Smoothing and Auxiliary Functions Based Cooperative Coevolution for Global Optimization

Fei Wei, Yuping Wang, Yuanliang Huo
School of Computer Science and Technology
Xidian University
Xi’an, China
Email: wyp9000@yahoo.com.cn

***Abstract*—In this paper, a novel evolutionary algorithm framework called smoothing and auxiliary functions based cooperative coevolution (Briefly, SACC) for large scale global optimization problems is proposed. In this new algorithm pattern, a smoothing function and an auxiliary function are well integrated with a cooperative coevolution algorithm. In this way, the performance of the cooperative coevolution algorithm may be improved. In SACC, the cooperative coevolution is responsible for the parallel searching in multiple areas simultaneously. Afterwards, an existing smoothing function is used to eliminate all the local optimal solutions no better than the best one obtained until now. Unfortunately, as the above takes place, the smoothing function will lose descent directions, which will weaken the local search. However, a proposed auxiliary function can overcome the drawback, which helps to find a better local optimal solution. A clever strategy on BFGS quasi-Newton method is designed to make the local search more efficient. The simulations on standard benchmark suite in CEC’2013 are made, and the results indicate the proposed algorithm SACC is effective and efficient.**

I. Introduction

Large scale global optimization problems with numerous local and global optima have arisen in many fields such as computer science, engineering design, and decision making.

A. Previous Work

In recent years, many new theoretical and computational contributions have been reported for solving large scale global optimization problems, e.g., cooperative coevolution [1]–[6], decomposition methods [7], [8], estimation of distribution algorithms [9]–[11], Memetic algorithms [12], [13]. However, due to the complexity of large-scale problems, designing a very effective algorithm becomes very difficult. In this paper, we focus on a novel algorithm framework based on cooperative coevolution.

For global optimization, in existing approaches, evolutionary algorithms (EAs) are one of the most efficient and popular algorithms. They exploit a set of potential solutions, named a population, and detect the optimal solution through cooperation and competition among individuals of the population. However, for EAs, the major challenge is that an algorithm may be trapped in the local optima of the objective function. This issue is particularly challenging when the dimension of the problem is high and there are numerous local optima. The performance on such problem deteriorates rapidly as the dimensionality of the problem increases. For these problems, the cooperative coevolution (CC), proposed by Potter and De Jong [5], is more popular and efficient than the classical evolutionary algorithms (EAs), and it is a framework for decomposition of problem into smaller subcomponents, each of which is evolved by using a separate EA [5]. Unfortunately it can also trap into the local optimal solutions, especially when the dimensionality of the problem or the interaction between variables increases. In order to solve these high-dimensional and complex problems effectively, we combine the smoothing function and auxiliary function into CC.

The smoothing function can eliminate all such local optimal solutions no better than the best solution found so far, and can keep all better local optimal solutions than the best solution found so far unchanged. But it can lose some useful information when looking for a descent direction. The auxiliary function transforms the current local minimal solution into a better local optimal solution of the original objective function. However, it often cost the auxiliary function a lot of time that searching for a global optimal solution by repeating the process of jumping from one local optimal solution to another better one. This makes the auxiliary function with a lower efficiency in search for a global optimal solution, especially for problem with large number of local optimal solutions. While CC can search the local optimal solutions simultaneously in multiple areas of the search space. If CC is integrated with smoothing function and auxiliary function to form a new algorithm, the new algorithm may overcome the limitations of CC, smoothing function and auxiliary function.

B. Goals

To address the above issues, the goal of the paper is to develop a new method that smoothing and auxiliary functions based CC for large scale global optimization. If it is specifically combined with the smoothing function, auxiliary function, and CC in a reasonable way, the resulted algorithm may be expected to have the following advantages: 1) can eliminate a large number of local optimal solutions with the evolving progress; 2) can escape from the local optima; 3) can search for multiple areas in the search space simultaneously by CC. Based on this motivation, in this paper, a new evolutionary algorithm called SACC is designed which integrates the advantages of smoothing function, auxiliary function and CC. The simulations are made on 15 benchmark functions for the CEC’2013 special session and competition on large-scale global optimization [14].

C. Organization

The reminder of the paper is outlined as follows. Section II presents the basic concepts and some remarks. In section III, an existing smoothing function is introduced. A novel auxiliary function is proposed in section IV. Section V shows a novel algorithm framework that smoothing and auxiliary function based cooperative coevolution. Numerical experiments are given in section VI. Section VII presents conclusions and future work directions.

II. BASIC CONCEPT AND SOME REMARK

In this paper, we consider the following global optimization problem:

$$(P) \quad \begin{cases} \min & f(x) \\ s.t. & x \in R^n. \end{cases}$$

where $f(x) : R^n \to R$. Suppose $f(x)$ satisfies the condition $f(x) \to +\infty$ as $\|x\| \to +\infty$. Then there exists a closed bounded domain Ω called an operating region that contains all local minimizers of $f(x)$. Then the global optimization problem (P) can be rewritten into an equivalent form as follows.

$$(P1) \begin{cases} \min & f(x) \\ s.t. & x \in \Omega. \end{cases}$$

where $\Omega = [l,u] = \{x | x \in l \leq x \leq u, l, u \in R^n\}$. Because Ω can be estimated before problem (P) is solved, so we can assume that Ω is known without loss of generality. We only consider problem (P1) in the following, and adopt the following symbols.
k : the iteration number;
x_k^*:the local minimizer of the objective function in the k-th iteration;
f_k^*:the function value at x_k^*;
x^*:the global minimizer of the objective function.

Assumption 1 The function $f(x)$ in (P1) is continuously differentiable in R^n and $f(x)$ has only a finite number of minimizers in Ω, and therefore every minimizer is isolated.

In the following, we first introduce an existing smoothing function in [15]. The details are as follows.

III. SMOOTHING FUNCTION

The existence of multiple local minima of a general non-convex objective function makes global optimization become a great challenge. The key issue for the global optimization problem is effectively handling a large number of local optimal solutions and finding the global optimal solution as soon as possible. In order to tackle this problem, a smoothing function [15] at the current local minimizer for is used as follows:

$$S(x,x_k^*) = f(x_k^*) + 1/2 \cdot \{1 - sign[f(x) - f(x_k^*)]\} \cdot [f(x) - f(x_k^*)] \tag{1}$$

Obviously, this smoothing function has the following properties:

- $S(x,x_k^*)$ will keep any better local optimal solution than x_k^* of $f(x)$ unchanged, i.e., for any local optimal solution x_{k+1}^* of $f(x)$ better than x_k^*, x_{k+1}^* is also a local optimal solution of $S(x,x_k^*)$.

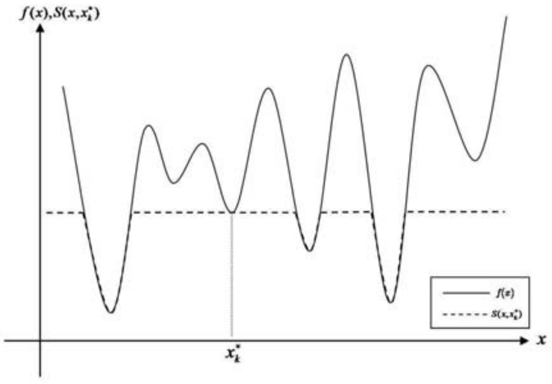

Fig. 1. $S(x,x_k^*)$ is indicated by the dotted line and has only three local optimal solutions, and three local optimal solutions of $f(x)$ were eliminated, where x_k^* is the best solution found so far.

- $S(x,x_k^*)$ will eliminate all local optimal solutions no better than the current local optimal solution x_k^*, and flatten the landscape at any point no better than x_k^*, i.e., for $\forall x \in \Omega$, if $f(x) \geq f(x_k^*)$, then $S(x,x_k^*) = f(x_k^*)$.

This means that the smoothing function will not destroy the region containing better solutions than the best solution found so far and the global optimal solution of $f(x)$.

The properties of smoothing function can be intuitively illustrated by a function with one variable in the following Fig. 1, where the solid line represents the original function, and the dotted line represents the smoothing function. It can be seen from Fig. 1 that the smoothing function (dotted line) eliminates three local optimal solutions, but keep the better three local optimal solutions unchanged.

However, it also can be seen from Fig. 1 that the smoothing function often loses some useful information while looking for a descent direction. For example, at point x_k^*, either right or left, the direction is not descending for smoothing function. To overcome this limitation, we design another function called auxiliary function. The auxiliary function can be constructed as follows.

IV. AUXILIARY FUNCTION

When we have found a local minimizer of $f(x)$ and used the smoothing function to flatten the landscape of $f(x)$, the difficulty arisen is how to escape from the current local minimizer to reach a lower local minimizer.

In this section, we propose an auxiliary function at a local minimizer x_k^* as follows:

$$F(x,x_k^*) = -\|x - x_k^*\|^2 g(f(x) - f(x_k^*)), \tag{2}$$

$$g(t) = \begin{cases} \pi/2, & t \geq 0, \\ r_1 \cdot \arctan(t^2) + \pi/2, & t < 0. \end{cases}$$

where r_1 is an adjustable large positive real number used as the weight factor.

Note that the proposed auxiliary function has some advantages: first, it has one parameter r_1 which is a positive real number as large as possible, thus it is easy to adjust.

Second, $\arctan(t^2) \in [0, \pi/2)$ is bounded, which ensures that the calculation of $F(x, x_k^*)$ will not overflow and is of numerical stability.

The auxiliary function can change the flat area, and has no local minimizer in the landscape higher than the landscape at the current local minimizer. The major issue for the auxiliary function is to find a lower minimizer than the current local minimizer of $f(x)$ when the current local minimizer is not a global minimizer of $f(x)$.

Using the auxiliary function on the smoothing function can escape from the current local minimizer and reach to a lower local minimizer, however, it cannot search in multiple region parallelly. When there are a lot of local minimizers and we are not lucky enough, we have to find all local minima in order to find the global minimizer, e.g., the smoothing function can only eliminate a few (or even no) local minima each time and the auxiliary function can only gradually jump from the current local minimizer to another which is just better than the current one. In this case, the search will enumerate almost all local minima in order to get the global minimizer. This will cost a lot of computation and cannot make the algorithm efficient. Note that evolutionary algorithm has the ability to exploit and explore the multiple regions of the search space in parallel, and has more possibility to find multiple local minima in one generation, especially at the beginning of the evolution. However, it is generally known that most of the evolutionary algorithms are easily trapped in the local optimum and appeared premature convergence. Thus, if we integrate evolutionary algorithm with smoothing and auxiliary functions, it is very possible to eliminate a lot of minima and jump from a local minimizer to a much better local minimizer, especially in the later stage of evolution, such that the computation of the proposed algorithm is much decreased and its efficiency may be increased.

In the following section, we integrate smoothing function and auxiliary function into cooperative coevolution to design a new algorithm: smoothing and auxiliary functions based cooperative coevolution (SACC).

V. SMOOTHING AND AUXILIARY FUNCTIONS BASED EVOLUTIONARY ALGORITHM

In order to make the advantages of our algorithm more clearly reflected, one of the most popular algorithms MLCC [6] is used in our algorithm. The crossover, mutation, selection will be introduced in [6]. The local search scheme is very important to the performance of evolutionary algorithms, and we will introduce how to design it in the following.

A. A local search strategy

A local search strategy is often executed after a good solution is found by using the evolutionary algorithm. A good local search strategy would be helpful to improve the searching efficiency of the algorithm. Some efficient local search algorithms, e.g., Conjugate Gradient Method, Newton's Method and Quasi Newton Method, require the gradients of functions. Therefore, these methods are not suitable for solving nondifferentiable problems. To adopt the advantages of these local search algorithms and avoid computing the gradients, a revised version of the BFGS Quasi Newton algorithm is designed. The detail is as follows:

Algorithm 1 (local search strategy)

```
Step 1. Initialization Step
    a. Choose a tolerance $\varepsilon > 0$, e.g., $\varepsilon = 1.0e-5$, and a small constant $\delta \in (0,1)$, $\sigma \in (0, 0.5)$.
    b. Give an initial point $x_0$ and an approximate inverse of the Hessian matrix $B_0$.
    c. $k = 0$.
Step 2. Calculate the forward difference quotient
    $[f(x_k + \triangle x) - f(x_k)]/\triangle x$ at $x_k$,
    if $\|g_k\| \le \varepsilon$
    stop $x^* = x_k$, and output $x_k$;
    else go to step 3;
    endif
Step 3. Obtain a direction $d_k$ by solving $B_k d_k = -g_k$.
Step 4. Perform a line search based on the Armijo criterion [16] to find an acceptable stepsize $\lambda_k = \delta^{m_k}$, where $m_k$ is the smallest non-negative integer that satisfy the following inequality:
        $$f(x_k + \delta^{m_k} d_k) \le f(x_k) + \sigma \delta^{m_k} g_k^T d_k.$$
    Then update $x_{k+1} = x_k + \lambda_k d_k$.
Step 5. Let $S_k = \lambda_k d_k$, and $y_k = g_(k+1) - g_k$, then
        $$\begin{cases} B_{k+1} = B_k + [\beta_k s_k s_k^T - B_k y_k s_k^T - s_k y_k^T B_k]/s_k^T y_k \\ \beta_k = 1 + (y_k^T B_k y_k)/(s_k^T y_k) \end{cases}$$
Step 6. Let $k = k + 1$, and go to step 2.
```

B. *Smoothing and auxiliary function based cooperative coevolution (SACC)*

Based on the preparations above, a novel evolutionary algorithm: smoothing and auxiliary functions based CC, briefly denoted by SACC, is proposed as follows.

Algorithm 2 (SACC)

```
Step 1. Initialization Step
    Let $k = 0$. Choose a positive real numbers $r_1$ large enough, $m > 0$ is a positive integer, $\varepsilon$ is a tolerance threshold, and the population size is $N$. Generate $N$ points uniform randomly, and put them into the initial population $POP(k)$.
Step 2. Evolve $POP(k)$ by MLCC on original function $f(x)$ for one generation, and then get set $OFF(k)$ of all offspring. Select the best individual among $POP(k) \bigcup OFF(k)$ . This best individual is denoted as $x_k^*$. Execute the local search at $x_k^*$ on $f(x)$ by algorithm 1 to get a local minimizer $x_{k+1}^*$ . Go to Step 7.
Step 3. Eliminate all local minima no better than $x_k^*$ by smoothing function: Construct a smoothing function $S(x, x_k^*)$ at $x_k^*$ by formula (1).
Step 4. Escape from the minimizer $x_k^*$ via auxiliary function: Construct an auxiliary function at $x_k^*$ by formula (2).
Step 5. Execute the local search at $x_k^*$ on $F(x, x_k^*)$ to get a local minimizer $y_k^*$ of the auxiliary function.
```

Step 6. Update the current best solution: Starting from y_k^*, do the local search on $f(x)$ by algorithm 1 to obtain a local minimizer x_{k+1}^* of $f(x)$. Then go to Step 7.

Step 7. If the termination condition is satisfied, $x_k^* = x_{k+1}^*$ is taken as a global minimizer of $f(x)$, stop; otherwise, if $f(x_{k+m}^*) - f(x_k^*) \leq \varepsilon$, where $m > 0$ is a perset positive integer, and ε is a tolerance threshold. Then go to step 3; else put $OFF(k)$ into $POP(k+1)$, let $k = k+1$, go to Step 2.

VI. NUMERICAL EXPERIMENTS

In this section, benchmark suite, parameters setting for SACC, and the simulation results are given.

A. Benchmark suite and parameters setting for SACC

- In this section, the proposed algorithm SACC is tested on CEC'2013 benchmark suite, the detailed description of which can be found in [14]. f1-f3 are fully separable functions, and f4-f11 are partially additively separable functions, and f12-f14 are overlapping functions and f15 is nonseparable functions.
- SACC is executed 25 independent runs for each test problem. In experiments, SACC was tested on an Intel(R) Core(TM) i7 CPU 870 with 2.93GHz in Matlab R2012a.
- Population size: $N = 50$.
- Parameters in algorithm SACC: $r_1 = 100$, $m = 50$, $\varepsilon = 1.0e-4$.

B. The simulation results

The statistic data over 1000 Dimension for all problems except f9 is 905 dimensional after maximum number of fitness evaluations ($MaxFEs = 3.0e6$) in 25 runs is listed in table I. The first column lists the number of function evaluations. Sort the best function values achieved after the given number of function evaluations in 25 runs from the smallest to the largest. The "Best" means that the smallest value in 25 runs; the "Median" is the median value in 25 runs; the "Worst" is the largest value in 25 runs; the "Mean" is the average value in 25 runs; the "Std" is the standard deviation in 25 runs.

The best, median, worst, mean, and standard deviation of the 25 runs are recorded in table I when $FEs = 1.2e5$, 6.0e5 and 3.0e6. From table I, we can see that the proposed algorithm SACC is very effective for fully separable functions f1-f3, and the results for partially additively separable functions f4-f11 and overlapping functions f12-f15 are far from their real global optimal values, this may be caused by several reasons. Firstly, the population may be easy to run into premature. Secondly, because some of the problems are not differentiable, if the selected parameters are not appropriate that may lead to the failure of the local search and the filled function. Thirdly, f12-f14 are overlapping functions and f15 is nonseparable function, which may make the grouping strategy in SACC do not work. Finally, the number of local minima grows exponentially as the number of decision variables increases, so that the computation cost (the number of function evaluations) also increases greatly.

Table II shows a comparison result of SACC with a recently proposed DECC-G [4] on CEC'2013 benchmark test functions, where Q is the quality of the 25 runs are recorded when $FEs = 3.0e6$, P is the test problems, V is the fitness values of problems, and A is the comparison algorithms. The results of the two algorithms were recorded under the same MaxFEs, where bold text means that the proposed algorithm is superior to DECC-G. From table II, we can see that there are 9 results of SACC are better than the results of DECC-G showed in [4]. Among them, the results of f1-f3 show that SACC is effective to fully separable functions. Only three functions in f4-f11 are better than DECC-G which show that SACC is worse than DECC-G. Unexpectedly, the results of f12-f15 by SACC are much better than that by DECC-G.

Fig.2 to Fig.7 show that the convergence curves of the six selected functions: f2, f7, f11, f12, f13, and f14. For each selected function, a single convergence curve be plotted using the average results over all 25 runs. From Fig.2 to Fig.7, we can see that the fitness values of the function decrease rapidly at the beginning of the evolution. This indicates that SACC is effective at the beginning of the evolution. However, at the later evolution process, the evolution become slower and the global optimal solutions have not been found for Fig.2 to Fig.7. The reason may be when the population runs into premature, because of the complexity of problems, the efficiency of the local search and the filled function is likely to become bad.

Overall, whether the data in tabel I, II, or figuers 2-7 shows that the proposed algorithm SACC is effective.

VII. CONCLUSION

The goal of this paper was to investigate a novel algorithm framework for global optimization. It was successfully achieved by developing SACC, which integrated a smoothing function and a novel proposed auxiliary function into a cooperative coevolution algorithm for global optimization. The new algorithm SACC not only could eliminate many local optimal solutions and escape from the local optima, but also could parallel search in the feasible domain.

In order to test the performance of the proposed algorithm SACC, a set of experiments were made on 15 standard benchmark problems in CEC2013 [14]. The results indicated the proposed algorithm was effective.

There are several relevant issues need to be further studied in the future. Firstly, design such evolutionary algorithms suitable to auxiliary functions. Secondly, design self-adaptive mechanism for the various evolution operators. Thirdly, design better auxiliary functions and study the efficient way to integrate the auxiliary function with evolutionary algorithms. At last, design new techniques to enhance EAs.

ACKNOWLEDGMENT

This work is supported by The National Natural Science Foundation of China (No. 61272119).

REFERENCES

[1] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," IEEE Transactions on Evolutionary Computation, vol. 16, pp. 210-224, April 2012.

TABLE I. EXPERIMENTS RESULTS FOR F1-F15

1000D		f1	f2	f3	f4	f5
1.2e5	Best	6.43e+03	3.36e+03	8.71e+00	1.21e+11	6.65e+06
	Median	1.25e+05	4.23e+03	1.07e+01	1.66e+12	1.32e+07
	Worst	2.90e+07	1.44e+04	1.29e+01	5.11e+12	2.31e+07
	Mean	2.76e+06	5.15e+03	1.08e+01	1.60e+12	1.37e+07
	Std	7.39e+06	2.67e+03	1.16e+00	1.15e+12	5.12e+06
6.0e5	Best	1.45e-21	3.78e+02	3.59e-04	3.32e+10	4.02e+06
	Median	9.87e-14	1.42e+03	4.46e+00	1.54e+11	1.16e+07
	Worst	1.32e-09	7.58e+03	1.03e+01	6.08e+11	2.03e+07
	Mean	1.82e-10	2.19e+03	4.95e+00	1.77e+11	1.19e+07
	Std	3.98e-10	1.80e+03	2.98e+00	1.37e+11	5.07e+06
3.0e6	Best	0.00e+00	2.88e+02	9.24e-14	8.48e+09	3.36e+06
	Median	0.00e+00	5.71e+02	1.21e+00	3.66e+10	6.95e+06
	Worst	6.81e-23	2.72e+03	3.76e+00	1.71e+11	1.40e+07
	Mean	2.73e-24	7.06e+02	1.11e+00	4.56e+10	7.74e+06
	Std	1.36e-23	4.72e+02	1.11e+00	3.60E+10	3.22e+06
1000D		f6	f7	f8	f9(905D)	f10
1.2e5	Best	2.45e+05	1.12e+09	6.18e+14	4.08e+08	1.92e+07
	Median	9.60e+05	5.26e+09	4.94e+16	8.75e+08	8.50e+07
	Worst	1.04e+06	1.38e+10	1.67e+17	1.72e+09	9.09e+07
	Mean	7.72e+05	5.75e+09	5.13e+16	8.98e+08	7.08e+07
	Std	2.97e+05	3.86e+09	4.67e+16	3.04e+08	2.59e+07
6.0e5	Best	1.86e+05	1.58e+08	4.34e+14	2.88e+08	1.46e+07
	Median	2.92e+05	1.13e+09	2.35e+15	7.37e+08	7.67e+07
	Worst	9.86e+05	3.04e+09	1.51e+16	1.60e+09	8.99e+07
	Mean	4.23e+05	1.21e+09	3.88e+15	7.69e+08	5.84e+07
	Std	2.53e+05	8.18e+08	3.59e+15	3.08e+08	2.97e+07
3.0e6	Best	1.57e+05	1.72e+06	1.47e+14	2.29e+08	1.38e+07
	Median	2.07e+05	1.58e+07	9.86e+14	5.77e+08	2.11e+07
	Worst	6.00e+05	1.18e+09	3.08e+15	1.01e+09	7.75e+07
	Mean	2.47e+05	8.98e+07	1.20e+15	5.98e+08	2.95e+07
	Std	1.02e+05	2.48e+08	7.63e+14	2.03e+08	1.93e+07
1000D		f11	f12	f13	f14	f15
1.2e5	Best	2.12e+11	8.64e+06	2.85e+10	2.11e+11	4.63e+07
	Median	6.81e+11	1.85e+07	4.42e+10	6.80e+11	3.34e+08
	Worst	2.01e+12	6.02e+09	7.59e+10	1.57e+12	1.50e+09
	Mean	8.06e+11	6.48e+08	4.69e+10	7.47e+11	4.28e+08
	Std	5.47e+11	1.46e+09	1.24e+10	3.86e+11	3.52e+08
6.0e5	Best	2.38e+10	1.96e+03	9.74e+09	6.17e+10	8.21e+06
	Median	1.19e+11	3.08e+03	1.59e+10	1.40e+11	1.43e+07
	Worst	5.81e+11	1.10e+04	2.70e+10	4.28e+11	2.11e+07
	Mean	1.57e+11	3.62e+03	1.63e+10	1.79e+11	1.46e+07
	Std	1.33e+11	2.13e+03	4.27e+09	1.04e+11	3.16e+06
3.0e6	Best	8.12e+07	2.43e+02	6.72e+08	8.21e+07	1.26e+06
	Median	5.30e+08	8.74e+02	1.51e+09	7.34e+09	1.88e+06
	Worst	2.30e+10	1.72e+03	3.40e+09	1.10e+11	4.90e+06
	Mean	2.78e+09	8.73e+02	1.78e+09	1.75e+10	2.01e+06
	Std	5.90e+09	3.71e+02	8.05e+08	2.87e+10	7.23e+05

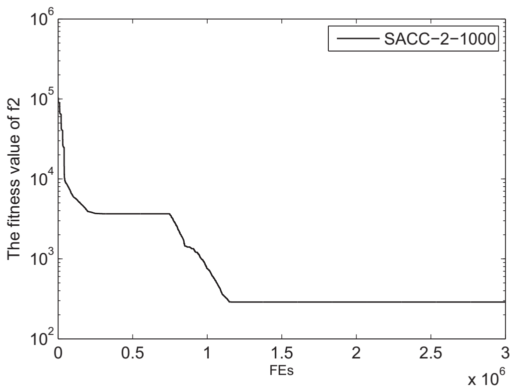

Fig. 2. The convergence curve of SACC on f2

[2] M. N. Omidvar, X. Li, Z. Yang, and X. Yao, "Cooperative coevolution for large scale optimization through more frequent random grouping," in Proc. IEEE Congress on Evolutionary Computation, 2010, pp. 1754-1761.

[3] Mohammad Nabi Omidvar, X. Li, and X. Yao, "Cooperative co-evolution with delta grouping for large scale non-separable function optimization," in Proc. IEEE Congress on Evolutionary Computation, 2010, pp. 1762-

TABLE II. COMPARISON BETWEEN SACC AND DECC-G ON 1000-D FUNCTIONS

P	A V Q	SACC	DECC-G
f1	Best	**0.00e+00**	1.75e-13
	Median	**0.00e+00**	2.00e-13
	Worst	**6.81e−23**	2.45e-13
	Mean	**2.73e−24**	2.03e-13
	Std	**1.36e−23**	1.78e-14
f2	Best	**2.88e+02**	9.90e+02
	Median	**5.71e+02**	1.03e+03
	Worst	2.72e+03	1.07e+03
	Mean	**7.06e+02**	1.03e+03
	Std	4.72e+02	2.26e+01
f3	Best	**9.24e−14**	2.63e-10
	Median	1.21e+00	2.85e-10
	Worst	3.76e+00	3.16e-10
	Mean	1.11e+00	2.87e-10
	Std	1.11e+00	1.38e-11
f4	Best	8.48e+09	7.58e+09
	Median	3.66e+10	2.12e+10
	Worst	1.71e+11	6.99e+10
	Mean	4.56e+10	2.60e+10
	Std	3.60E+10	1.47e+10
f5	Best	**3.36e+06**	7.28e+14
	Median	**6.95e+06**	7.28e+14
	Worst	**1.40e+07**	7.28e+14
	Mean	**7.74e+06**	7.28e+14
	Std	3.22e+06	1.51e+05
f6	Best	1.57e+05	6.96e-08
	Median	2.07e+05	6.08e+04
	Worst	6.00e+05	1.10e+05
	Mean	2.47e+05	4.85e+04
	Std	1.02e+05	3.98e+04
f7	Best	**1.72e+06**	1.96e+08
	Median	**1.58e+07**	4.27e+08
	Worst	**1.18e+09**	1.78e+09
	Mean	**8.98e+07**	6.07e+08
	Std	**2.48e+08**	4.09e+08
f8	Best	1.47e+14	1.43e+14
	Median	9.86e+14	3.88e+14
	Worst	3.08e+15	7.75e+14
	Mean	1.20e+15	4.26e+14
	Std	7.63e+14	1.53e+14
f9(905D)	Best	2.29e+08	2.20e+08
	Median	5.77e+08	4.17e+08
	Worst	1.01e+09	6.55e+08
	Mean	5.98e+08	4.27e+08
	Std	2.03e+08	9.89e+07
f10	Best	1.38e+07	9.29e+04
	Median	2.11e+07	1.19e+07
	Worst	7.75e+07	1.73e+07
	Mean	2.95e+07	1.10e+07
	Std	1.93e+07	4.00e+06
f11	Best	**8.12e+07**	4.68e+10
	Median	**5.30e+08**	1.60e+11
	Worst	**2.30e+10**	7.16e+11
	Mean	**2.78e+09**	2.46e+11
	Std	**5.90e+09**	2.03e+11
f12	Best	**2.43e+02**	9.80e+02
	Median	**8.74e+02**	1.03e+03
	Worst	1.72e+03	1.20e+03
	Mean	**8.73e+02**	1.04e+03
	Std	3.71e+02	5.76e+01
f13	Best	**6.72e+08**	2.09e+10
	Median	**1.51e+09**	3.36e+10
	Worst	**3.40e+09**	4.64e+10
	Mean	**1.78e+09**	3.42e+10
	Std	**8.05e+08**	6.41e+09
f14	Best	**8.21e+07**	1.91e+11
	Median	**7.34e+09**	6.27e+11
	Worst	**1.10e+11**	1.04e+12
	Mean	**1.75e+10**	6.08e+11
	Std	**2.87e+10**	2.06e+11
f15	Best	**1.26e+06**	4.63e+07
	Median	**1.88e+06**	6.01e+07
	Worst	**4.90e+06**	7.15e+07
	Mean	**2.01e+06**	6.05e+07
	Std	**7.23e+05**	6.45e+06

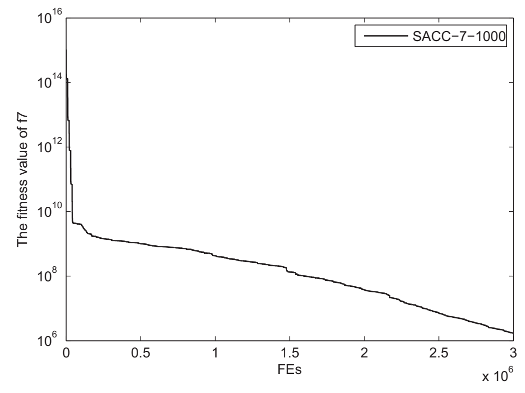

Fig. 3. The convergence curve of SACC on f7

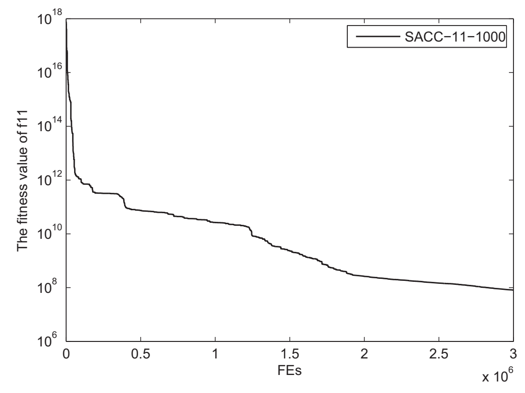

Fig. 4. The convergence curve of SACC on f11

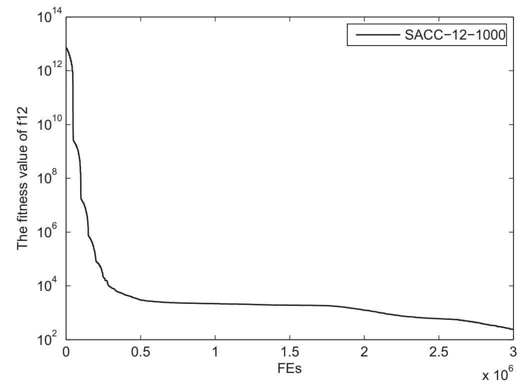

Fig. 5. The convergence curve of SACC on f12

1769.

[4] Z. Yang, K. Tang, and X. Yao, ”Large scale evolutionary optimization using cooperative coevolution,” Information Sciences, vol. 178, pp. 2986-2999, August 2008.

[5] Mitchell A. Potter and Kenneth A. De Jong, ”A cooperative coevolutionary approach to function optimization,” in Proc. International Conference on Parallel Problem Solving from Nature, vol. 2, 1994, pp. 249-257.

[6] Z. Yang, K. Tang, and X. Yao, ”Multilevel cooperative coevolution for large scale optimization,” in Proc. IEEE Congress on Evolutionary Computation, 2008, pp. 1663-1670.

[7] Z. Wu and N. Huang, ”A study of the characteristics of white noise using

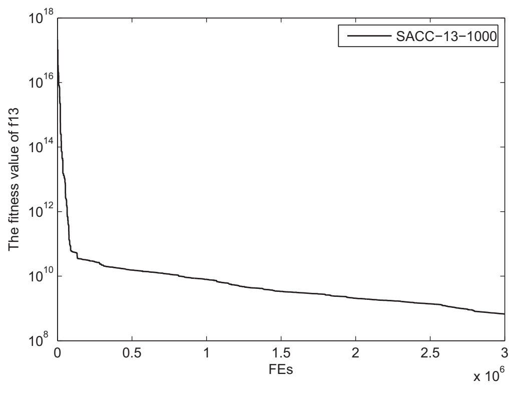

Fig. 6. The convergence curve of SACC on f13

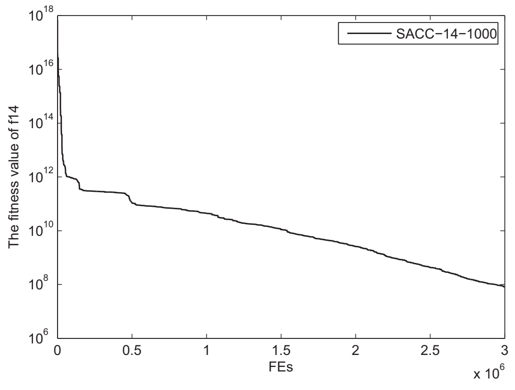

Fig. 7. The convergence curve of SACC on f14

the empirical mode decomposition method,” in Proc. the royal society A: mathematical physical engineering sciences, vol. 460, 2004, pp. 1597-1611.

[8] R. Rach, J. S. Duan, ”Near-field and far-field approximations by the Adomian and asymptotic decomposition methods,” Applied Mathematics and Computation, vol. 217, pp. 5910-5922, 2011.

[9] S. Ivvan Valdez, Arturo Hernndez, Salvador Botello, ”A Boltzmann based estimation of distribution algorithm,” Information Sciences, vol. 236, pp. 126-137, July 2013.

[10] L. Wang, C. Fang, ”A hybrid estimation of distribution algorithm for solving the resource-constrained project scheduling problem,” Expert Systems with Applications, vol. 39, pp. 2451-2460, February 2012.

[11] Chang Wook Ahn, Jinung An, Jae-Chern Yoo, ”Estimation of particle swarm distribution algorithms: Combining the benefits of PSO and EDAs,” Information Sciences, vol. 192, pp. 109-119, June 2012.

[12] G. Iacca, F. Neri, E. Mininno, Y. S. Ong, and M. H. Lim, ”Ockham’s Razor in memetic computing: three stage optimal memetic exploration,” Information Sciences, Elsevier, vol. 188, pp. 17-43, 2012.

[13] F. Caraffini, F. Neri, G. Iacca, and A. Mol, ”Parallel memetic structures,” Information Sciences, Elsevier, vol. 227, pp. 60-82, 2013.

[14] X. Li, K. Tang, M. Omidvar, Z. Yang and K. Qin, ”Benchmark Functions for the CEC’2013 Special Session and Competition on Large Scale Global Optimization,” Technical Report, Evolutionary Computation and Machine Learning Group, RMIT University, Australia, 2013.

[15] Y. P. Wang and D. L. Liu, ”A global optimization evolutionary algorithm and its convergence based on a smooth scheme and line search,” Chinese Journal of Computers, vol. 29, no. 4, pp. 670-675, 2006.

[16] J. Nocedal and S. J. Wright, Numerical Optimization, New York: Springer-Verlag, 1999.