8th International Conference on Advanced Computational Intelligence
Chiang Mai, Thailand; February 14-16, 2016

Particle Swarm Optimization with Monte-Carlo Simulation and Hypothesis Testing for Network Reliability Problem

Lu-Yao Wu, Wei-Neng Chen*, Hao-Hui Deng, and Jun Zhang
Key Lab. of Machine Intelligence and Advanced Computing
Ministry of Education, Guangzhou, China
Sun Yat-sen University, Guangzhou, China
Email: cwnraul634@aliyun.com

Yun Li
School of Engineering
University of Glasgow
Glasgow G12 8LT, U.K.

Abstract—**The performance of Monte-Carlo Simulation(MCS) is highly related to the number of simulation. This paper introduces a hypothesis testing technique and incorporated into a Particle Swarm Optimization(PSO) based Monte-Carlo Simulation(MCS) algorithm to solve the complex network reliability problem. The function of hypothesis testing technique is to reduce the dispensable simulation in network system reliability estimation. The proposed technique contains three components: hypothesis testing, network reliability calculation and PSO algorithm for finding solutions. The function of hypothesis testing is to abandon unpromising solutions; we use monte-carlo simulation to obtain network reliability; since the network reliability problem is NP-hard, PSO algorithm is applied. Since the execution time can be better decreased with the decrease of Confidence level of hypothesis testing in a range, but the solution becomes worse when the confidence level exceed a critical value, the experiment are carried out on different confidence levels for finding the critical value. The experimental results show that the proposed method can reduce the computational cost without any loss of its performance under a certain confidence level.**

Keywords*—*Monte-Carlo simulation, network reliability, network reliability optimization, particle swarm optimization, hypothesis testing.

I. Introduction

In this paper, a hypothesis testing method, an effective methodology in statistics, is employed and incorporated into a Particle Swarm Optimization(PSO) based Monte-Carlo Simulation(MCS) algorithm to solve the complex network reliability problem[1]. These algorithm can avoid unnecessary calculation, save processing time and improve query efficiency.

Reliability optimization problem has attracted significant attention in recent years due to the importance of reliability in various kinds of systems[2], [3], [4], [5], [6]. The network reliability optimization problem is to find a balance between cost and reliability.

Complex network reliability problem is different from ordinary network reliability problem, it can be complex and irregular. Monte-Carlo methods can be used to solve any problem having a probabilistic interpretation. They have been used to solve many network reliability problems[1], [7], [8], [9], [10], [11].

The network reliability problem involves more than one constraint and objective and has been proven to be NP-hard. Due to the problem complexity, traditional deterministic algorithms are unable to solve practical large-scale instances in acceptable time.

Evolutionary computation (EC) and swarm intelligence (SI) techniques have gained increasing attention in the past two decades. The advantages of corresponding algorithms are obvious: 1) conceptual simplicity; 2)high efficiency ; 3) flexibility; 4)robustness; 5) having potential to use domain knowledge and to hybridize with other techniques, etc. Particle swarm optimization (PSO) is among the most popular population-based search algorithms in the evolutionary computation community. It is conceptually simple and has shown to be very effective in solving optimization problems[12], [13].

Up to now, various methods have introduced hypothesis testing technique for improving its performance. Wang et al.[14] using hypothesis testing to achieve better results by reducing repeated searches for those solutions with similar performance. The work of Liang et al.[15] shows that hypothesis testing may have better reliability and lower energy consumption than point estimation under certain condition.

Other statistics techniques have been used in the area of Evolutionary Computation(EC) before our work[16]. In this paper, we use hypothesis testing for reducing the calculation of unpromising particles. As a result, the number of function evaluations can be reduced. There are some other works before us also focus on reducing the number of function evaluations. For example, Kim et al. [17] choose K-means algorithm for dividing the whole population into several clusters, and reducing the number of function evaluations by evaluating only one in each clusters; Jin et al. [18] not only introduce cluster algorithm but also ANN techniques for doing so; minimax optimization is another way to achieve the goal[19].

The advantages of our work are summarized as follows: 1) using hypothesis testing technique to abandon unpromising solutions; 2) the addition of heuristic method can make sure the

*Corresponding Author.

This work was supported in part by the NSFC projects Nos. 61379061, 61332002, 6141101191, in part by Natural Science Foundation of Guangdong for Distinguished Young Scholars No. 2015A030306024, in part by the Guangdong Special Support Program No. 2014TQ01X550, and in part by the Guangzhou Pearl River New Star of Science and Technology No. 151700098.

existence of *gbest* while gives a directional guidance for the other particles and keep the components reliabilities as small as possible; 3) the combination of Monte-Carlo simulation and Particle swarm optimization puts forward a new solution in dealing with solving the complex network reliability problem.

The organisation of the remaining contents is as follows. In Section II, it is a brief introduction of the proposed problem and canonical MCS-PSO algorithm. The basic statistics concept of binomial distribution and hypothesis testing is described in Section III. Section IV presents the algorithm in detail. Computational simulation and the setting of parameters are given in Section V, and some conclusions follow in Section VI.

II. MCS-PSO

A. Reliability Optimization Problem

The purpose of this work is to solving Reliability Optimization Problem. Given n components and its corresponding reliability r_i, and each component's cost is decided by r_i and its cost function. We assume that the function of cost and reliability are known in advance. The problem is to find an optimal allocation of reliability in the network in order to minimize the cost under the constraint. The problem can be formulated as

$$\text{minimize} \quad C(\mathbf{r}) \tag{1}$$

subject to

$$R(\mathbf{r}) \geq R^c \tag{2}$$

$$\mathbf{r} = (r_1, r_2, \dots, r_n) \geq \mathbf{r}^c = (r_1^c, r_2^c, \dots, r_n^c). \tag{3}$$

The object of this problem is to minimize the cost while the constraint of reliability must satisfied at the same time. The problem involves more than one constraint and objective and has been proven to be NP-hard.

B. Monte-Carlo Simulation for Network Reliability

Monte-Carlo methods rely on repeated random sampling to obtain numerical results. They are often used in physical and mathematical problems and are most useful when it is difficult or impossible to use other mathematical methods.

Monte-Carlo methods can be used to solve any problem having a probabilistic interpretation. They have been used to solve many network reliability problems[1], [7], [8], [9], [10], [11].

Since the reliability function in our problem cannot be easily obtained by using other mathematical methods, MCS algorithm is applied here to get the approximation. The pseudocode of MCS algorithm can be described as follows.

```
1:  procedure PROBABILITY DEPTH-FIRST SEARCH($a$)
2:      label node $a$ as reached
3:      if $a$ is the target node then
4:          return
5:      end if
6:      for $i \leftarrow 1$ to point number do
7:          if the edge between node $a$ and node $i$ has not been considered, and the reliability of the edge$>0$ then
8:              Generate a random number from uniform$(0,1)$, say k
9:              if $k <$the reliability between node $a$ and node $i$ then
10:                 Procedure PROBABILITY DEPTH-FIRST SEARCH($i$)
11:             end if
12:         end if
13:     end for
14: end procedure

15: procedure IS CONNECTED($R$)
16:     Label every node as not reached
17:     Procedure PROBABILITY DEPTH-FIRST SEARCH($s$)    ▷ $s$ denote the source node
18:     if target node $t$ has been visited then
19:         Return true
20:     end if
21:     Return false
22: end procedure

23: procedure MCS($R$, $M$)
24:     success $=0$, k $=0$
25:     while $k<M$ do
26:         if IS CONNECTED($R$)==true then
27:             success=success+1
28:         end if
29:         k=k+1
30:     end while
31:     Return success/M    ▷ return the estimator of the network
32: end procedure
```

In the implementation of PROBABILITY DEPTH-FIRST SEARCH, the input is the index of current node, say a, it is easier to let a try to run through all nodes and label down each node reached or not.

The input of function IS CONNECTED, R represent an array, each dimension of which is the reliability of every edge. The effect of this function is to decide the network is connected or not by above function.

Next function is MCS, the inputs are R as above and simulation times M. By M times Monte-Carlo Simulation, the system reliability of network can be estimate.

Some statistical characteristics involved in MCS algorithm are summarized as follows.

property 1: The expectation of the estimated reliability value obtained from MCS is an unbiased estimator for the exact reliability R.[20]

property 2: If the margin of error, denoted by ε, is the maximum likely difference (with probability $(1-\alpha)$) between the observed sample proportion $\hat{p}$ and the true value of the population proportion p. The margin of error ε can be found as shown in

$$\varepsilon = Z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{M}} \tag{4}$$

where $\hat{p}(1-\hat{p})$ will always be less than or equal to 0.25. The total number of replications of the simulation must be taken to be at least

$$M \geq \frac{Z_{\alpha/2}^2}{4\varepsilon^2} \tag{5}$$

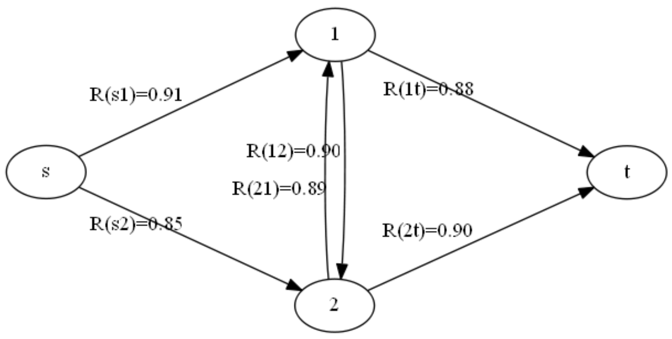

Fig. 1. The illustration network.

The proposed algorithm can be illustrated using the illustration network shown in Fig.1. We assume that each node is perfectly reliable.

Step 0 Let k= 0, success= 0.
Step 1 Go to function IS CONNECTED(R).
Step 2 Label every node as not reached.
Step 3 Go to function PROBABILITY DEPTH-FIRST SEARCH(s).
Step 4 Label the source node s as reached.
Step 5 Choose node 1, generate a random number from uniform$(0, 1)$, say $r^* = 0.32$, and $R(s1) = 0.91 > r^* = 0.32$.
Step 6 Go to function IS CONNECTED(1).
Step 7 Choose node 2, generate a random number from uniform$(0, 1)$, say $r^* = 0.92$, and $R(12) = 0.90 < r^* = 0.92$.
Step 8 Then choose another node t, generate a random number from uniform$(0, 1)$, say $r^* = 0.43$, and $R(1t) = 0.88 > r^* = 0.43$.
Step 9 Return to function IS CONNECTED(R).
Step 10 Function IS CONNECTED(R) return $true$.
Step 11 success = success +1.
Step 12 Next replication.

C. Introduction To PSO

Particle Swarm Optimization (PSO) algorithm is one of the most promising population-based search algorithm for solving optimization problems. PSO was introduced by Kennedy and Eberhart in 1995 for solving optimization problems[21]. In PSO algorithm, each particle keeps track of a position which is the best solution it has achieved so far as $pbest$ and globally optimal solution is stored as $gbest$.

To find the global optimum of the optimization problem, the particles learn from the personal best and global best positions. Specifically, the learning mechanisms in the canonical PSO can be summarized as follows:

$$V_i(t+1) = \omega V_i(t) + c_1 R_1(t)(pbest_i(t) - X_i(t)) + c_2 R_2(t)(gbest(t) - X_i(t)) \quad (6)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (7)$$

where t is the generation number, $V_i(t)$ and $X_i(t)$ represent the velocity and position of the i-th particle, respectively; ω is termed inertia weight, c_1 and c_2 are the acceleration coefficients, $R_1(t)$ and $R_2(t)$ are two vectors randomly generated within $[0, 1]^n$, with n being the dimension of the search space; $pbest_i(t)$ and $gbest(t)$ denote the personal best of the i-th particle and the global best of the swarm, respectively.

III. Hypothesis Testing

A. Binomial Distribution

A Bernoulli experiment is a random experiment, the outcome of which can be classified in but one of two mutually exclusive and exhaustive ways. In this paper, each trial of the network only have two possible answers: success or failure. A sequence of Bernoulli trials occurs when a Bernoulli experiment is performed several independent times so that the probability of success, denoted by p, remains the same from trial to trial. That is, in such sequence, we let p denote the probability of success on each trial.

If the random variable X follows the binomial distribution with n trials and the probability of success p, we write $X \sim B(n, p)$. The probability of getting exactly k successes in n trials is given by the probability mass function:

$$f(k; n, p) = Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k} \quad (8)$$

The cumulative distribution function can be expressed as:

$$F(k; n, p) = Pr(X \leq k) = \sum_{i=0}^{\lfloor k \rfloor} \binom{n}{i} p^i (1-p)^{n-i} \quad (9)$$

B. The Central Limit Theorem

Theorem 1 (The Central Limit Theorem): Let X_1, X_2, ..., X_n denote the observations of a random sample from a distribution that has mean μ and positive variance σ^2. Then the random variable

$$Y = \frac{(\sum_{i=1}^{n} X_i - n\mu)}{\sqrt{n}\sigma} = \frac{\sqrt{n}(\bar{X_n} - \mu)}{\sigma} \quad (10)$$

has a limiting distribution that is normal with mean $zero$ and variance 1.

The proof can be found in many textbooks on probability theory and mathematical statistics [22][23][24].

According to The Central Limit Theorem, we know that $\bar{X}$ and $\sum_{i=1}^{n} X_i$ have approximate normal distributions, provided that n is large enough. Let $X_1, X_2, \ldots, X_n$ denote a random sample from a distribution that is $B(1, p)$. If $Y_n = X_1 + X_2 + \cdots + X_n$, it is known that Y_n is $B(n, p)$. Calculation of probabilities concerning Y_n used to employ formulate (8) and (9) under normal circumstances, but now can be greatly simplified by making use of the fact that

$$\frac{(Y_n - np)}{\sqrt{np(1-p)}} = \frac{\sqrt{n}(\bar{X_n} - p)}{\sqrt{p(1-p)}} = \frac{\sqrt{n}(\bar{X_n} - \mu)}{\sigma} \quad (11)$$

has a limiting distribution that is normal with mean $zero$ and variance 1.

C. Hypothesis Testing

In our problem, a network reliability is evaluated upon several times Monte-Carlo Simulation, and compared against a certain reliability constraint to determine whether hypothesis H_0 or hypothesis H_1 is to be chosen.

Hypothesis H_0 corresponds to the case where the network reliability is not smaller than reliability constraint, and hypothesis H_1 corresponds to the network reliability is smaller than reliability constraint. The Monte-Carlo Simulation terminates on the timing if hypothesis H_1 is chosen, and this particle will never be chosen as $gbest$ in PSO; otherwise, the simulation continues to receive corresponding network system reliability.

We test $H_0 : p \geq R^c$ against the one-sided alternative $H_1 : p < R^c$, where p is probability of success which calculated by n trials, R^c is reliability constraint in our problem. In dealing with above test, at significance level $\alpha = 0.05$, reject H_0 if

$$Z = \frac{p - R^c}{\sqrt{\frac{R^c(1-R^c)}{n}}} < -c \tag{12}$$

where c is the 95th percentile of the normal distribution with mean $zero$ and variance 1.

IV. Description of Proposed Algorithm

This paper aims at improving the canonical MCS-PSO algorithm by reducing the computational cost of Monte-Carlo Simulation. Before the introduction of the main flow of the new algorithm, we import several methods to overcome the drawbacks of this algorithm.

A. Some improvement

1) Find A Feasible Initial Solution: Yeh[1] proposed a heuristic method to find an initial solution at the beginning of algorithm. In this paper, in consideration of it is possible that every particle's hypothesis testing may be rejected in the first generation of canonical PSO, the addition of heuristic method can make sure the existence of $gbest$. The initial solution also gives a directional guidance for the other particles while keep the components reliabilities as small as possible. The following will explain the detail of the heuristic method.

```
1:  procedure HEURISTIC
2:      Assign the lower bound of the component reliability to the first particle.
3:      Apply MCS to decide the corresponding system reliability $R^*(\mathbf{r})$.
4:      while $R^*(\mathbf{r}) < R^c$ do
5:          for $i \leftarrow 1$ to dimension number do
6:              Let the value of dimension $i$ be $0.25 \times (1 - r_i) + r_i$.
7:              Apply MCS to get $R_i^*(\mathbf{r})$.
8:              Adjust dimension to its original reliability.
9:          end for
10:         if $\forall i, R_i^*(\mathbf{r}) < R^c$ then
11:             Assign the dimension with maximal $R_i^*(\mathbf{r})$ value to $j$.
12:         else
13:             Find the minimum cost $C(\mathbf{r})$, assign its index to $j$.
14:         end if
15:         Let the value of dimension $j$ be $0.25 \times (1-r_j)+r_j$, and apply MCS to update $R^*(\mathbf{r})$.
16:     end while
17: end procedure
```

In the implementation of HEURISTIC function, lower bound is the minimum value of each component can be accepted. Assign the lower bound of the each component's reliability to the first particle, and evaluate the reliability of the particle. If it is not big enough, then increase the value of the most resultful component slightly until the reliability of the particle meets our expect.

2) Penalty Function: The addition of penalty function allows the search in the infeasible space, and tends to yield optimum solution more rapidly and produce better final solutions. By penalizing the infeasible solutions, the population can converge at the feasible optimum solution after several generations. C_i denote the total system cost of particle i. If the estimate value of reliability of particle i is smaller than R^c, a penalized cost will calculated by

$$C_i(r) \times \left(\frac{R^c}{R^*(r)}\right)^{\lambda} \text{ when } R^*(\mathbf{r}) < R^c \tag{13}$$

where λ is an amplification parameter[25]. The penalty function is decided by both the cost function $C_i(r)$ and the ratio of R^c in to $R^*(r)$.

B. The Main Flow of The Proposed Algorithm

The explanations of the proposed MCS-PSO method are as follows.

Step 0 Let Interation_Number= 0.

Step 1 Using heuristic method to obtain the first particle and the other particles are initialized randomly.

Step 2 Apply hypothesis test to each particle.

1) If the particle does not pass the hypothesis test, then it will not be chosen as $gbest$ and the cost of this particle will assign as infinite. Go back to Step 2 for the next particle.
2) Else if null hypothesis H_0 was not been reject, then apply MCS to find corresponding $R^*(\mathbf{r})$. The total cost is calculated by specific corresponding component reliability. If $R^*(\mathbf{r}) < R^c$, the cost will amplify by penalty function (13).

Step 3 Update the pbest, gbest, velocity, and position of each particle based on function (6) and (7). Interation_Number=Interation_Number+1.

Step 4 If the maximum Iteration_Number is reached, then stop; else go to Step 2.

V. Experimental Results and Summary

A. Experimental setting

In our experiment, networks are taken from the resource constrained project scheduling problem (RCPSP)[26]. As test instances we have employed the standard set j30 for solving the complex network reliability problem, shown in Fig.2. The instance have 48 components.

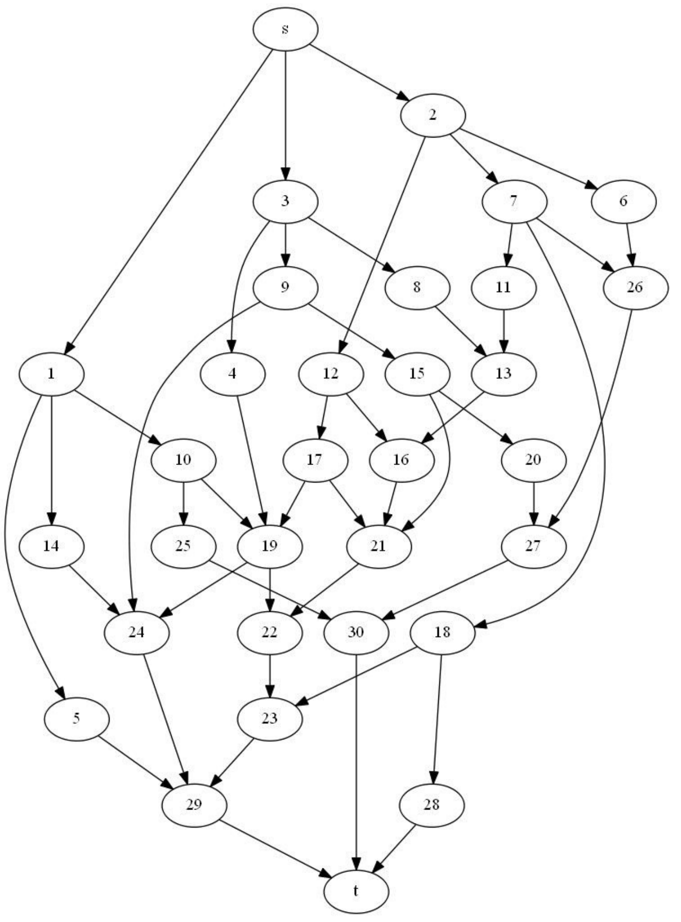

Fig. 2. The j30 network.

TABLE I. THE SETTING OF HTMCS-PSO

Number of particle	80
Number of dimension	48
Max position	1
Min position	0.6
Max velocity	0.4
Min velocity	-0.4
Iteration	100
Cognitive factor	0.8
Social factor	0.8
Inertia weight ω	0.9→0.4
Heuristic simulation	5000
Simulation	1000
Hypothesis testing simulation	30
λ	10

TABLE II. THE COST FUNCTION OF EACH COMPONENT

Component i	α_i	β_i	$C(r_i) = \alpha_i - \beta_i \times ln(1 - r_i)$
$i\%9 = 1$	120	14.7	$120 - 14.7 \times ln(1 - r_i)$
$i\%9 = 2$	120	14.7	$120 - 14.7 \times ln(1 - r_i)$
$i\%9 = 3$	90	8.75	$90 - 8.75 \times ln(1 - r_i)$
$i\%9 = 4$	100	9.9	$100 - 9.9 \times ln(1 - r_i)$
$i\%9 = 5$	65	5.64	$65 - 5,64 \times ln(1 - r_i)$
$i\%9 = 6$	100	9.9	$100 - 9.9 \times ln(1 - r_i)$
$i\%9 = 7$	90	8.75	$90 - 8.75 \times ln(1 - r_i)$
$i\%9 = 8$	160	12	$160 - 12 \times ln(1 - r_i)$
$i\%9 = 0$	160	12	$160 - 12 \times ln(1 - r_i)$

The corresponding numerical parameters and data are presented in Table I for comparing the performance between standard MCS-PSO and proposed HTMCS-PSO. We use 80 particles, and each particle with 48 dimensions. The reliability is in the interval $[0, 1]$, so the maximum position is set to 1, while the minimum position is no less than lower bound 0.6. The maximum velocity is determined by the distance between maximum position and minimum position. Cognitive factor and the social factor are both set to 0.8. Inertia weight ω started with a value 0.9 and linearly decreased to 0.4 when the iteration number reached 100[27]. We apply hypothesis testing method with 30 replications of Monte-Carlo Simulations to decide if one particle is promising solution. The promising particle will finish 1000 replications to get $R^*(\mathbf{r})$. If the simulation result $R^*(\mathbf{r})$ does not satisfy the lower bound of the constraint R^c, the penalty function (13) with $\lambda = 10$ will be applied. Otherwise, the cost will be determined by specific corresponding component reliability only. We assume that the function of cost and reliability are known in advance. Table II has listed the cost function of each component.

B. Experiment result analysis

Table III shows the results of the proposed HTMCS-PSO with different confidence level. Different confidence level means different stringency in hypothesis testing. When confidence level is small, solutions are more likely to be abandoned. This will result in less computational cost, but may missing the best solution as well. When confidence level is large(close to 1), the hypothesis testing is harder to be rejected and the best solution has less opportunity to be discarded. When confidence level is equal to 1, the proposed algorithm is exactly same as standard MCS-PSO.

We are interested in the effect of the setting of confidence level on the final solution. To prove that the proposed HTMCS-PSO efficiently reduces computational cost without any loss of the performance under a certain confidence level, we repeated each test 30 times in the testing experiments which can be regarded as large sample test.

In the experiment, we performed two different hypothesis testing to analyse the performance of HTMCS-PSO method:

1) Performance test: The first hypothesis testing to determine whether the solutions using the HTMCS-PSO will performance worse than the standard MCS-PSO.

Let μ_{ht} be an average solution of HTMCS-PSO with a special confidence level, as in Table III, and μ be an average solution of standard MCS-PSO. The test is then

$$\alpha{=}0.05$$
$$H_0 : \mu_{ht} \geq \mu$$
$$H_1 : \mu_{ht} < \mu.$$

We use a t-test to test these two populations. From Table III, we find that when Confidence Level is equal to 0.8, P-value was calculated less than 0.05, then reject H_0, which means

TABLE III. Solutions for the Proposed HTMCS-PSO With Different Confidence Level

Replication	Confidence Level									
	0.8		0.9		0.95		0.99		MCS-PSO	
	cost	FEs	cost	FEs	cost	FEs	cost	FEs	cost	FEs
1	5982.33	5394	5982.2	7007	5980.59	6792	5974.87	7512	5974.87	8000
2	5974.17	5496	5998.46	6896	5978.4	6980	5965.04	7569	5979.22	8000
3	5992.95	5674	6004.57	6828	5970.13	6843	5977.09	7579	5982.44	8000
4	5977.19	5672	5987.79	6901	5967.55	7028	5964.41	7511	5960.31	8000
5	5977.21	5580	5971.89	6900	5963.88	6993	5987.36	7517	5965.83	8000
6	5972.31	5660	5979.43	6932	5978.11	6853	5990	7485	5981.11	8000
7	5984.48	5674	5982.86	6860	5985.12	6925	5980.26	7593	5965.14	8000
8	5981.49	5717	5983.74	6931	5962.14	6932	5971.67	7523	5970.94	8000
9	5970.54	5298	5972.44	6887	5978.75	6889	5975.93	7585	5970.02	8000
10	5985.89	5533	5979.52	6988	5963	6923	5988.23	7607	5968.05	8000
11	5986.69	5659	5964.32	6878	5988.86	6961	5977.38	7635	5979.76	8000
12	5983.81	5580	5985.32	6967	5981.65	6781	5986.7	7525	5980.06	8000
13	5973.95	5683	5988.32	6917	5974.46	6792	5978.57	7589	5972.85	8000
14	5988.7	5676	5983.5	6974	5967.16	6870	5986.88	7527	5990.25	8000
15	5965.51	5538	5965.62	7025	5966.92	6995	5988.22	7592	5986.97	8000
16	5995.66	5681	5970.94	6935	5963.4	6968	5965.03	7524	5977.29	8000
17	5984.94	5718	5993.4	7017	5970.87	6994	5993.22	7447	5985.9	8000
18	5994.73	5680	5994.95	6889	5976.01	6915	5966.29	7576	5982.83	8000
19	5989.81	5691	5969.21	6849	5998.06	6885	5963.46	7540	5994.5	8000
20	5975.41	5608	5999.5	7041	6000.48	6856	5961.37	7614	5970.26	8000
21	5980.21	5725	5962.59	6947	5998.9	6987	5975.92	7586	5974.17	8000
22	5982.72	5571	5971.09	6863	5986.79	6965	5988.18	7580	5978.91	8000
23	6009.33	5415	5987.41	6708	5967.82	6866	5957.43	7537	6013.71	8000
24	5977.83	5713	5976.84	6933	5970.08	6767	5976.97	7604	5983.66	8000
25	5999.26	5592	5976.86	6795	5976.69	6945	5976.52	7580	5994.06	8000
26	5977.48	5561	5970.45	6901	5978.61	6939	5966.94	7563	5972.26	8000
27	5994.16	5554	5987.81	6880	5968.52	6922	5978.22	7503	5975.96	8000
28	5992.28	5492	5963.04	6815	5981.95	6880	5977.54	7587	5981.73	8000
29	5991.53	5676	5976.34	6894	5982.27	6963	5966.41	7586	5973.55	8000
30	5986.81	5497	5982.61	6916	5981.97	6932	5974.97	7555	5989.04	8000
Best	5965.51	5298	5962.59	6708	5962.14	6767	5957.43	7447	5960.31	8000
Average	5984.312667	5600.266667	5980.434	6909.133333	5976.971333	6911.366667	5976.036	7557.7	5979.188333	8000
Variance	90.90339264	11503.37471	124.53508	5172.395402	112.2486326	4908.86092	93.73694207	1856.010345	113.5659385	0
P-value	0.02726337	3.14749E-41	0.330009782	2.39927E-36	0.882313723	1.19614E-36	0.78882453	1.84276E-31	1	1

a lot of promising particles have been abandoned and it is less likely to get good results. What's more, when Confidence Level $\geq$ 0.9, P-value tends to be larger than 0.05, then cannot reject H_0, it also corresponds to our assumption that the proper choose of confidence level will not affect the efficiency in finding solutions but can significantly reduce the calculation times.

2) Efficiency test: The next hypothesis testing is aiming at judge if the proposed algorithm has significantly reduced the Fitness Evaluation times(FEs) or not.

Let μ_{FEs} be an average FEs of HTMCS-PSO with a special confidence level, as in Table III, and $\mu_{FEs=8000}$ be an average FEs of standard MCS-PSO. The test is then

$$\begin{aligned} \alpha &= 0.05 \\ H_0 &: \mu_{FEs} \geq \mu_{FEs=8000} \\ H_1 &: \mu_{FEs} < \mu_{FEs=8000}. \end{aligned}$$

We use a t-test to test these two populations. As can be seen in Table III, p-values of all hypothesis testing are far more less than 0.05, which means the effect of the proposed algorithm is remarkable. Even in the situation that confidence level is set to be as big as 0.99, FEs has been decreased by 6.9%.

In summary of above experiment, from Table III, we find that the results with proper confidence level have almost same performance as original MCS-PSO, while have reduced computational cost by about 5 to 10 percent. But when confidence level equal to 0.8 or so, solutions are easily fall into rejection region, which means promising solutions are also more likely to be rejected. Confidence level is an important parameter in the proposed algorithm, the proper choose of it can significantly reduce calculation, but an improper one can also make solution worse. There is no clear division of proper and improper one, experiments should be apply to pick an useful one.

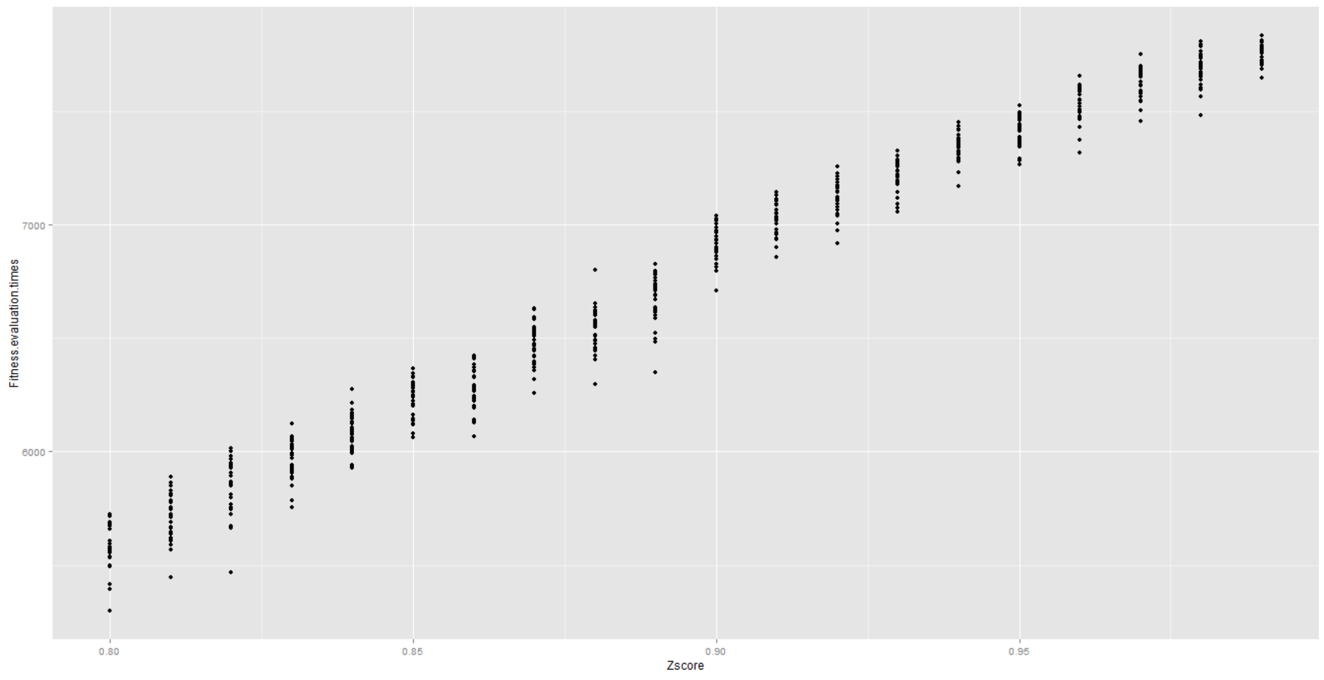

Fig. 3. The tendency of FEs with different confidence levels.

C. The relation between FEs and confidence levels

We are also interested in the relation between FEs and confidence levels. We pick 21 different confidence levels, with each confidence level we repeated 30 times experiment to inspect the tendency of FEs.

Fig. 3. shows the tendency of FEs with different confidence level. When confidence level is equal to 1, the value of FEs must be 8000, which consist of 80 particles times 100 generations. When confidence level is equal to 0.8, the mean of FEs is almost equal to 5600 which is far more less that the original FEs 8000. FEs increase with confidence level, which also meets our common knowledge and experience.

VI. Conclusion

We have proposed an efficient HTMCS-PSO with less fitness evaluation by hypothesis testing. It evaluate particles by a small amount of Monte-Carlo Simulation, and abandon unpromising solutions for saving processing time and improve query efficiency. This hybrid MCS-PSO with hypothesis testing can efficiently reduces the evaluation times without any loss of the performance under a certain confidence level. Results from several experiments show that the algorithm has almost same performance to original MCS-PSO that evaluates far more times than the proposed HTMCS-PSO.

Such a hypothesis testing technique is very useful for problems that require high cost to evaluate individuals. We hope that the proposed hypothesis testing technique could also be extended to solve various kinds of problems.

References

[1] W.-C. Yeh, Y.-C. Lin, Y. Y. Chung, and M. Chih, "A Particle Swarm Optimization Approach Based on Monte Carlo Simulation for Solving the Complex Network Reliability Problem," IEEE Transactions on Reliability, vol. 59, no. 1, pp. 212–221, 2010.

[2] T. Aven, "Availability evaluation of oil/gas production and transportation systems," Reliability engineering, vol. 18, no. 1, pp. 35–44, 1987. [Online]. Available: http://dx.doi.org/10.1016/0143-8174(87)90050-3

[3] K. Aggarwal, J. Gupta, and K. Misra, "A simple method for reliability evaluation of a communication system," Communications, IEEE Transactions on, vol. 23, no. 5, pp. 563–566, 1975.

[4] M. Samad, "An efficient algorithm for simultaneously deducing minimal paths as well as cuts of a communication network," Microelectronics Reliability, vol. 27, no. 3, pp. 437–441, 1987.

[5] W.-J. Ke and S.-D. Wang, "Reliability evaluation for distributed computing networks with imperfect nodes," Reliability, IEEE Transactions on, vol. 46, no. 3, pp. 342–349, 1997.

[6] P. Doulliez and E. Jamoulle, "Transportation networks with random arc capacities," Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle, vol. 6, no. 3, pp. 45–59, 1972.

[7] H. Kumamoto, K. Tanaka, and K. Inoue, "Efficient evaluation of system reliability by Monte Carlo method," Reliability, IEEE Transactions on, vol. 26, no. 5, pp. 311–315, 1977.

[8] W. N. Chen, J. Zhang, O. Liu, and H. L. Liu, "A Monte-Carlo ant colony system for scheduling multi-mode projects with uncertainties to optimize cash flows," 2010 IEEE World Congress on Computational Intelligence, WCCI 2010 - 2010 IEEE Congress on Evolutionary Computation, CEC 2010, 2010.

[9] H. Cancela, F. Robledo, G. Rubino, and P. Sartor, "A Monte Carlo sampling plan for estimating diameter-dependent network parameters," Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2012 4th International Congress on, pp. 766–771, 2012.

[10] W.-C. Yeh, "A MCS-RSM approach for network reliability to minimise the total cost," International Journal of Advanced Manufacturing Technology, vol. 22, no. 9-10, pp. 681–688, 2003.

[11] W.-C. Yeh, C.-H. Lin, and Y.-C. Lin, "A MCS Based Neural Network Approach to Extract Network Approximate Reliability Function," pp. 287–297.

[12] W.-N. Chen, J. Zhang, Y. Lin, N. Chen, Z.-H. Zhan, H.-H. Chung, Y. Li, and Y.-H. Shi, "Particle swarm optimization with an aging leader and

challengers," Evolutionary Computation, IEEE Transactions on, vol. 17, no. 2, pp. 241–258, April 2013.

[13] W.-N. Chen, J. Zhang, H.S.H. Chung, W-L Zhong, W.-G. Wu, and Y.-H. Shi, "A novel set-based particle swarm optimization method for discrete optimization problems," Evolutionary Computation, IEEE Transactions on, vol. 14, no. 2, pp. 278–300, April 2010.

[14] L. Wang, L. Zhang, and D. Z. Zheng, "A class of hypothesis-test-based genetic algorithms for flow shop scheduling with stochastic processing time," International Journal of Advanced Manufacturing Technology, vol. 25, no. 11-12, pp. 1157–1163, 2005.

[15] Y. Liang, D. Rajan, and O. Eliezer, "Sequential Frame Synchronization Based on Hypothesis Testing with Unknown Channel State Information," IEEE Transactions on Communications, vol. 6778, no. c, pp. 1–1, 2015. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7115103

[16] J. Zhang, Z. H. Zhang, Y. Lin, N. Chen, Y. J. Gong, J. H. Zhong, H. S. H. Chung, Y. Li, and Y. H. Shi, "Evolutionary computation meets machine learning: A survey," IEEE Computational Intelligence Magazine, vol. 6, no. 4, pp. 68–75, 2011.

[17] H.-S. Kim and S.-B. Cho, "An efficient genetic algorithm with less fitness evaluation by clustering," Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546), vol. 2, 2001.

[18] Y. Jin and B. Sendhoff, "Reducing fitness evaluations using clustering techniques and neural network ensembles," Genetic and Evolutionary Computation–GECCO 2004, pp. 688–699, 2004.

[19] A. Zhou and Q. Zhang, "A surrogate-assisted evolutionary algorithm for minimax optimization," 2010 IEEE World Congress on Computational Intelligence, WCCI 2010 - 2010 IEEE Congress on Evolutionary Computation, CEC 2010, pp. 0–6, 2010.

[20] G. S. Fishman, "A Monte Carlo Sampling Plan for Estimating Network Reliability," Operations Research, vol. 34, no. 4, pp. 581–594, 1986.

[21] J. Kennedy and R. Eberhart, "Particle swarm optimization," Neural Networks, 1995. Proceedings., IEEE International Conference on, vol. 4, pp. 1942–1948 vol.4, 1995.

[22] M. F. Triola, "Elementary statistics". Pearson/Addison-Wesley Reading, MA, 2006.

[23] P. G. Hoel *et al.*, "Introduction to mathematical statistics." Introduction to mathematical statistics., no. 2nd Ed, 1954.

[24] I. Bárány and V. Vu, "Central limit theorems for gaussian polytopes," Annals of Probability, vol. 35, no. 4, pp. 1593–1621, 2007.

[25] Y.-C. Liang and A. E. Smith, "An ant colony optimization algorithm for the redundancy allocation problem (RAP)," IEEE Transactions on Reliability, vol. 53, no. 3, pp. 417–423, 2004.

[26] R. Kolisch and A. Sprecher, "PSPLIB - A project scheduling problem library," European Journal of Operational Research, vol. 96, no. 1, pp. 205–216, 1997.

[27] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), pp. 69–73, 1998.