Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

A Robust Evolutionary Algorithm for Large Scale Optimization

S.S. Poorjandaghi. A. Afshar*

Electrical Engineering Department, Amirkabir University of Technology, Tehran, Iran
** Corresponding author, Tel:9821-64543391; e-mail: aafshar@aut.ac.ir*

Abstract: In this paper we present a new cooperative coevolution hybrid Taguchi-genetic algorithm (CCHTGA) to solve high-dimensional optimization problems with continuous variables which are typical in large scale systems. The CCHTGA employs cooperative coevolution frame to decompose the problem into several subsystems and adopts a novel decomposition technique "*random grouping strategy*". Subsystem optimizer is an improved hybrid Taguchi-genetic algorithm (HTGA) which is effective, more robust and rapidly convergent. The performance of the CCHTGA compared to a state-of-the-art cooperative coevolution particle swarm algorithm (CCPSO2). The simulation results obtained from the application of this method to function optimization indicate that CCHTGA can be a competitive algorithm to achieve better and robust results.

Keywords: Cooperative coevolution, Evolutionary algorithms, Large-scale optimization, Random grouping, Taguchi method.

1. INTRODUCTION

Hybrid Taguchi-genetic algorithm (HTGA) is a robust, fast convergent and statistically sound evolutionary algorithm proposed by Tsai et al. (2004) which is shown to be an efficient algorithm to solve 100 dimension problems. In the proposed HTGA, the Taguchi method is combined in the genetic crossover operation to generate better offspring and give more robust results.

In recent years, large scale optimization has received significant attention in evolutionary computation domain. The performance of many known optimization algorithms deteriorates as the number of decision variables increases. On the other hand, many real-world problems require optimizing large number of decision variables. Thus, new optimization algorithms should be capable of facing the challenges of optimization of high-dimensional problems.

Divide-and-conquer (D&V) is a powerful strategy to face high-dimensional optimization problems. Potter and Jong (1994) introduced Cooperative Coevolution (CC) frame which adopts D&V strategy. CC is the most common frame to tackle large scale optimization problems which decompose a problem into several subsystems each evolved by a separate evolutionary algorithm (EA).

The main challenge of CC is how to divide a whole evolutionary system into subsystems since the performance of it is highly dependent on variable interactions. The ideal decomposition strategy should evolve highly interacting variables in the same subsystem while retain the interaction between subsystems as low as possible (Liu et al. 2001). The presence of interactions between decision variables are generally known as *non-separability* (Salomon 1995) or *epistasis* (Klug et al. 2008). The simplest decomposition strategy is to divide a problem into a set of one dimensional problem. However, this strategy does not work well in presence of variable interactions. Two major efforts in collecting interacting variables in the same subsystem are *random Grouping* (Yang et al. 2008) and *Delta Grouping* (Omidvar et al. 2010). Yang shows that by increasing the frequency of random grouping process, the probability of capturing interaction variables in one subsystem increases. In delta grouping if interaction variables lie in different subsystems, the improvement interval of them would be limited. This is the central idea of delta grouping.

The efficiency of an evolutionary algorithm is measured in terms of the number of fitness evaluations needed to discover the optimal solution. This is an important measure of an optimization algorithm, hence fewer fitness evaluation require less CPU time.

Moreover, when running an EA for different runs under completely the same condition, the search trail is normally different for each run due to presence of randomness in nature of an EA. Correspondingly, the final solution of each run using the same EA under the same condition may result in dissimilar optimal solutions. These differences depend on the initial population or any random based evolutionary operator of an EA. An EA is said to be robust if its performance under all types of initial condition is similar. This is an important property for real-world optimization problems. The results of a robust EA are trustable and approximating the required fitness evaluation to find the optimal solution is possible.

In this paper, we present a robust algorithm, the CCHTGA, which utilizes the CC frame to solve large scale optimization problems and random grouping strategy to handle non-separable functions. Furthermore, the improved HTGA algorithm is employed for each subsystem optimizer to enhance the algorithm to be robust and efficient. The motivation of this method is that a solution is not only of high performance, but also of convincing robustness. Robustness of an optimal solution can be seen from the viewpoint that it is insensitive to minor variations of the design variables or environmental parameters. There are two major approaches to improve the robustness of an optimal solution. One is to optimize the expectation of the objective function in a neighbourhood of a target point (Tustsui 1997) and the other is to minimize the variance of the objective function (Das 2000). None of these two approaches can individually guarantee of finding a robust optimal solution. In the first approach, it is possible that positive and negative deviations from the target point cancel each other and therefore a robust solution cannot be found. Alternatively, if the second approach is used, the solution may not be optimal. Thus, it is essential to consider both approaches as an objective. In other words, a trade-off between optimality and robustness should be used. In this paper, we defined a new taguchi criterion to enhance the algorithm in order to find a robust optimal solution.

This paper is organized as follows. In Section 2, the improved HTGA algorithm is introduced. Section 3 presents the CCHTGA. Section 4 demonstrates and analyses the experimental results. Finally Section 5 concludes this paper and present future works.

2. Improved hybrid Taguchi-genetic algorithm

The HTGA combines the traditional GA with the Taguchi method (Tsai 2004). The Taguchi method is based on statistical and sensitivity analysis and widely used in robust design (Phadke 1989). Taguchi used robust parameter design to determine the optimal levels of factors which minimize the sensitivity to noise. Thus, the mean response is close to the desired target while it has minimum variation. Orthogonal array and the signal to noise ratio (SNR) are two major tools used in the Taguchi method. Several SNRs can be defined based on the type of problem characteristics: continues or discrete; nominal-is-the-best, smaller-the-best, or larger-the-best. Further details can be found in (Phadke 1989).

Many designed experiments use matrix structures called orthogonal arrays to distinguish the proper combination of factor levels. An orthogonal array is a fractional factorial matrix, which assures a balanced comparison of levels of any factors or interaction of factors (Tsai et al. 2004). Each rows of orthogonal array matrix represents the level of the factors in each experiment, and each column represents a definite factor. The general symbol for k-level standard orthogonal array is

$$L_n\left(k^{n-1}\right) \tag{1}$$

Where $n = 2^m$ which m is a positive integer greater than 1, k is the number of levels for each factor and $n-1$ in number of columns in the orthogonal array. Additional details of orthogonal arrays can be found in (Phadke 1989).

In HTGA, the two tools of the Taguchi method, two-level orthogonal array and SNR, are employed. Taguchi method is used between crossover and mutation operations. Then, the combination of Taguchi with crossover operation is used to detect better genes to enhance the algorithm.

In this paper, two major modifications are done with respect to the initial version of the HTGA proposed by Tsai (2004). First, we modify the SNR function to handle all types of optimization problems accurately and also define new mutation rule to enhance the search capability of the algorithm. This modified HTGA is then used as a subsystem optimizer in the CC frame. All subsystem optimizers perform individually.

Here, we describe the modified HTGA as follows.

Step 1) Initial population is generated by uniform random distribution within specified range for each variable.

Step 2) Perform one cut-point Crossover operation (Tsai et al. 2004) with probability of crossover rate p_c .

Step 3) Select a suitable two-level orthogonal array for matrix experiments.

Step 4) Choose two chromosomes randomly for Taguchi operation.

Step 5) Calculate the function values and SNRs of n experiments in the orthogonal array $L_n\left(2^{n-1}\right)$ as (2). Each column of $L_n\left(2^{n-1}\right)$ represents a gene in chromosome. Experiments are built based on the fact that each gene of an offspring is belong to whether the first or second randomly chosen chromosome from step 4. That's why the two-level orthogonal array is used.

$$SNR_i = 1 \Big/ \left(z_i - f(\hat{y})\right)^2, i = 1, \ldots, n \tag{2}$$

In (2) z_i is the function value for n th experiment. f is the objective function and $\hat{y}$ is the best chromosome so far. A large SNR value means lower squared deviation. The objective for attaining robust design is to have the highest SNR value. More description around (2) can be found at the end of this section.

Step 6) Calculate the effects of the various factors (E_{f1} and E_{f2}) as follows.

$$E_{fl} = sum\, of\, SNR_l\, for\, factor\, f\, at\, level\, l\ , l = 1, 2$$

Where i is the experiment number, f is the factor name, and l is the level number.

Step 7) One optimal chromosome is generated based on the results from Step 6.

Step 8) Repeat Steps 4 through 7 until the expected $\left(\frac{1}{2}\right) \times M \times p_c$ has been met. In which M is the population size and p_c is the crossover rate.

Step 9) The population via the Taguchi method is generated.

Step 10) Perform mutation operation with probability of p_m. First choose half of chromosome genes randomly, and then mutate these genes as follows.

$$\begin{aligned}
&if\,(p < 0.5)\\
&\qquad x_i = l_i + r \times (u_i - l_i)\\
&else \qquad\qquad\qquad\qquad\qquad\qquad (3)\\
&\qquad x_i = b_i + (r \times 2 - 1) \times |y_i - \hat{y}_i|
\end{aligned}$$

Where p and r are random numbers between 0 and 1, x_i is the ith gene value of chromosome, l_i and u_i are the lower and upper bounds for x_i, $\hat{y}_i$ is the corresponding gene value of best chromosome, and y_i is the ith gene of the best experience of current chromosome which is under mutation process.

Step 11) Offspring population is generated.

Step 12) Selection survivals using roulette wheel approach with stochastic universal sampling.

Two major improvement of the proposed HTGA include modifying SNR function and using new mutation rule, respectively in Step 5 and 9. In (2), consider the term $\left(z_i - f(\hat{y})\right)^2$. We call this term *best-squared deviation (BSD).* For a group of n solutions, if the function values are $z_i = z_1, z_2, ..., z_n$, then the BSD for this group of n solutions can be derived as follows resulting in (4). Let $f(\hat{y}) = m$ and $\alpha = \frac{m}{\overline{z}}$, then

$$\begin{aligned}
BSD &= \frac{1}{n}\left[(z_1 - m)^2 + (z_2 - m)^2 + ... + (z_n - m)^2\right]\\
&= \frac{1}{n}\sum_{i=1}^{n}\left(z_i - 2z_i m + m^2\right)\\
&= \frac{1}{n}\sum_{i=1}^{n}(z_i)^2 - 2\overline{z}m + m^2\\
&= \frac{1}{n}\sum_{i=1}^{n}(z_i)^2 - 2\overline{z}^2 + \overline{z}^2 + (\overline{z} - m)^2\\
&= \frac{1}{n}\sum_{i=1}^{n}(z_i - \overline{z})^2 + (\overline{z} - m)^2 \qquad (4)\\
BSD &= \sigma^2 + (\overline{z} - m)^2 = \sigma^2 + \overline{z}^2(1 - \alpha)^2
\end{aligned}$$

According to (4), the responses at each setting of solutions are treated as a measure that would be indicative of not only the mean of some quality characteristics, but also the variance of those characteristics.

Since the minimization problem is considered in this paper (smaller-the-better measure); the target could be set as zero. Therefore, by setting $f(\hat{y}) = m = 0$ in (4), (5) is obtained.

$$BSD = \sigma^2 + \overline{z}^2 \qquad (5)$$

Equation (5) is related to the $SNR_i = 1/z_i^2$ which is defined in (Tsai et al. 2004). Equation (5) works well for the problems with zero target value, but many real-world problems have different target values. The advantage of using (4) is that one can achieve results in all cases. Besides, this is a self-adaptive characteristic, as the algorithm improves, $f(\hat{y})$ get closer to the optimal value. Therefore, in the beginning of the algorithm, SNR performs more exploration, while at the end it has exploitation function.

In Step (9), two mutation update rule each with probability of p is performed. The value of p can be simply set as 0.5. The first update rule clearly performs exploration, while the second performs exploitation near the best chromosome. By changing the value of p, these two process can be handled through the running of the algorithm, but in this paper it is simply set as 0.5.

3. Cooperative Coevolution Hybrid Taguchi-Genetic Algorithm

In this algorithm, we use CC framework to solve large scale optimization problems. Using this approach the decision variables are divided into several subsystems each of which is optimized with a separate improved HTGA. In order to handle non-separable problems, we use random grouping strategy with more frequent grouping to increase the probability of capturing interacting variables into the same subsystem. Besides, to avoid bound violation, we use a simple mirror method as follows.

$$x_i' = \begin{cases} 2 \times l_i - x_i & : x_i < l_i \\ x_i & l_i < x_i < u_i \\ 2 \times u_i - x_i & x_i > u_i \end{cases} \qquad (6)$$

In which x_i is the decision variable, l_i and u_i are respectively lower and upper bounds of x_i and x_i' is the modified value for x_i. In this method, the search space is doubled in each dimension and reconnected from the opposite bounds to avoid discontinuities, i.e., if a chromosome exceeds the bound, it is returned to the search space in proportion to the violation distance. This approach helps the algorithm to find the optimal solutions which exist near the bound. For two dimensions, this idea is presented in Fig. 1. It's obvious that just infeasible solution is reflected in the feasible space.

The algorithm framework of CCHTGA is described as algorithm 1.

Algorithm 1: Pseudo code of CCHTGA

```
Generate initial population
Arrange the decision variables vector randomly
Divide $n$ decision variables into $K$ subsystems each with $s$ dimension where $s$ is randomly chosen from a set of denominators of $n$ which are divisible to $K$, so $n = K \times s$
for t=1 to maxIterations do
  if $(f(\hat{y}_t) \geq f(\hat{y}_{t-1}))\ or\ (f(\hat{y}_t) - f(\hat{y}_{t-1}) < 0.1 \times f(\hat{y}_{t-1}))$ then
  Perform random Grouping
  for each subsystem $j \in [1...K]$ do
    for each chromosome $i \in [1...pop.size]$ do
      if $f\left(best\left(j, G_j.x_i\right)\right) < f\left(best\left(j, G_j.y_i\right)\right)$ then
        $G_j.y_i \leftarrow G_j.x_i$
      if $f\left(best\left(j, G_j.y_i\right)\right) < f\left(best\left(j, G_j.\hat{y}_t\right)\right)$ then
        $G_j.\hat{y}_t \leftarrow G_j.y_i$
      if $f\left(best\left(j, G_j.\hat{y}_t\right)\right) < f\left(\hat{y}_t\right)$ then
        jth part of $\hat{y}_t \leftarrow G_j.\hat{y}_t$
      Bound handling of $\hat{y}_t$
    end
  end
  for each subsystem $j \in [1..K]$ do
      HTGA operation of $G_j$
  end
end
```

In which n is the number of decision variables, *maxIterations* is the maximum number of iterations, $\hat{y}_t$ is the best chromosome of *t*th iteration and $f(\hat{y}_0) = \max$ real value, *pop.size* is the population size, G_j is the *j*th subsystem, x_i is the *i*th chromosome, y_i is the best experience of x_i, $G_j.x_i$ is the *i*th chromosome of G_j (same meaning for $G_j.y_i$ and $G_j.\hat{y}_t$) and $best(j,z)$ is the function which replaces the *j*th part of $\hat{y}_t$ with z.

The grouping structure is checked for update at the beginning of iteration. The update condition is defined with the purpose of increasing the random grouping frequency. This would help the algorithm to capture interaction variables into the same subsystem.

The first nested loops update the best experience of *i*th chromosome in the *j*th subsystem ($G_j.y_i$) and the subsystem best chromosome ($G_j.\hat{y}$). Besides, the *j*th subsystem best $G_j.\hat{y}$ is used to update the $\hat{y}_t$. In the second loop each subsystem is applied to the improved HTGA operation as subsystem optimizer. The bound handling method is just applied to the $\hat{y}_t$ vector to facilitate population diversity.

Fitness evaluation of *i*th chromosome in *j*th subsystem is calculated as $f\left(best\left(j, G_j.x_i\right)\right)$. This is the *coevolution* part of the algorithm, since the collaboration is made through the vector $\hat{y}_t$. Note that the original dimension array is recorded to restore the actual dimension arrangement when it is necessary.

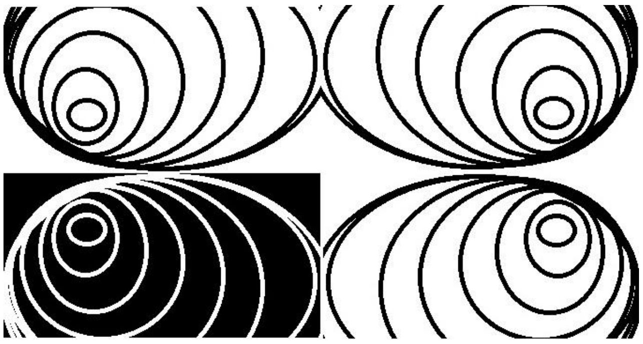

Fig. 1. 2D Bound handling presentation

4. Simulation studies

4.1 Benchmark functions

We use 6 benchmark functions for our simulations which are adopted by CEC 2008 Special session and Competition on

Large Scale Global Optimization (Tang et. al 2007). These functions are presented in Table 1.

The comparison results between CCHTGA and CCPSO2 (Li and Yao 2011) is done on CES'08 functions with D=1,000 and presented in Table 2. Each test function was simulated with 25 independent runs. The mean function value and the standard deviation of the function values are all recorded. The population size was set to 30, the maximum number of fitness evaluations set to $5000 \times n$ (where n is the number of dimensions), the mutation rate is set to 0.7 and the crossover rate is 0.8.

Table 1 Summary of the six CEC'08 test functions

Functions	Modality	Separability
f_1 : Shifted Sphere	Unimodal	Separable
f_2 : Shifted Schwefel	Unimodal	Nonseparable
f_3 : Shifted Rosenbrock	Multimodal	Nonseparable
f_4 : Shifted Rastrigin	Multimodal	Separable
f_5 : Shifted Griewank	Multimodal	Nonseparable
f_6 : Shifted Ackley	Multimodal	Separable

4.2 Results of CEC'08 functions

According to Table 2, the proposed CCHTGA can find optimal or close-to-optimal solutions. The best results among two algorithms are shown in bold. For f_1, f_2, f_3, f_4 and f_5, CCHTGA outperforms the CCPOSO2. Regardless of better result of CCPSO2 on f_6, the mean fitness value of CCHTGA is certainly close to optimal solution. Except for the test function f_6 which is a separable function, the CCHTGA gives smaller standard deviations of function values than the CCPSO2; therefore, the CCHTGA has a more stable solution quality. Moreover, the small differences between mean and standard deviation values in most functions indicate that the CCHTGA is relatively a robust algorithm.

It can be seen from Table 2 that CCHTGA outperforms CCPSO2 on non-separable functions which are more difficult to optimize. Furthermore, its performance on separable functions is quite well. Therefore, the proposed CCHTGA can find the optimal or near-optimal solutions, and give better mean solution quality and more stable solution quality than CCPSO2. Moreover, the comparison results among CCHTGA, MLCC (Yang et al. 2008) and DECC-ML (Omidvar et al. 2010) on CEC'08 functions over 1,000 dimensions are presented in Table 3. The results of MLCC and DECC-ML are based on the literature (Ren and Wu 2013). The best results among the three algorithms are shown in bold. According to Table 3, CCHTGA outperforms the other algorithms on two out of three functions, and regardless of better performance of DECC-DML on f_6, the final results of CCHTGA is close to what is achieved by DECC-DML.

The convergence result of the f_3 test function is shown in Fig. 2. In order to save more space, the convergence results for other test functions are not shown, as they look almost identical. We can see that CCHTGA shows faster convergence behaviour in little iteration compared to CCPSO2. It shows that the CCHTGA has great global search capability that helps the optimizer to find the optimal solution. CCPOS2 tended to converge slower than CCHTGA. This may be attributed to the use of the Cauchy distribution in CCPSO2 to encourage more exploration (Li and Yao 2011). On the other hand, CCHTGA converged particularly fast thanks to exploitation capability of orthogonal arrays which is employed by taguchi method.

Table 2 Results on CEC'08 functions over 1,000 dimensions

Functions	CCHTGA	CCPSO2
f_1	**1.26e-41 (8.71e-37)**	5.18e-13 (5.76e-8)
f_2	**1.30e-03 (6.98e-04)**	7.82e+01 (3.54e+02)
f_3	**1.38e-04 (3.87e-03)**	1.33e+03 (9.17e+04)
f_4	**1.06e-14 (3.91e-15)**	1.99e-01 (5.41e+01)
f_5	**1.11e-16 (8.42e-14)**	1.18e-03 (1.93e-01)
f_6	3.38e-08 (2.98e-08)	**1.02e-12 (9.37e-13)**

Table 3 Results on f_3 , f_4 and f_6 functions over 1,000 dimensions

Functions	CCHTGA	MLCC	DECC-DML
f_3	**1.38e-04 (3.87e-03)**	1.7e+03 (1.80e+02)	9.69e+02 (3.51e+02)
f_4	**1.06e-14 (3.91e-15)**	1.73e-11 (2.21e+00)	1.62E+02 (2.98E+01)
f_6	3.38e-08 (2.98e-08)	1.28e-13 (3.70e-12)	**1.10e-13 (8.22e-15)**

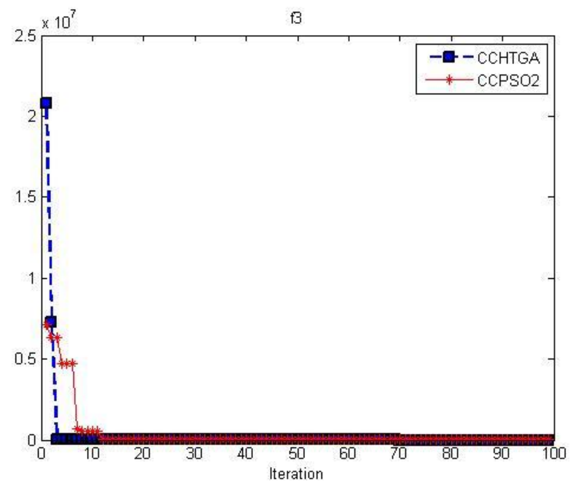

Fig. 2. Convergence plot for f_3

6. CONCLUSIONS

In this paper, we proposed the CCHTGA to improve the performance of the HTGA on large scale optimization problems. The CCHTGA utilizes the CC frame and Taguchi method to enhance the algorithm. Some new techniques have been used to enhance the global search ability of the algorithm. The algorithm employs the random grouping strategy to handle non-separable problems. The CCHTGA was compared with state-of-the-art evolutionary algorithm CCPSO2. The simulation results showed that the CCHTGA outperformed CCPSO2 on five out of six CEC'S 2008 test functions, while it had faster convergence speed. Moreover, the systematic reasoning ability of the Taguchi method is combined into the algorithm to obtain better and more robust results.

In future, we are planning to apply CCHTGA to solve the real-world problems, such as large scale reservoir operation problems. In addition, new decomposition techniques as well as extension of the idea to multi-objective optimization problems will be taken into consideration.

REFERENCES

Phadke, M.S. (1989). Quality engineering using robust design. *Prentice Hall*

Potter, M.A., and De Jong K.S. (1994). A cooperative co-evolutionary approach to function optimization. *Proceedings of the third conference on parallel problem solving nature,* pp 249-257.

Salomon, R. (1995). Re-evaluating genetic algorithm performance under coordinate rotation of benchmark function – a survey of some theoretical and practical aspects of genetic algorithms. *BioSystems,* 39:263-278.

Tustsui, S. (1997). Genetic algorithms with a robust solution searching scheme. *IEEE Transactions on Evolutionary Computation,* 1(3):201-208.

Das, I. (2000). Robustness optimization for constrained nonlinear programming problems. *Engineering Optimization,* 32(5):585-618.

Liu, Y., Yao, X., Zhao, Q. and Higuchi, T. (2001). Scalling up fast evolutionary programming with cooperative coevolution. *Proceeding of Congress on Evolutionary Computation,* pages 1101-1108.

Tsai, J.T., Liu, T.K. and Chou, J.H. (2004). Hybrid tagushi-genetic algorithm for global numerical optimization. *IEEE Trans. Evol. Comput.,* vol. 8, no. 4, pp. 365-377.

Tang k., Yao X., Suganthan PN., MacNish C., Chen YP,. Chen CM., Yang Z. (2007). Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. Tech Rep, Nature Inspired Computation and Application Laboratory, USTC, China.

Klug, W.S., Cummings, M.R., Spencer, C., Spencer, C.A. and Palladino M.A. (2008). *Concepts of Genetics.* Pearson, 9 edition.

Yang, Z., Tang K. and Yao X. (2008). Large scale evolutionary optimization using cooperative coevolution. *Information Science,* 178:2986-2999.

Omidvar M.N., Li, X. and Yao X. (2010). Cooperative co-evolution with delta grouping for large scale non-separable function optimization. *Proceeding of IEEE Congress on Evolutionary Computation (CEC),* pages 1762-1769.

Li, X.D., Yao, X. (2011). Cooperatively coevolving particle swarms for large scale optimization. *IEEE Transaction Evolutionary Computation,* 16(2):210-224.

Ren, Y.F., Wu, Y. (2013). An efficient algorithm for high dimensional function optimization. *Soft Computing,* col1.17, no.6, pp. 995-1004.