3133E-02	5.7546E+02	1.6753E-08	1.9788E-07	-1.3892E+04
	25th(worst)	3.2142E-02	4.7299E+01	2.3165E-02	1.4924E+03	1.2139E-06	1.3024E-02	-1.3860E+04
	Mean	1.3591E-03	4.6456E+01	1.9810E-02	2.8340E+02	9.4814E-08	7.2045E-04	-1.3935E+04
	Std	6.4235E-03	1.1567E+00	7.8188E-03	4.7450E+02	2.6099E-07	2.6705E-03	3.4509E+01
5.00e+6	1th(best)	0.0000E+00	3.4985E-02	2.6963E-06	0.0000E+00	0.0000E+00	1.1596E-11	-1.4070E+04
	7th	0.0000E+00	4.0570E-02	3.9946E-04	0.0000E+00	0.0000E+00	1.2108E-11	-1.4040E+04
	13th(median)	0.0000E+00	4.7536E-02	3.9946E-04	0.0000E+00	0.0000E+00	1.2392E-11	-1.3998E+04
	19th	0.0000E+00	5.0680E-02	3.9946E-04	0.0000E+00	0.0000E+00	1.2733E-11	-1.3954E+04
	25th(worst)	0.0000E+00	7.6393E-02	4.0023E-04	0.0000E+00	0.0000E+00	1.3188E-11	-1.3944E+04
	Mean	0.0000E+00	4.7239E-02	3.4060E-04	0.0000E+00	0.0000E+00	1.2378E-11	-1.3999E+04
	Std	0.0000E+00	8.5799E-03	1.3839E-04	0.0000E+00	0.0000E+00	4.4797E-13	2.9397E+01

Table 4 Benchmark problems

Benchmark problems	Search range	Optimal solution value F(x*)
$F_1(x) = \sum_{i=1}^{D} z_i + f\_bias_1, z = x - o$ $o = [o_1, o_2, ..., o_D]$: the shifted global optimum.	[-100,100]	-450
$F_2(x) = \max_i \{\lvert z_i \rvert, 1 \le i \le D\} + f\_bias_2, z = x - o$ $o = [o_1, o_2, ..., o_D]$: the shifted global optimum.	[-100,100]	-450
$F_3(x) = \sum_{i=1}^{D-1} (100(\mathrm{z_i}^2 - \mathrm{z_{i+1}})^2 + (\mathrm{z_i} - 1)^2) + f\_bias_3, z = x - o + 1$ $o = [o_1, o_2, ..., o_D]$: the shifted global optimum.	[-100,100]	390
$F_4(x) = \sum_{i=1}^{D} (\mathrm{z_i}^2 - 10\cos(2\pi \mathrm{z_i}) + 10) + f\_bias_4, z = x - o$ $o = [o_1, o_2, ..., o_D]$: the shifted global optimum.	[-5,5]	-330
$F_5(x) = \sum_{i=1}^{D} \frac{\mathrm{z_i}^2}{4000} - \prod_{i=1}^{D} \cos(\frac{z_i}{\sqrt{i}}) + 1 + f\_bias_5, z = x - o$ $o = [o_1, o_2, ..., o_D]$: the shifted global optimum.	[-600,600]	-180
$F_6(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)) + 20 + f\_bias_6,$ $z = x - o, o = [o_1, o_2, ..., o_D]$: the shifted global optimum.	[-32,32]	-140
$F_7(x) = \sum_{i=1}^{D} fractal1D(x_i + twist(x_{(i \bmod D)+1}))$ $twist(y) = 4(y^4 - 2y^3 + y^2)$ $fractal1D(x) \approx \sum_{k=1}^{3} \sum_{1}^{2^{k-1}} \sum_{1}^{ran2(o)} doubledip(x, ranl(o), \frac{1}{2^{k-1}(2 - ranl(o))})$ $doubledip(x, c, s) = \begin{cases} (-6144(x-c)^6 + 3088(x-c)^4 - 392(x-c)^2 + 1)s, -0.5 < \mathrm{x} < 0.5 \\ 0, \text{otherwise} \end{cases}$ $ranl(o)$: double, pseudorand omly chosen, with seed o, with equal probabilit y from the interval [0,1]. $ran2(o)$: integet, pseudorand omly choose, with seed o, withequal probabilit y from the set {0,1,2}	[-1,1]	unknown

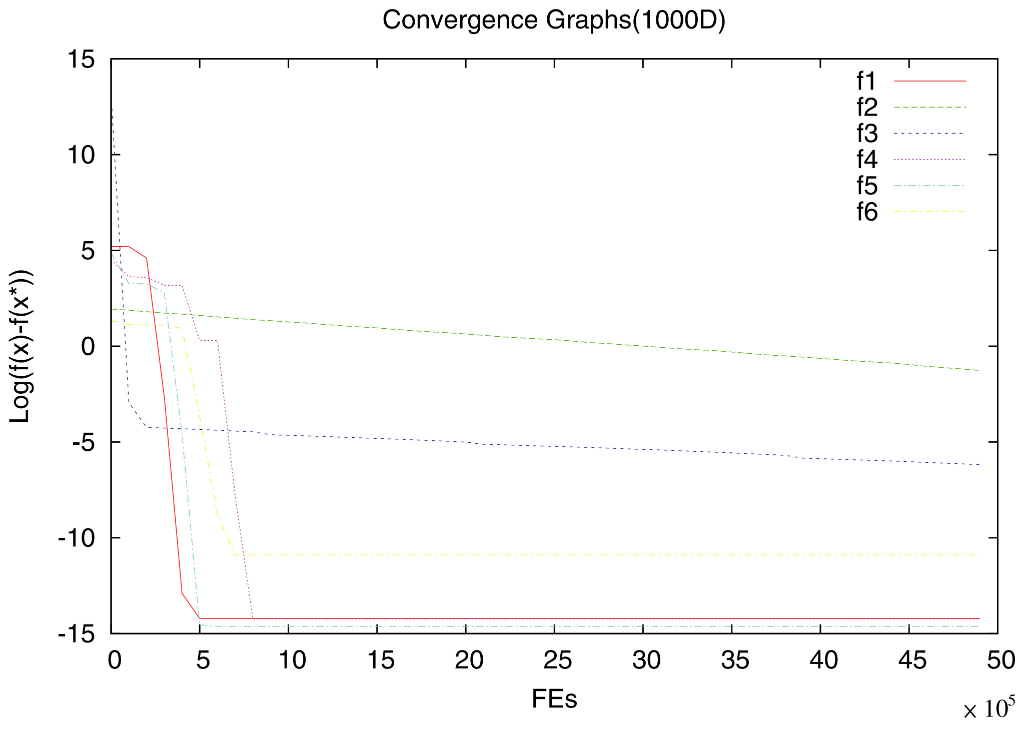

Figure 1 Convergence graphs for problems 1-6 with dimension D = 1000.

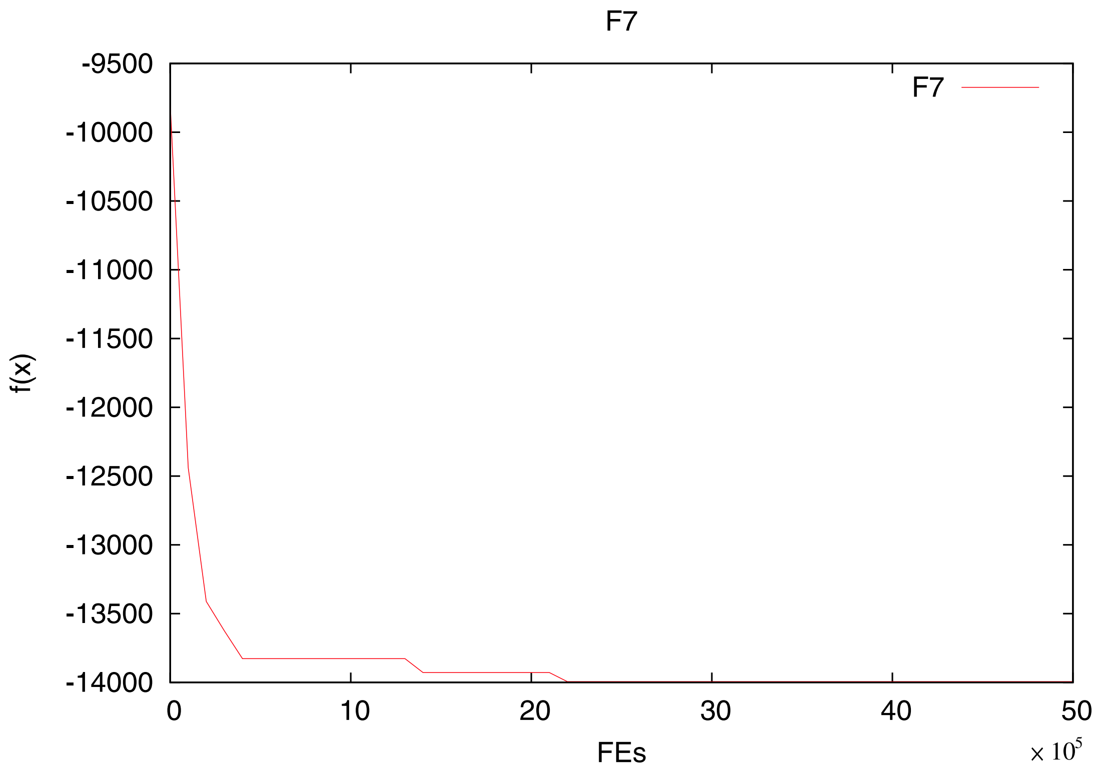

Figure 2 Convergence graph for problem 7 with dimension D = 1000.