A Combined Evolutionary Algorithm for Real Parameters Optimization

Jinn-Moon Yang and C.Y. Kao*
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
E-mail: moon@solab.csie.ntu.edu.tw

Abstract - The real coded genetic algorithms (RCGA) have proved to be more efficient than traditional bit-string genetic algorithm in parameter optimization, but the RCGA focuses on crossover operators and less on the mutation operator for local search . Evolution strategies (ESs) and evolutionary programming (EP) only concern the Gaussian mutation operators. This paper proposes a technique, called combined evolutionary algorithm (CEA), by incorporating the ideas of EP and GAs into ES. Simultaneously, we add the local competition into the CEA in order to reduce the complexity and maintain the diversity. Over 20 benchmark function optimization problems are taken as benchmark problems. The results indicate that the CEA approach is a very powerful optimization technique.

I. Introduction

The function optimization problems have widespread application domain, including those of optimizing models, solving systems of nonlinear equations and control problem. Recently, the technique has been applied to train the weight of neural networks[1].

The Function optimization problems can be generalized in the following standard form[1]:

$Minimize\ f(x_1, x_2, \ldots, x_n)$

or

$Maximize\ f(x_1, x_2, \ldots, x_n)$

where x_i *is a real parameter subject to* $a_i \le x_i \le b_i$

a_i *and* b_i *are some* constant constraint

Evolutionary computations, based on natural selection in Darwin's theory, are studied and applied in three standard formats: genetic algorithms, evolution strategies, and evolutionary programming [3][4]. These methodologies have great robustness and problem independence; therefore, they are powerful optimization techniques in many applied domains, such as combinatorial optimization[6], function optimization [1][2][9][8], control, and machine learning[3,4]. Genetics algorithms[4][11], the most known and discussed in USA, stress chromosomal operator, such as crossover operator. Evolution strategies[9][15] and evolutionary programming [2][3] emphasize mutation operation for the behavioral link between parents and offspring, rather than the genetic link which is stressed in genetic algorithms.

Traditional GAs use bit-string to represent problem domains, but this representation is not suitable and not natural in some application domains, such as real-coded. In recent years, some authors[1,12] suggested the real-coded GA to represent the problem and proved this approach could gain better performance, but they focused on some modified crossover operators called blend crossover only . Basically, the operator may be viewed as a simple heuristic crossover in real-coded parameters. The main power and operators of GAs are crossover operators. Some studies[2,3,16,17] argued over the power of crossover operators. They explained and indicated that these operator are inappropriate in some applications. So far, the GAs lack fine local tuning capabilities .

Schwefel[9] extended the $(1+\mu)$-ES towards a $(\mu+\lambda)$-ES and (μ,λ)-ES for numerical optimization and proposed an algorithm for the capability of self-adapting step size. Both methods were all the same except the selection philosophy. The $(\mu+\lambda)$-ES selected the top μ population, based on fitness value, form μ (parents) and λ (children), but the (μ,λ)-ES selected the top μ population form the λ (children) only. That is, the lifetime for each individual is only one generation in the (μ,λ)-ES. The selection of the ESs is strictly deterministic and elicited policy, with the possibility of causing the premature convergence.

* All correspondence should be sent to the second author.

The Gaussian mutation operator[9], [15], could be viewed as heuristic mutation operator, the main power of ESs. The main strategies parameter in ESs is the mutation size and the step size control. Many studies discussed the operator and parameters control.

Fogel[3] wrote "there are two essential differences between ESs and EP. 1. ESs rely on strict deterministic selection. EP typically emphasizes the probability nature of selection by conducting a stochastic tournament for survival at each generation,... 2. ...,ESs may use recombination operators to generate new trials, but EP does not, as there is no sexual communication between species."

Generally, the typical ESs use a parents/offspring ratio of 1/6 ($\mu/\lambda=1/6$) as opposed to the 1 ratio used in the EP. So, the EP can be viewed as $(\mu+\mu)$–ES when excluding the tournament selection operator.

II. The Combined Evolutionary Algorithm

The CEA technique combines the philosophy of ESs,GAs and EP. The detailed algorithm is shown in Fig. 1. The basic idea of CEA is similar to the evolution strategies. However, there are three essential differences between CEA and ESs.

1) CEA incorporates the EP stochastic tournament selection[3] to replace the ES deterministic and elicited selection.

2) CEA views the blend crossover[12] as big-step-size mutation, and the crossover rate must decease dynamically. This operator is responsible for global search in the beginning phase. At the initial phase, we hope this operator can guide the search direction approximately in order to reduce the unnecessary search and concentrate on the interesting area as possibly as. The combination operator of ESs focuses on the step size only[15].

3) CEA Incorporates the local competition as GESA[6] in order to avoid the ill effect of greediness. The children, generated from the same parent by the Gaussian mutation operator, compete with each other, and only the best child survives and participates in the selection operators . That is, only $(\mu+\mu)$ individuals have the probability to become population of the next generation. Respectively, Both $(\mu+\lambda)$-ES and (μ,λ)-ES select from all children.

Initially, the CEA generates 2*P feasible solutions by uniform distribution in the feasible search space. Using the EP tournament selection (each member competes against k (general k=10) others, chosen at random based on the corresponding fitness value. Then, each member is assigned a score w(0<=w<=k) based on the number of the competition wins. Select P members with the most wins to be the parents for the next generation) to select P members. These selected members become the parents of blend crossover (BLX-0.5). The BLX-0.5 randomly selects a point from the line connecting two parents extension of half the distance of the parents ate each end. In the BLX-0.5, we use the GA tournament selection(size=2) to select two parents and to generate one child by the BLX-0.5. In the process, the BLX-0.5 generates P candidates. These candidates become the parents of the Gaussian mutation operators. Each parent generates n children, based on Gaussian mutation operator(i.e $x'=x+\delta*N(0,1)$. The children generated by the same candidate are viewed as a family. In each family only one best child, based on evaluated value, can survive. Therefore, exactly, two members (the parent and the best child) are put into the population pool in each family. Thus, the population size is invariant(2*P individuals).

The original real-code GAs use blend crossover and random mutation, so they lack fine local tuning capabilities. The Gaussian mutation operator is good local search method[7]. We suggest the combination of the blend crossover and Gaussian mutation operator. Initially, the crossover operator finds the approximated solution and the mutation operator adjusts the bias. The crossover may reduce the performance when near the optimal and the mutation become the main power, respectively. Therefore, we increase the length of Gaussian mutation and decrease the crossover rate dynamically. This strategy will improve the quality greatly.

The local competition of Gaussian mutation search can avoid early superstar domination the whole of population. Exactly one child in the same family can survive in CEA but at most λ children can survive in ES, respectively. The EP selection tournament strategy plus the local competition can maintain the diversity. To reduce the complexity of EP tournaments selection is another benefit of local competition. The complexity of EP-selection is TS * P + P log P where P is the population size and the TS is the tournament size (times of the competition). The size in CEA is 2μ and 8μ in general $(\mu+\lambda)$-ES, respectively. Therefore, the local completion can reduce the complexity.

The CEA technique has three phases in the evolutionary search.

1) initial phase: The primary purpose in the phase is to find the approximate solution and to reduce the search space. The crossover operator is the main operator for global search and mutation operator for adjusting the bias. The phase takes about 10% of the computing time.

2) middle phase: Both mutation and crossover are equal important. This phase takes about 20% of computing time.

3) last phase: The main purpose in the phase is to tune the solution in order to find the best solution . The Gaussian mutation is primary operator and very little disturbance is made by crossover operator. The phase spends about 70% computing time.

```
1. Set the strategy parameters: Local_Search_Length, Cross_Over_Rate and GaussianMutationSize( $\delta$)
2. Randomly generates 2*P candidates by uniform distribution form feasible solution
3. Compute the fitness of each candidates. ( $E(X_i)$ ,i=1,....,2P)
4. Select $X_i$(i=1,...,P) parents by EP tournament selection from $X_i$(i=1,...,2P)
5 For each $X_i$(i=1,...,P) execute blend crossover(BLX-0.5) generate a child C
    if E(C) < E(Xi) then replace Xi else keep Xi ( when minimize the fitness function)
6. Generate $X_i$(i=P+1,...,2P) for each $X_i$(i=1,...,P){
        generate Local_Search_Length "offspring" through Gaussian mutation operator ($x'=x+\delta*N(0,1)$) from the same parent
        select the best offspring based on fitness as the child
      ($X_i$(i=P+1,...2P) .
  }
7 Change the parameters for next generation when satisfy some constraints {
        Cross_Over_Rate=Cross_Over_Rate - 0.05;
        Local_Search_Length=Local_Search_Length + 1;
        $\delta$= 0.95 * $\delta$
  }
8. If discovery sufficient solution or exhaust the available time then terminate else goto step 3

parameters setting: (for all benchmark problems)
 Local_Search_Length: initial = 3, MaxLen = 10,
                      increase by 1 for each 10 generations
CrossOverRate: initital = 0.5, MinRate = 0.05,
               decrease by 0.05 for 10 generations
δ : initial=0.2*abs(Max_Constraint-MinConstraint),
               decrease rate = 0.95.
Poplation_size = 50
```

Fig 1. The Combined Evolutionary Algorithm.

III. The CEA Approach to Function Optimization: Empirical Results, findings and discussions

We have implemented the CEA algorithm on function optimization. The strategy parameters (Local_Search_Length, Gaussian_Mutation_Size, and CrossOverRate) were given the value as in the fig 1 for all testing function. At first, we compared the CEA technique with other evolutionary algorithms for over 20 benchmark problems. The CEA always gained better results. The CEA is also robust in various problems. We discuss the effect and influence of strategy parameters, and try to explain the reasons.

CEA's performance has been compared with traditional GA, real-code GA and EP. We select over 20 benchmark functions and these test functions are given in the table 1. The F1 to F7 and f9 have been studied by De Jong[11], Schaffer[10] and Wright[1]. These problems are benchmark functions in GAs. Yip[6] studied the F8 and compared with simulated evolution. F11 to F19 were collected and studied for EP by Koon[8]. These functions are the benchmark function in the SIAM journal on Scientific Computing. The F21 to F23 are studied in McDonnell [13]. The three functions have 30 dimensions and studied in ES. Above all, F23 is very complex function and high dimensions(30). Fig 2 shows one-dimensional slice of F23. All these functions search for the global minimum..

Table 1: the testing function

function #	dimension #	domain	global minimum	Source (reference)
F1	3	-5.12, 5.12	**0**	F1 of De Jong (1975)[11]
F2	2	-2.048, 2.048	**0**	F2 of De Jong(1975)[11]
F3	5	-5.12, 5.12	**-30**	F3 of De Jong (1975)[11]
F4	30	-1.28, 1.28	**0**	F4 of De Jong (1975)[11]
F5	2	-65.536, 65.536	**0.998**	F5 of De Jong (1975)[11]
F6	2	-100,100	**0**	F6 of Schaffer(1989)[10]
F7	2	-100,100	**0**	F7 of Schaffer(1989)[10]
F8	2	-10,10	**1**	Appendix of P.Yip(1995)[6]
F9	2	-65.536, 65.536	**0.998**	F5R of A.H Wright(1991)[1]
F11	2	-5,5	**1.032**	F1 of G.H. Koon(1995)[8]
F12	2	-20,20	**-186.731**	F2 of G.H. Koon (1995)[8]
F13	4(N=4)	-20,20	**-156.665**	F3 of G.H. Koon (1995)[8]
F14	2	-5,5	**0**	F4 of G.H. Koon (1995)[8]
F15	2(n=4)	-20,20	**-2429.415**	F5 of G.H. Koon (1995)[8]
F16	2	-5,5	**-2**	F6 of G.H. Koon (1995)[8]
F17	2	-20,20	**0.398**	F7 of G.H. Koon (1995)[8]
F18	1	0.1,6.0	**-5.534**	F8 of G.H. Koon (1995)[8]
F19	1	-20,20	**-3.373**	F9 of G.H. Koon(1995)[8]
F21	30	-30,30	**0**	F2 of J.R. McDonnell(1995) [13]
F22	30	-30,30	**0**	F3 of J.R. McDonnell(1995)[13]
F23	30	-30,30	**0**	F9 of J.R. McDonnell(1995)[13]

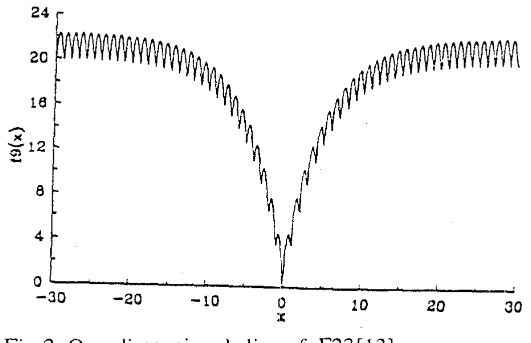

Fig 2. One-dimensional slice of F23[13]

First. in order to compare with EP, the functions from F11 to F19 were executed with over 50 runs by CEA. We started the parameters as in the fig 1 and terminated the process after 1000 generations(about 480,000 function evaluations). The CEA found all function global minimums at 200 generations (about 86,000 function evaluation) in each run. We used F15 as an example to shows the convergence curve and given in Fig 3. The summary of these experiment are given in Table 2. The EP, after executing 2000 generations (about 100,000 function evaluations), cannot find the global minimum for complexity functions (F12,F13.F14 and F15). As shown in Table 2, the CEA outperforms the EP on these benchmark problems..

Table 2. Comparison with EP[G.H.Koon][8] on F11 to F19(benchmark problem in the Journal on Scientific Computing. For each function the CEA runs 50 times (terminated 1000 generations(about 480000) and finds the global minimum about 200 generations(about 86,000 function evaluations).

Function #	EP[8]	CEA(mean)	Optimal
F11	-1.0312	-1.0312	-1.0312
F12	-186.693*	-186.731	-186.731
F13	-156.119*	-156.665	-156.665
F14	0.001*	0	0
F15	-2429.353*	-2429.415	-2429.415
F16	-2.000	-2.000	-2.000
F17	0.398	0.398	0.398
F18	-5.534	-5.534	-5.534
F19	--3.373	--3.373	--3.373

*: EP cannot find the global minimum of the function. and CEA can find the global minimum all functions.

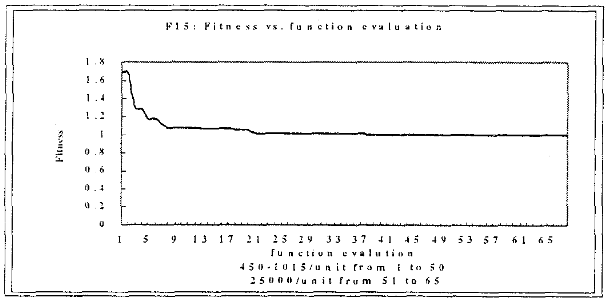

Fig 3: The convergent curve of F15

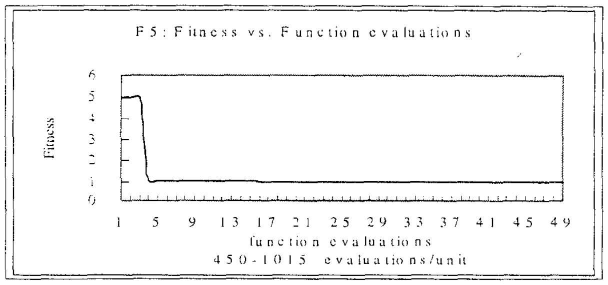

Fig 4: The convergent curve of F5

Table 3 compares the performance of CEA with that of the traditional GA and real-coded GA[1] for F1 to F7 and F9. This result indicates that CEA outperforms the traditional GA and real-coded GA. The CEA can find the global minimum for all except function F7 (mean different value about equal 1E-6). CEA needed about 50 generations(about 13,000 function evaluations) for F5 and F9, while other functions needed about 200 generations (about 86,000 function evaluations). The convergent curve of the more complex function F5 was given in fig 4. In these experiments we found the real-coded GA to converge optimal faster than CEA. But the real-code GA could not find the global optimum when it trapped into the local optimal. This result implies that the blend crossover has poor ability for local search. Fig 6 indicates and also explains that the blend crossover is not suitable for local search in the real-coded GA.

Table 3. CEA compares with GA and RCGA [A.H.Wright] [1] on F1 to F7 and F9 . The CEA value in table are the mean of 50 runs

Function #	Real-code GA	Traditional GA	CEA(mean)	Optimal
F1	0.00004	0.0062	0	0
F2	0.0207	1.0192	0	0
F3	0.5730	1.4740	0	0
F4	-1.0906	9.7311	0	0
F5	9.7598	1.0116	0.9980	0.9980
F6	0.0200	0.0310	0	0
F7	0.0584	0.5888	0.0000065*	0
F9(5R)	11.4997	8.6699	0.9980	0.9980

*: the CEA cannot find the global minimum

The GESA[6] approach incorporates the local competition in the same family and the authors use the F8 as a benchmark problem to compare with simulated evolution. The GESA needed about 800,000 function evaluations to find the global optimum in each runs. In the same search space, the CEA executes 50 runs and table 4 shows the result. Fig 5 shows the convergence curve. Each experiment of the CEA finds the global minimum and needs 330,000 function evaluation on average. The main reason may be the local competition length. Each parent in GESA generates 20 children to compete on average, but the CEA only generates 3 to 10 children. Maybe the crossover and EP-selection cause the result.

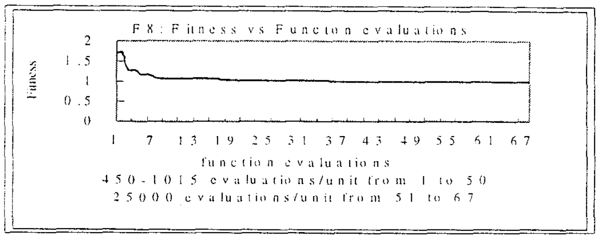

Fig 5: The convergent of F8

In order to explain the effect of blend crossover in real coded representation, we set the crossover rate to 1 and fixed the rate. Table 4 indicates the high and fixed crossover rate policy will reduce the performance when the function is complex. The result may be caused by two main factors.

1) The crossover operator is a larger step search than Gaussian mutation and is not suitable for the fine tuning when the solution is near the optimal. Therefore, in the last phase, the crossover must be inhibited or reduced to very low probability.

2) The Gaussian mutation is main role in the last phase and has excellent ability for fine tuning.

Table 4. A comparison of blend crossover rate strategy: high and fixed rate policy vs. lower and dynamic rate policy (mean value of 50 runs)

Function #	static: crossover rate =1	CEA: dynamic: crossover rate (from 0.5 to 0.05 decease by 0.05)	Optimal
F1	0	0	0
F2	0	0	0
F6	0.0003	0	0
F8	1.001329	1	1
F22	228.4286	0	0
F23	11.72984	1.82E-07	0

The step size of Gaussian mutation is the most important parameter for ESs and EP. Therefore, many studies have discussed the control of the step size. In this paper, we use very a simple rule, fixed decreasing rate, to control the steps size. Table 5 indicates the influence of the Gaussian mutation size in CEA. Primarily, This experiment showed that this simple rule is enough good when the initial step size is not small enough. Because small mutation size makes the CEA too greedy, it causes the premature convergence.

In order to explain and prove the power of local competition, we used only the mutation operator and local competition. For each function, randomly execute 50 runs. We discovered that the results were almost as well as the original CEA almost except the F8 and F23. This result might be caused by the complexity of function and lack of BLX-0.5 crossover operator.

Table 5. The effect and influence of Gaussian mutation size(mean value of 50 runs)

Function #	initial_size =0.05*(Max_Constraint - Min_Constraint)	initial_size =0.2*(Max_Constraint - Min_Constraint)	Optimal
F5	2.09841	0.998	0.998
F6	0	0	0
F8	1	1	1
F9	2.19549	0.998	0.998
F22	12.78935	0	0
F23	9.52435	1.82E-07	0

Using Gaussian mutation as a search operator, the typical search length for EP and ESs is 1 and 7, respectively. CEA views the search length as competition numbers. Table 5 indicates the influence of the Gaussian local search length. With more variables (e.g. F22 and F23) and complex functions the search length must be longer. The local search spent most computing time in CEA. Search length over 10 is not worthwhile considering the the tradeoff between search time and quality. The effect of CEA's parameters, based on the complex and high dimensions F23, is showed in fig. 6.

Table 6. The effect influence of local search length (mean value of 50 runs)

Function #	search length =1	CEA: dynamic search length from 3 to 10 increase by 1)	Optimal
F6	0.006668	0	0
F8	1.0027	1	1
F13	-156.664663	-156.664663	-156.664663
F15	-2429.414767	-2429.414767	-2429.414767
F22	449.6078	0	0
F23	12.4816	1.82E-07	0

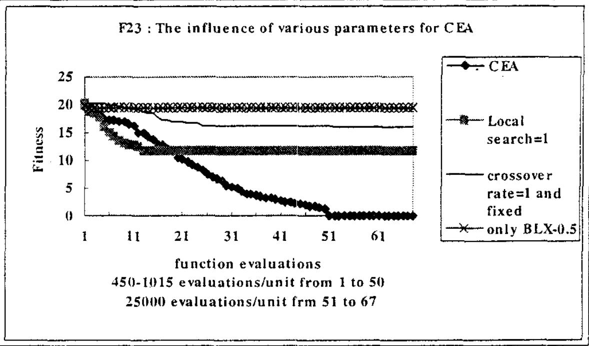

Fig 6. The convergent curve for various parameters based on F23

IV. Conclusion

Incorporating the ideas of EP and GAs into evolution strategies, CEA was found to be a very powerful optimization technique. As demonstrated with the use of the 20 benchmark function problems, the CEA outperformed other simulated evolutionary algorithms, including the EP,ESs, traditional GA and real-coded GA. The CEA uses the same parameters with small population size and simple control rule to solve all functions and converge to the optimum or point very near the optimal(mean different value lower than 1E-6).

The evolutionary search process of CEA can be viewed in three phases. In the first phase, use crossover (BLX-0.5) as a global search operator in order to find approximate solutions. The Gaussian mutation can be viewed as a local search operator to adjust the bias. In the second phase, both the crossover operator and mutation operator are equally important, in order to explore and exploit the search information. In the last phase, the crossover becomes less important and mutation becomes the main operator.

In the future research, we will use CEA to solve NP-Hard combinatorial optimization problems.

reference:

[1]. A.H. Wright,"Genetic Algorithm for real Parameter Optimization", in Proc FOGA '91, pp 205-218.

[2]. D.B. Fogel and J.W. Atmar, "Comparing Genetic Operators with Gaussian Mutations in Simulated Evolutionary Processes Using linear Systems," Biological Cybernetic, vol, 63, 1993, pp 111-114.

[3]. D.B. Fogel ,"An Introduction to Simulated Evolutionary Optimization", ,IEEE trans. Neural Networks, vol 5,no. 1, Jan. 1994, pp 3-14

[4]. D.E. goldberg, Genetic Algorithms in search, Optimization & Machine Learning, Reading. MA: Addison-Welsley.1989.

[5]. D.E. goldberg and K.Deb," A Comparative Analysis of Selection Schemes Used in Genetic Algorithms," in Proc. FOGA 91, pp 69-93.

[6]. D. Yip and Y.H. Pao, "Combinatorial Optimization with Use of Guided Evolutionary Simulated Annealing," IEEE trans. Neural Networks, vol. 6,no. 2, Mar 1995, pp 290-295

[7]. F. Hoffmeister and T. Back,"Genetic Algorithms and Evolution Strategies: Similarities and difference," in Proc. of the First International Conference on Parallel Problem Solving from Natural, 1991 , pp 455-470.

[8]. G.H. Koon and A. V. Sebald, "Some interesting Functions for Evaluating Evolutionary Programming Strategies," in Proc Evolutionary Programming IV 1995, pp 479-499

[9]. H.P. Schwefel, Numerical Optimization for Computer Models. Chichester, UK:wiley 1981.

[10]. J.D. Schaffer, etal. " A Study of Control Parameters Affect On-line Performance of Genetic Algorithm for Function Optimal," in Proc of the Third ICGA, Morgan Kaufmann, 1989, pp 51-60.

[11]. K.A. De Jong. Analysis of the behavior of a class of genetic Adaptive System, Ph.D. Dissertation, Department of Computer and Communication Science,University of Michigan, 1975.

[12]. L.J Eshelman & J.D. Schaffer, " Real-coded Genetic Algorithms and Interval-Schemata," in Proc. FOGA '93.

[13]. Mcdonnell and D.E. Wagen,"An Empirical Study of Recombination in Evolutionary Search," in Proc Evolutionary Programming IV 1995, pp 465-448.

[14]. P.J.B. Hancock, "An Empirical Comparison of Selection methods in the evolutionary algorithms," in Proc. of the First International Conference on Parallel Problem Solving from Natural, 1991 , pp 81-94.

[15]. T. Back, F. Hoffmeister and H.P. Schwefel, " A survey of Evolution Strategies," in Proc. ICGA' 91 , pp 2-9.

[16]. D.B. Fogel ,"An Evolutionary Approach to traveling Salesman Problem," Biological Cybernetic, vol, 60, 1988, pp 139-144.

[17]. P.J. Angeline, G.M. Saunders, and J.B.Pollack ,"An Evolutionary Algorithm that Constructs Recurrent Neural Networks," IEEE trans. on Neural Networks, vol 5,no. 1, Jan. 1994, pp 54-65.