Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms.

Conference Paper · January 2005

DOI: 10.1109/CIMCA.2005.1631619 · Source: DBLP

Jinghui Zhong
South China University of Technology
58 PUBLICATIONS **622** CITATIONS
SEE PROFILE

Jun Zhang
Sun Yat-Sen University
213 PUBLICATIONS **5,317** CITATIONS
SEE PROFILE

Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms

Jinghui Zhong[1], Xiaomin Hu[1], Min Gu[2] and Jun Zhang[1] (Corresponding author), *MIEEE*
[1]SUN Yat-sen University, P.R.China, [2]Wuxi Institute of Technology, P.R. China
(This work was supported in part by the National Natural Science Foundation of China No. 60573066 and in part by Guangdong Natural Science Foundation No. 5003346, Guangdong, China.)
junzhang@ieee.org

Abstract

This paper presents the comparison of performance on a simple genetic algorithm (SGA) using roulette wheel selection and tournament selection. A SGA is mainly composed of three genetic operations, which are selection, crossover and mutation. With the same crossover and mutation operation, the simulation results are studied by comparing different selection strategies which are discussed in this paper. Qualitative analysis of the selection strategies is depicted, and the numerical experiments show that SGA with tournament selection strategy converges much faster than roulette wheel selection.

1. Introduction

John Holland pioneered genetic algorithms by simulating evolution of the nature. During the past forty years, genetic algorithms have been applied in many fields such as pattern recognition, robotics, artificial life, experts system, electronic and electrical field, cellular automata, etc [1]-[7].

By using the genetic algorithms to solve a problem, we first present the candidate solutions as a sequence of values, and define an evaluation function to evaluate the candidate solutions. An artificial genetic system uses concepts like population and generation to simulate the natural genetic system. One population consists of a certain number of individuals which serve as candidate solutions. New generation of population is created by genetic operations such as selection, crossover and mutation in iteration.

According to the Darwinian principles of survival, which is called "the survival of the fittest", the excellent individuals have far more chances to adapt themselves to the environment and survive, while the inferior ones die out [8]. The survivals reproduce new individuals with better genes which make the new generation more endurable to the nature. Similarly, the GAs(Genetic Algorithms) use this process of reproduction as a basic genetic operation for the algorithms. Currently there are several selection strategies, such as roulette wheel, tournament selection, rank-based selection and deterministic sampling.

In this paper, roulette wheel selection and tournament selection are adopted in the SGA and a comparison has been made on the results. The results of the experiments show that the algorithm with tournament selection strategy converges much faster than roulette wheel selection

2. Simple genetic algorithm

The SGA is composed of three genetic operations: selection, crossover and mutation [9].The SGA uses the steps as below:

Step1. Encode the given problems in gene strings.
Step2. Create an initial population consists of a certain number of individuals.
Step3. Perform sub-steps as follows until the termination condition is satisfied:
*(a)*Evaluate the fitness value of each individual in the population.
*(b)*Create a new population by three genetic operations as follows:
- According to the fitness value, individuals are chosen with a probability. Replicate the selected ones to form a new population.
- Create two new individuals by two parents who are selected probabilistically from the population and recombine them at the crossover point.
- Create a new individual by mutating an existing individual with the probabilistically selected.

Step4. When the process ceases, we can find a solution from the result of the genetic algorithm.

In the operation of selection, the selection of an individual is based on its fitness value. The better the fitness of the individual is, the larger the probability it is to be chosen. Without any changes, the selected one is replicated into the next generation of the population. In addition, individuals with poor fitness value may be selected because the genetic selection is probabilistic. To implement the operation of selection, strategies such as roulette wheel and tournament selection can be used. Different selection strategies affect the performance of the algorithm differently.

Crossover is performed on two selected individuals at a time. With a probability, two individuals are chosen, and then the crossover point is selected randomly, and the pair of selected individuals undergoes the process of crossover. Two offspring are produced by the operation of crossover. For example, considering two parental individuals represented by two binary strings S1, S2 that S1=0 0 1 1 1 | 0 1 0, S2=1 1 1 0 1 | 0 1 1, they randomly produce a number *5* as the crossover position. After swapping the substrings after the crossover position, we obtain two offspring S1=0 0 1 1 1 | 0 1 1, S2=1 1 1 0 1 | 0 1 0.

The operation of mutation starts with selecting an individual probabilistically from the population. Then a mutation point is chosen at random. The character at mutation point is changed and then the new individual is copied into the next generation of the population. The probability of mutation is usually set to be very small.

3. Selection strategies

A. Roulette Wheel Selection

Roulette wheel selection is the most frequently used selection strategy. It is a proportional selection strategy which has the similar selecting principle as roulette wheel. Fig.1 shows a roulette wheel. The whole roulette is partitioned into several sectors in different area corresponding to different amount of money. When the spun roulette wheel stops, the sector which the pointer points at is chosen and the gamester gets the amount of money corresponding to the sector. Although we can't foresee which sector the pointer will point to, the probability of a sector to be chosen can be evaluated. It is proportional to the magnitude of the central angle of the sector. The bigger the central angle of the sector is, the higher probability the pointer will point at the sector. Similarly, in the genetic algorithms, the whole population is partitioned by the individuals, each sector representing an individual. The proportion of the individual's fitness value to the total fitness values of the whole population decides the area of the sector corresponding to the individual and decides the probability of the individual to be selected into the next generation.

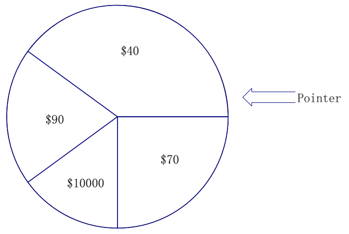


Fig.1. Roulette wheel

The following steps show selection using roulette wheel strategy:

1) Compute the sum of the fitness value of every individual in the population.

2) Evaluate the relative fitness value of each individual. Compute the proportion of each individual's fitness value to the sum of fitness value of all individuals in the whole population. The proportion represents the probability of the individual to be selected.

3) Partition a roulette according to the proportions computed in the second step. Every sector represents an individual. The area of the sector is proportional to the individual's probability to be selected. Spin the roulette wheel *n* times where *n* is the number of individuals of the population. When the spun roulette stops, the sector where pointer pointing at represents the corresponding individual being selected.

Now, consider a population with size *n* (the number of the individuals in the population is *n*), P=$\{a_1,a_2,a_3,...,a_n\}$, individual a_i has the fitness value of $f(a_i)$, then the probability for a_i to be selected is

$$ps(a_i) = \frac{f(a_i)}{\sum_{i=1}^{n} f(a_j)}, j = 1,2,...,n$$

Fig 2 shows the roulette wheel selection.

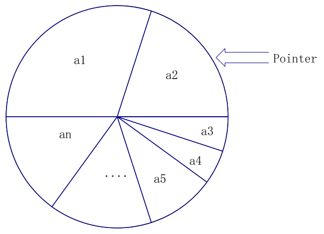


Fig 2 Roulette wheel selection shows that the survival probability of each individual is proportional to its relative fitness.

B. Tournament Selection

Tournament selection is also a selection strategy which selects individuals based on their fitness value. The basic idea of this strategy is to select the individual with the highest fitness value from a certain number of individuals in the population into the next generation. In the tournament selection, there is no arithmetical computation based on the fitness value, but only comparison between individuals by fitness value. The number of the individuals taking part in the tournament is called tournament size. The selection performs following the steps as below:

1) Randomly select several individuals from the population to take part in the tournament. Choose the individual that has the highest fitness value from the individuals selected above by comparing the fitness value of each individual. Then the chosen one is copied into the next generation of the population.
2) Repeat step1 *n* times where *n* is the number of individuals of the population.

TABLE I. LIST OF 7 TEST FUNCTIONTS

Test functions	Constraint	function value	Difference
$f1(x_1,x_2)= x_1^2+x_2^2$	[-10,10]	Maximum	2
$f2(x_1,x_2,x_3,x_4)= x_1^2+x_2^2+ x_3^2+ x_4^2$	[-10,10]	Maximum	2
$f3(x_1,x_2)= -x_1*\sin(\sqrt{\lvert x_1\rvert})-x_2*\sin(\sqrt{\lvert x_2\rvert})$	[-500,500]	Minimum	0.1
$f4(x_1,x_2)= x_1+x_2$	[0,100]	Maximum	1
$f5(x_1,x_2)=x_1^2-10*\cos(2*\pi*x_1)+10+x_2^2-10*\cos(2*\pi* x_2)+10$	[-50,50]	Minimum	0.1
$f6(x_1,x_2)= \lvert x_1\rvert+\lvert x_2\rvert+\lvert x_1*x_2\rvert$	[-10,10]	Maximum	0.1
$f7(x_1,x_2)=4*x_1^2-2.1*x_1^4+x_1^6/3.0+x_1*x_2-4*x_2^2+4* x_2^4$	[-10,10]	Minimum	0.001

4. Experiments

In this section, we are going to discuss the design of SGA. When using a different selection strategy, SGA has the same crossover and mutation operation. The algorithm is described as below:

```
BEGIN
  WHILE (Times<1000) Do
    BEGIN SGA
      Initialization;
      Evaluation;
      Keep the highest fitness value;
      generation:=0;
      WHILE(generation<MAXGENS) Do
        BEGIN
            generation++;
            Selection;
            Crossover;
            Mutation;
            Evaluation;
        END;
    END SGA ;
END ;
```

In the algorithm above, *Times<1000* is the termination condition, which indicates the times for executing SGA. Because GAs are stochastic computational techniques, we have to perform SGA many times for one problem so that we can get a statistically good result.

POPSIZE represents the number of individuals of the population. *MAXGENS* is the maximum number of the generations in the evolutionary process and we set *MAXGENS* as *1000* here. *PXOVER* is the probability of the individuals selected to cross over and *PMUTATION* is the probability of the individual selected to mutate. For each problem using SGA, we have different parameter groups. We set the value of *POPSIZE* as 50, 100, 150, 200, 250, the value of *PXOVER* as 0.1, 0.3, 0.5, 0.7, 0.9, and the value of *PMUTATION* as 0.05, 0.1, 0.15, 0.2, 0.25. After performing permutation and combination to the three parameters of *POPSIZE*, *PXOVER*, *PMUTATION* for SGA, we have 100 parameter groups. Therefore, to each problem to be solved, the algorithm will be executed 100 times with different groups of parameters, each time with a parameter group. The algorithm uses roulette wheel and tournament selection for the operation of selection and with each strategy SGA has 100 results respectively. Specially, the number of the individuals taking part in the tournament is 6 for the tournament selection strategy.

We have experimented on seven test functions using SGA. TABLE I shows the seven test functions. Each function has a prescribed search domain given in *Constraint* column of the table. *Problem* column indicates the solution we want to obtain. The optimal solution of each function is known beforehand.

When the absolute difference between the candidate solution and the optimal solution is smaller than a given value, which is shown in the *Difference* column, the process of SGA ceases. For a given function, each time of the 1000 times for executing SGA, we keep the value of *generation* when the process of SGA terminates. The value of *generation* is in integer ranged from 0 to 1000. Each value of *generation* has a corresponding counter preserving the times for SGA obtaining this *generation*.

5. Results and discussions

In this section, we will have a discussion on the comparative performances of the SGA with different selection strategies. There are totally 200 results for each function according to different parameter groups and different selection strategies. For each parameter group, SGA will be executed 1000 times. Each time we obtain a value of *generation* when process ceases. The smaller the value of *generation* is, the more quickly the algorithm converges.

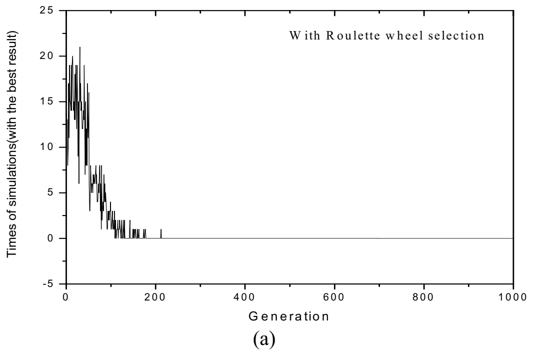
Let us focus on the results of the algorithm based on this parameter group: *POPSIZE*=100, *PXOVER*=0.1, *PMUTATION*=0.15. With these parameters, SGA is executed for 1000 times. The following figures show the results of the seven functions, where X Axis represents the value of *generation*, Y Axis represents the times for obtaining satisfied solutions at a certain *generation*. A point in the figure represents a certain times of working out the satisfied solutions at a certain generation. Take Fig.3(a) for example, the point in the curve where X Axis values *100* has a Y Axis value as *4*, which means there are *4* times when SGA works out satisfied solutions in the *100th* generation. SGA obtains the satisfied solutions at earlier generation in more times, the performance of the algorithm will be better.

Based on the result in Fig.3, TABLE II evaluate the sums of getting satisfied solutions in early generations by comparing Fig.3(a) and Fig.3(b). In the first 10 generations, the algorithm hits 135 times in roulette wheel selection while 427 times in tournament selection. In the first 20, 30, 40 and 50 generations, the algorithm hits 297, 436, 581 and 703 times in roulette wheel selection while 747, 914, 971 and 990 times in tournament selection respectively. After 100 generations, the algorithm with tournament selection has all worked out the satisfied solution in the 1000 trails, but the one using roulette wheel selection only hits 952 times on the satisfied solution. We can see that tournament selection performs much better than roulette wheel selection in function $f1(x_1,x_2)= x_1^2+x_2^2$. Fig.4 to Fig.9 which uses different functions also shows that the algorithm with tournament selection performs better than roulette wheel selection.

TABLE II.

SUMS OF GETTING SATISFIED SOLUTIONS IN EARLY GENERATIONS BASED ON ROUTLETTE WHEEL SELECTIO IN Fig3(a) AND TOURNEMENT SELECTION IN Fig.3(b) TOTALLY 1000 TRAILS, WHILE 1000 GENERATIONS IN ONE TRAIL.

Selection Strategy	Generation					
	10	20	30	40	50	100
Roulette wheel	135	297	436	581	703	952
Tournament	427	747	914	971	990	1000



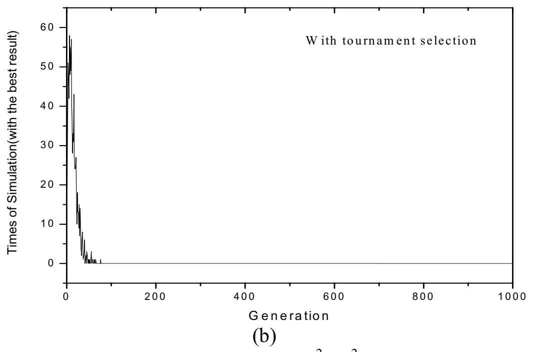
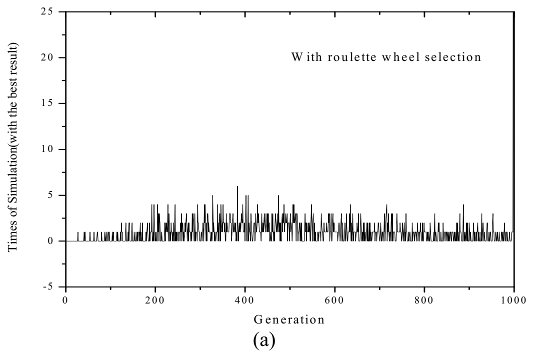
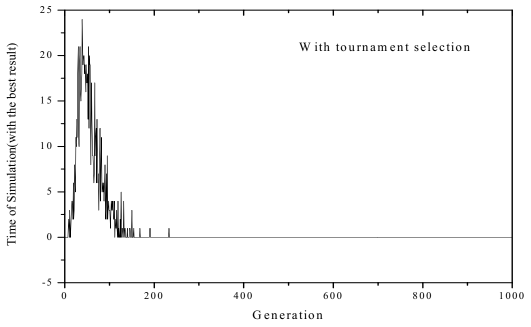


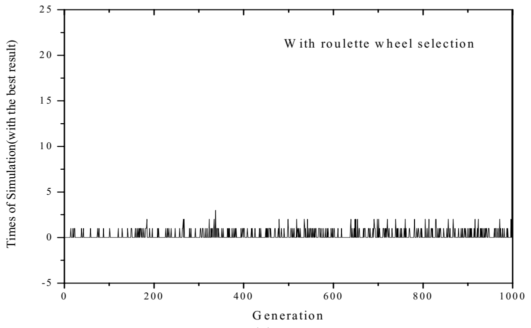
Fig.3. Results for function $f1(x_1,x_2)= x_1^2+x_2^2$: (a) by roulette wheel selection,(b) by tournament selection.



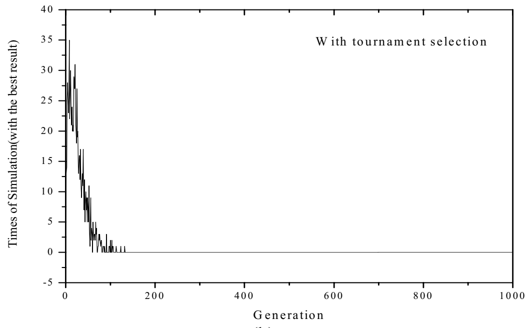


(b)

Fig.4. Results for function $f2(x_1,x_2,x_3,x_4)= x_1^2+x_2^2+ x_3^2+ x_4^2$: (a) by roulette wheel selection,(b) by tournament selection.

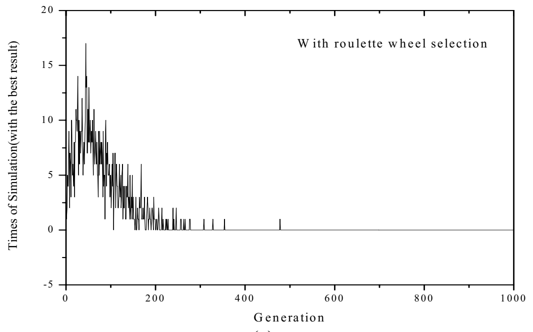


(a)

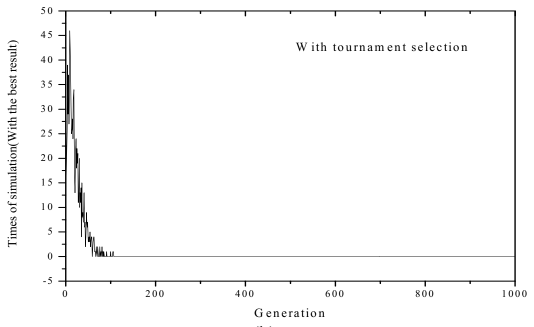


(b)

Fig.5. Results for function $f3(x_1,x_2)= - x_1*\sin(\sqrt{|x_1|})- x_2*\sin(\sqrt{|x_2|})$: (a) by roulette wheel selection,(b) by tournament selection.

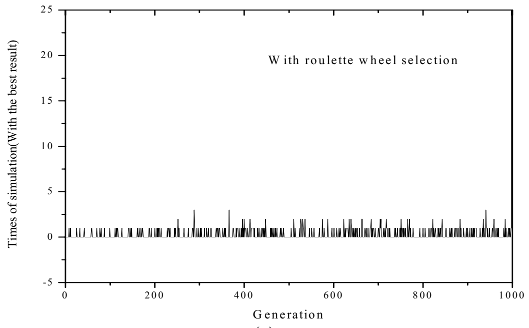


(a)

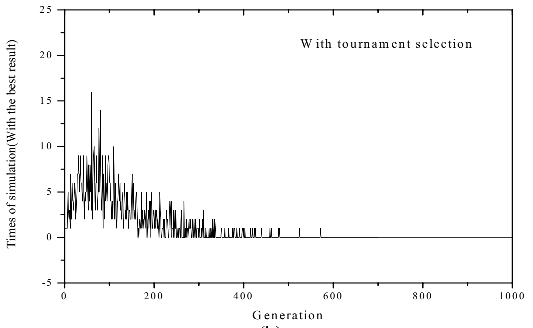


(b)

Fig.6. Results for function $f4(x_1,x_2) = x_1+x_2$: (a) by roulette wheel selection,(b) by tournament selection.

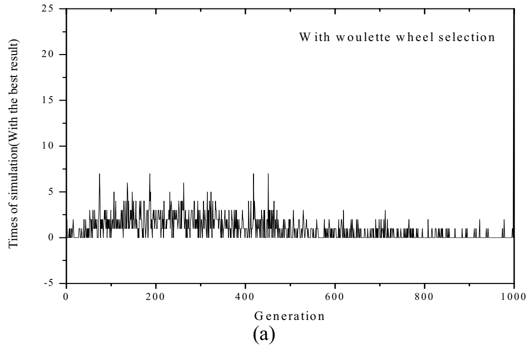


(a)

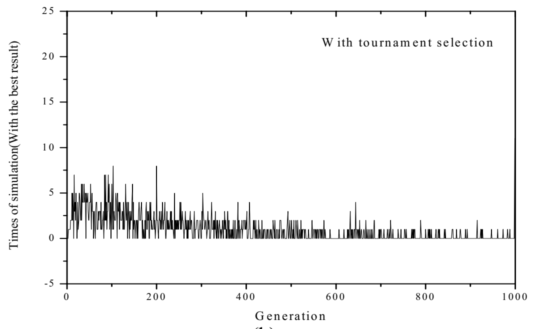


(b)

Fig.7. Results for function $f5(x_1,x_2)=x_1^2-10*\cos(2*\pi*x_1)+10+x_2^2-10*\cos(2*\pi* x_2)+10$: (a) by roulette wheel selection,(b) by tournament selection.

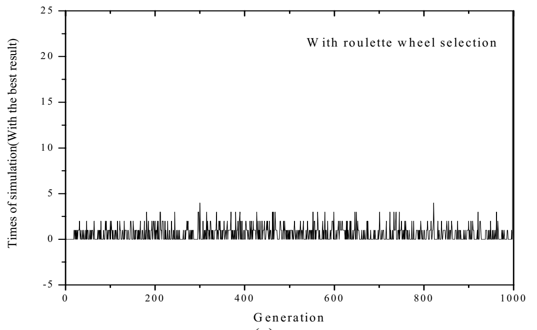


(a)

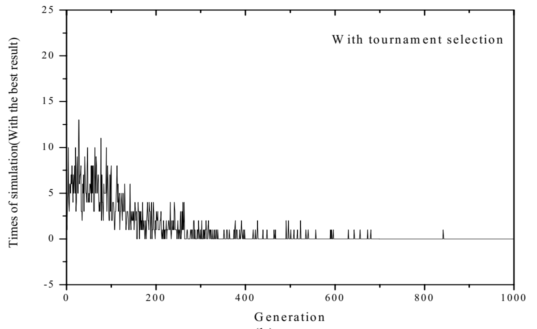


(b)

Fig.8. Results for function f6(x_1,x_2)= $|x_1|+|x_2|+|x_1*x_2|$: (a) by roulette wheel selection,(b) by tournament selection.



(a)



(b)

Fig.9. Results for function f7(x_1,x_2)=$4*x_1^2-2.1*x_1^4+x_1^6/3.0+x_1*x_2-4*x_2^2+4*x_2^4$: (a) by roulette wheel selection,(b) by tournament selection.

From the results shown above, SGA using tournament selection always obtains the satisfied solutions with more times at earlier generations than roulette wheel selection. This indicates SGA based on tournament selection converges more quickly than roulette wheel selection. The quicker the speed of convergence of the algorithm is, the less time for the algorithm taking to solve a problem. From this point of view, SGA using tournament selection has better performance than roulette wheel selection. For other parameter groups, the results are the same that GA converges more quickly with tournament selection.

6. Conclusions

This paper introduces a comparison of performance of SGA using different selection strategies. From the experiment of 7 functions tested, SGA based on tournament selection is more efficient in convergence than roulette wheel selection. As the size of population is large enough, we did not encounter the problem of prematurity. Therefore, the performance of SGA based on tournament selection outperforms the one based on roulette wheel selection.

7. References

[1]. N. Chaiyaratai̇ia and A. M. S. Zalzala, Recent Developments in Evolutionary and Genetic Algorithms: Theory and Applications, Genetic Algorithms in Engineering Systems: Innovations and Applications, 2-4 September 1997, Conference Publication NO. 446.

[2]. D.S.Huang, Zheru Chi and Wan-Chi Siu, "A case study for constrained learning neural root finders," Applied Mathematics and Computation, vol.165, no. 3, pp.699-718, 2005.

[3]. D.S.Huang, Horace H.S.Ip, Law Ken C K and Zheru Chi, "Zeroing polynomials using modified constrained neural network approach," IEEE Trans. On Neural Networks, vol.16, no.3, pp.721-732, 2005.

[4]. D.S.Huang, Horace H.S.Ip and Zheru Chi, " A neural root finder of polynomials based on root moments," Neural Computation, Vol.16, No.8, pp.1721-1762, 2004.

[5]. D.S.Huang, "A constructive approach for finding arbitrary roots of polynomials by neural networks," IEEE Transactions on Neural Networks，Vol.15, No.2, pp.477-491, 2004.

[6]. D.S.Huang, Wen-Bo Zhao, "Determining the centers of radial basis probabilities neural networks by recursive orthogonal least square algorithms," Applied Mathematics and Computation, vol 162, no.1 pp 461-473, 2005.

[7]. D.S.Huang, Horace H.S.Ip, Zheru Chi and H.S.Wong, "Dilation Method for Finding Close Roots of Polynomials Based on Constrained Learning Neural Networks," Physics Letters A, Vol.309, No.5-6, 443-451, 2003.

[8]. Wael Mustafa, *Optimization of Production Systems Using Genetic Algorithms*, International Journal of Computational Intelligence and Applications Vol. 3, No. 3, 2003, pp. 233-248.

[9]. Kwang Y. Lee, Xiaomin Bai, Youn-Moon Park, *Optimization Method for Reactive Power Planning by Using a Modified Simple Genetic Algorithm*, IEEE Transactions on Power Systems, Vol.10, No.4, November 1995, pp.1843-1850.