Proceedings of International Conference
on Algorithms

A Family Competition Genetic Algorithm for Nonlinear Constrained Optimization

J. M. 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- *Real world continuous numerical optimization problems are often heavily constrained conditions that divide into linear constraints and non-linear constraints; however, there exists no unique deterministic approach to nonlinear optimization problems. This paper proposes a technique, called family competition genetic algorithm (FCGA), which incorporates the ideas of evolution strategies(ESs) and family competition into GA. The FCGA is similar to the rule of Olympic Games. The family competition can be viewed as local competition of each area, and selection is global competition in the universal tournament. The family competition can reduce the time complexity and maintain the diversity of population. Eight nonlinear constrained optimization problems are taken as benchmark problems. The results indicate that the FCGA approach is a very powerful optimization technique.*

1. Introduction

Real world continuous numerical optimization problems are often heavily constrained conditions that divide into linear constraints and non-linear constraints. There are some unifying methods to solve linear programming, but there exists no unique approach to nonlinear optimization. Constrained global optimization problems are widespread applied to many domains, such as structural optimization, engineering design, VLSI design and database problems, economies of scales, allocation and location problems and quadratic assignment[13,25]. The methods that determine the solution of nonlinear constrained optimization problems can be divided into deterministic methods and stochastic methods. Traditional deterministic global search methods ,such as feasible direction methods, generalized reduced gradient method and penalty function methods[25,13,14], typically make strong assumptions regarding the objective function, such as continuity and differentiability. In the recent year, there has been growing effort to apply evolutionary algorithms, including genetic algorithms(GAs) [11,15,16,17,24] evolution strategies(ESs), and evolutionary programming(EP) to general constrained optimization problems in order to overcome these limitations. Without loss of generality, we discuss minimization in this paper. The general nonlinear programming optimization problem can be formulated as the following form:

$$\begin{aligned}
&\text{Minimize } f(x) && (1)\\
&\text{subject to}\\
&g_i(x) <= 0, \text{ for } i=1,\ldots,n, && \text{(inequality constraints)}\\
&h_j(x) = 0, \text{ for } j=1,\ldots,m, && \text{(equality constraints)}\\
&x \in W,
\end{aligned}$$

where W is a set of constraint, $x \in R^n$ and n is the dimension, g_i, h_j can be linear or non-linear constrainted function, if x satisfies all constraints then x is called as a feasible solution else x is called as an unfeasible solution, and if any one of $f(x), g_i$ or h_j is a nonlinear function then the problem is called nonlinear programming problem else the problem is called linear programming problem.

The traditional bit-strings GAs[4,20] applied to numerical optimization problem have some limitations and drawbacks[1,7]. The real coded genetic algorithms (RCGA) [7] 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. Therefore some research has[2,3] pointed out that the crossover operators are not always suitable for local search. Evolution strategies (ESs)[6,5,9] focus on Gaussian mutation and allow the application of some simple recombination operators. In the recent year, some authors incorporated various modified crossover operators in order to improve the power of ESs. Evolutionary programming (EP)[2,3] only concerns the Gaussian mutation operators.

This paper proposes a technique, called family competition genetic algorithm (FCGA), by incorporating the ideas of family competition and ESs into GA. We add the family competition into the FCGA in order to reduce the computing complexity and maintain the diversity of the population. Automatically balanced the exploration-exploitation can be represented by the FCGA. In early search stage, the system is lack of available knowledge, so FCGA uses crossover operators and bigger step size Gaussian mutation operator to explore the search space. Naturally, the more FCGA spent, the more information. was accumulated. Therefore, FCGA will reduce the step size of Gaussian mutation automatically in order to spend more time on exploitation. Thus, FCGA will spend more

time on exploitation and less time on exploration eventually. These policy that drastically improves the performance will be discussed in this paper The similar ideas of FCGA have been applied the function optimization problems[10,27] and these results indicate that the FCGA approach is a very powerful optimization technique

2. Constraint Handling methods

The global non-linear constrained optimization problems belong to the class of NP-hard problem. Even some problems of determining an approximate global solution are NP-hard, unfortunately.

Any evolutionary computation technique applied to a particular constrained problem should address the issue of handling unfeasible solutions. Generally, a search space consists of two disjointed subsets of feasible and unfeasible subspaces respectively. It is very important and non-trivial to deal with unfeasible individuals. The penalty is too large to find the better solution because the feasible areas are divided by the unfeasible solution areas. The evolutionary algorithm is very difficult to override these unfeasible areas. This policy may cause EAs to find a local optimal. The penalty is too small to find the feasible solutions [25,21,18]. Presently, there are some methods used to handle constraints with evolutionary algorithms: stationary penalty function[11,21,28], non-stationary penalty function[15], repair algorithm[22], co-evolutionary algorithm[23], reject the unfeasible solutions[16] and hybrid the stochastic and deterministic method[26]. The general fitness function form based on feasible solution and penalty for unfeasible solution can be defined as following:

$$fitness(x) = \alpha * f(x) + \eta(g) * PF(x) \qquad (2)$$

where

g is the generation number in experimental design,

$f(x)$ is the original optimizing function and

$PF(x)$ is the penalty function for unfeasible solution.

Then, the general form of $PF(x)$ defined as

$$PF(x) = \sum_{i=1}^{n+m} \theta(p_i(x)) * (p_i(x))^{r_{p_i(x)}}$$

$$\text{where } p_i(x) = \begin{cases} \max(0, g_i(x)) & 1 \le i \le n \\ \text{abs}(h_{j-n}(x)), & n+1 \le j \le n+m \end{cases}$$

θ is the level weight function of penalty function,

α is the weight of original function,

r is the power of the penalty function, and

$f(x), g_i(x), h_j(x)$ are defined in formula (1),

3. The Family competition genetic algorithm

The FCGA technique combines the philosophy of ESs, GAs and family competition. The FCGA views family competition as local competition and deterministic with elitist selection as global competition. Local competition and global competition can reduce the computing tiome of sectection operator and can keep the diversity of population, so FCGA has more opportunity to find the global optimal solution. FCGA is similar to the rule of Olympic Games. Family competition can be viewed as each country selecting a sportsman to participate with tournament and the elitist selection as selecting the best sportsmen in the Olympic Games. The detailed algorithm is shown in Fig. 1. The basic idea of FCGA is similar to the evolution strategies. However, there are three essential differences between FCGA and ESs.

1) FCGA incorporates the family competition in order to avoid the ill effect of greediness. The children, generated from the same parent by the crossover and Gaussian mutation operator, compete with each other, and only the best child can be selected by 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 which generated by the same parent.

2) FCGA incorporates simple self-adaptive and decreasing rate, similar to temperature of simulated annealing, to control the standard derivation size. The big-step-size mutation, viewed as a global search operator, explores the search space to avoid the bias of initial population and loss of global information. But the initial step-size and decrease rate is depend on the population size.

3) FCGA views the crossover operator and mutation operator as the same important operator. ESs view the crossover as minor operator and the mutation operator as main operator.

The FCGA has other basic differences from traditional GAs except above three characteristics

1). FCGA uses the real code representation not bit-strings.

2). FCGA uses Gaussian mutation as mutation operation not bit mutation and the mutation rate is 1 for each variable.

3). FCGA uses elitist and deterministic selection that selects μ best individuals out of the union of parents and children. This strategy is $(\mu+\mu)$-selection in ES.

3.1 Crossover

Any operator that combines two or over two parents is called a crossover operator. FCGA uses two different crossover operators: discrete crossover and blend crossover with extended 0.5(BLX-0.5). In order to explain the crossover operator we let two parents are $x=(x_1,...,x_n)$ and $y=(y_1,...,y_n)$ and the generated offspring is $c=(c_1,...,c_n)$ where n is the number of variables.

The discrete crossover, the recombination operators of ESs , is used in ESs widely and empirically has better results on object variables[9]. Discrete crossover generates the corners of the hybercube defined by the x and y. This operator is similar to the uniform crossover of GAs and works as follows:

$c_i=x_i$ or y_i with equality probability (3)

The blend crossover with extended 0.5[1,7], used successfully in GAs and ESs[8], generates a child on the extended line defined by x and y . It can be formulated as follows:

$c_i=x_i+\beta(y_i-x_i)$ β is chosen uniform randomly in [-0.5,1.5] (4)

3.2 Mutation

Generally, mutation operation yields a mutated offspring from only one parent. In FCGA we use Gaussian mutation, widely used in ESs and EP, to generate a child by first mutating the standard derivations(δ) and then by mutating the object variables according to the modified normal probability density function of $N(0,\delta)$. ESs use "self-adaptation" technique[6,9] to control the standard derivations(δ) but this technique may not fit to our family competition algorithm. Therefore, we use the annealing-like concept to control the standard derivations(δ). This is realized as follows:

$$c_i=x_i+\delta_i*N_i(0,1) \quad (5)$$
$$\delta_i = decease\_rate * \delta_i$$

3.3 Selection and family competition

FCGA uses elitist and deterministic selection that selects μ best individuals out of the union of parents and children. This strategy is $(\mu+\mu)$-selection in ES. The family competition operator selects the best child to survive and other children in the same family die; that is, the children, generated from the same parent, must compete with each other and only the best individual survives. Family competition can avoid early superstar domination of the whole population because exactly one child in the same family can survive in FCGA but at most λ children can survive in ES respectively. Therefore FCGA can avoid premature. To reduce the complexity of selection is another benefit of family competition. The complexity of the selection is P^2 where P is the population size. The family competition reduces the population size from 7μ to 2μ . Therefore, the local completion can reduce the computing complexity of selection.

1. *Set the strategy parameters: Competition_Length , and GaussianMutationSize(δ_k), Population_Size(P)*
2. *Randomly generates 2*P candidates by uniform distribution form the feasible range of Xi= ($x_{i1},...,x_{in}$)), where n is the number of variables.*
3. *Compute the fitness based on penalty function of each candidates. (fitness(Xi) ,i=1,...,2P)*
4. *Select X_i $_{(i=1,...,P)}$ chromosomes as parents by elitist and deterministic selection from X_i $_{(i=1,...,2P)}$*
5. *Generate Competition_Length children for each Xi(i=1,...,P) by*
   C_{ij}(xi1,...,xin)= Dircrete_Crossover(X_i, $X_{random(P)<>i}$)
   with probability 0.8 or
   BLX-0.5(X_i, $X_{random(P)<>i}$) with probability 0.2
   $C_{ij}(x_{i1},...,x_{in}) = x_{ik}+\delta_k*N(0,1)$ (1<= k <= n)
6. *$X_{i+P}(x_{i1},...,x_{in})=min(fitness(C_{ij}))$*
   (1<=i<=P, 1<=j<=Competition_Length)
7. *δ_k =decease_rate * δ_k (1<=k<=n n is the number of variables)*
8. *If discovery sufficient solution or exhaust the available time then terminate else goto step 4*

parameters setting: (for all benchmark problems)
Competition_Length=6
δ_k : initial=β*abs(Max_Constraint$_k$-MinConstraint$_k$)
where β = 0.2
decrease rate = 0.95.
Poplation_size = 100.
Max_Function_Evalution = 150,000.

Fig. 1. The Family Competition Genetic Algorithm.

3.4 Fitness function

We use formula (2) as basis of fitness function, these coefficients must be adjusted according to different strategies . We construct two algorithm on all testing problems. Algorithm 1 is based on the assumption that a feasible solution is always better than an unfeasible solution, that is , rejecting all unfeasible solutions. Algorithm 2 uses a non-stationary with multi-stages penalty function that depends on evolutionary generation number and level penalty, that is, the penalty coefficient is according to the violated size of the solution. Table 2 shows these parameters of fitness function of algorithm 2.

We have discussed the influence of strategy variables, including the crossover rate, the efficiency discrete crossover and BLX-0.5, family search length, initial standard derivation size and decreasing rate[27]. We discovered that the hybrid discrete crossover and BLX-0.5 with probability 0.8 and 02 is best and the BLX-0.5 is the worst. Other parameters are shown as Fig. 1.

4: Experimental results

In the section, we apply the FCGA approach to some general constrained optimization problems. We expected that the collection of problems presented here was used for purposes of comparing various evolutionary algorithm for general nonlinear programming. These problems have been studied by some authors[18,13,14,11,28] . When collecting the eight problems we consider many important factors such as the type of the objective function, the number of variables, the number of constrains, the types of constraints and the practical factor in application domain. Table 1 summaries all test cases. Michalewicz[18] used six kinds of evolutionary algorithms to solve problem #1 to problem #5 and the results were shown in Table 3. Homaifar[11] used traditional GAs to studied the problem #6 to problem #8. (there are some mistakes for problem #7 and problem #8 in their paper). The strategy parameters, including the population size, terminal conditions, decreasing rate of the step size, and the initial step size, are defined in Fig. 1. Table 2 shows the setting parameters for each testing problem when they are based on the non-stationary and multi-stage assignments weight penalty function.

Problems #1:

This problem was studied by Michalewicz [18] comparing the performance of various evolutionary algorithms.

$$C_1(X) = 5(x_1 + x_2 + x_3 + x_4) - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i ,$$

subject to

$$2x_1+2x_2+x_{10}+x_{11} \le 10,\ 2x_1+2x_3+x_{10}+x_{12} \le 10,$$
$$2x_2+2x_3+x_{11}+x_{12} \le 10,\ -8x_1+2x_{10} \le 0,\quad -8x_2+2x_{11} \le 0,$$
$$-8x_3+2x_{12} \le 0,\ -2x_4-x_5+x_{10} \le 0,\ -2x_6-x_7+x_{11} \le 0,\ -2x_8-x_9+x_{12} \le 0,$$
$$0 \le x_i \le 1, i=1,\ldots,9,\quad 0 \le x_i \le 100, i=10,11,12$$

Problems #2:

The problem is taken from Hock[14] (problem no is 106) and was studied by Michalewicz [18] to compare the performance of various evolutionary algorithms.

$$C_2(X) = x_1 + x_2 + x_3 ,$$

subject to

$$1-0.0025(x_4+x_6) \ge 0,\ 1-0.0025(x_5+x_7-x_4) \ge 0,$$
$$1-0.001(x_8-x_5) \ge 0,\ -x_1x_6-833.33252x_4-100x_1+8333.333 \ge 0,$$
$$x_2x_7-1250x_5-x_2x_4+1250x_4 \ge 0,\ x_3x_8-1250000-x_3x_5+2500x_5 \ge 0,$$
$$100 \le x_1 \le 10000,\ 1000 \le x_i \le 10000, i=2,3,\ 10 \le x_i \le 1000, i=4,\ldots,8.$$

Problems #3:

The problem is taken from Hock[14] (problem no is 100) and was studied by Michalewicz [18] to compare the performance of various evolutionary algorithms.

$$C_3(X) = (x_1-10)^2 + 5(x_2-12)^2 + x_3^4 + 3(x_4-11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 ,$$

subject to

$$127 - 2x_1^2-3x_2^4-x_3-4x_4^2-5x_5 \ge 0,\ 282-7x_1-3x_2-10x_3^2-x_4+x_5 \ge 0$$
$$196 - 23x_1-x_2^2-6x_6^2+8x_7 \ge 0,\ -4x_1^2-x_2^2+3x_1x_2-2x_3^2-5x_6+11x_7 \ge 0$$
$$-10.0 \le x_i \le 10.0, i=1,\ldots,7.$$

roblems #4:

The problem is taken from Hock[14] (problem no is 80) and was studied by Michalewicz [18] to compare the performance of various evolutionary algorithms.

$$C_4(X) = e^{x_1x_2x_3x_4x_5},$$

subject to

$$x_1^2+x_2^2+x_3^2+x_4^2+x_5^2 = 10,\ x_2x_3-5x_4x_5 = 0,\quad x_1^3+x_2^3 = -1$$
$$-2.3 \le x_i \le 2.3\ ,i=1,2,\quad -3.2 \le x_i \le 3.2\ ,i=3,4,5$$

Problems #5:

The problem is taken from Hock[14] (problem no is 113) and was studied by Michalewicz [18] to compare the performance of various evolutionary algorithms.

$$C_5(X) = x_1^2+x_2^2+x_1x_2-14x_1-16x_2+(x_3-10)^2+4(x_4-5)^2+$$
$$(x_5-3)^2+2(x_6-1)^2+5x_7^2+7(x_8-11)^2+2(x_9-10)^2+(x_{10}-7)^2+45,$$

subject to

$$105-4x_1-5x_2+3x_7-9x_8 \ge 0,\ -10x_1+8x_2+17x_7-2x_8 \ge 0,$$
$$-3(x_1-2)^2-4(x_2-3)^2-2x_3^2+7x_4+120 \ge 0,\ -5x_1^2-8x_2-(x_3-6)^2+2x_4+40 \ge 0,$$
$$-x_1^2-2(x_2-2)^2+2x_1x_2-14x_5+6x_6 \ge 0,\ -0.5(x_1-8)^2-2(x_2-4)^2-3x_5^2+x_6+30 \ge 0,$$
$$3x_1-6x_2-12(x_9-8)^2+7x_{10} \ge 0,\ -10.0 \le x_i \le 10.0,\quad i=1,\ldots,10$$

Problems #6:

This problem that is taken from Colvilles is cited by Hock[14] (problem no is 14), Himmelblau[25] (problem no is 1) and was studies by Homaifar[11] and by Fogel [28] using traditional GA and evolutionary programming, respectively.

$$C_3(X) = (x_1-2)^2 + (x_2-1)^2$$

subject to

$$x_1 = 2x_2 - 1,\quad -x_1^2/4 - x_2^2 + 1 \ge 0$$

Problems #7:

This problem that is taken from Colvilles is cited by Hock[14] (problem no is 87) and was studies by Homaifar[11] using traditional GA.

$$C_6(X) = f_1(X) + f_2(X)$$

$$f_1(X) = \begin{cases} 30x_1 & 0 < x_1 < 300 \\ 31x_1 & 300 < x_1 < 400 \end{cases} \qquad f_2(X) = \begin{cases} 28x_2 & 0 < x_2 < 100 \\ 29x_2 & 100 \le x_2 < 200 \\ 30x_2 & 200 \le x_2 < 1000 \end{cases}$$

$$x_1 = 300 - (x_3x_4\cos(1.48477) + 0.90798x_3^2\cos(1.47588))/131.078$$
$$x_2 = (-x_3x_4\cos(1.48477+x_6) + 0.90798x_4^2\cos(1.47588))/131.078$$
$$x_5 = (-x_3x_4\sin(1.48477+x_6) + 0.90798x_4^2\sin(1.47588))/131.078$$
$$200 - (x_3x_4\sin(1.48477-x_6) - 0.90798x_3^2\sin(1.47588))/131.078 = 0$$

subject to

$$0 \le x_1 \le 400,\ 0 \le x_2 \le 1000,\ 340 \le x_3,\ x_4 \le 420,\ -1000 \le x_5 \le 1000,\ 0 \le x_6 \le 0.5236$$

Problems #8:

This problem that is taken from Colvilles is cited by Hock[14] (problem no is 83) and was studies by Homaifar[11] and by Fogel [28] using using traditional GA and evolutionary programming, respectively.

$$C_7(X) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

subject to

$$0 \le 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 \le 92,$$

$$90 \le 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \le 110,$$

$$20 \le 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \le 25,$$

$$78 \le x_1 \le 102, \quad 33 \le x_2 \le 45, \quad 27 \le x_3, x_4, x_5 \le 45$$

Table 1 The summary of the testing problems

note1 : LI(Linear Inequality), LE(Linear Equation),NI(Nonlinear Inequality) and NE(Nonlinear Equation)

note2: There are some mistakes in reference [11] for problem 7 and problem 8.

pro #	Variables and Function type	Con. type	The x value of optimal solution	reference optimal solution	reference
1	13 quadratic	9(LI)	(1,1,1,1,1,1,1,1,1,3,3,3,1)	-15	[13,18]
2	8 linear	3(LI) 3(NI)	(597.3167,1359.943,5110.071, 182.0174,295.5985,217.9799, 286.4162,395.5979)	7049.3309	[14,18]
3	7 polynomial	4(NI)	(2.330499,1.951372, -0.4775414,4.365726, -0.624487,1.038131,1.594227)	680.6300	[14,18]
4	5 nonlinear	3(NE)	(-1.717143,1.595709,1.827247, -0.7636413,-0.7636450)	0.05394984	[14,18]
5	10 quadratic	3(LI) 5(NI)	(2.171996,2.363683,8.773926, 5.095984,0.9906548,1.430574, 1.321644,9.828726,8.280092, 8.375927)	24.3062	[14,18]
6	2 quadratic	1(LE) 1(NI)	(0.8228756,0.9114378)	1.3934651	[11,14]
7	6 polynomial	4(NE)	(201.78,100,383.07,420,-10907,0.07314)	8953.44	[11,14]
8	5 polynomial	3(NI)	(78,33,29.99526,45,36.77581)	-30665.538	[11,14]

Table 2: The parameters of fitness function (formula (2)) of algorithm 2 for the eight experimental problems

pro #	γ	θ	η
1,2,3,5,8	*if* $p_i < 1$ then θ=1 else θ=2	*if* $p_i < 0.01$ then θ=10 *else if* $p_i \le 0.1$ then θ=20 *else if* $p_i \le 1.0$ then θ=100 *else* θ=300	g*sqrt(g)
6			sqrt(g)
4		1	0.0025*g
7		1	g

Table 4 indicates these experimental results, including the best solution, mean value and standard deviation. Fig. 2 and Fig. 3 show the typical convergent curve of algorithm 1 (based on the assumption that of feasible solutions are always better than unfeasible solution) and algorithm 2 (based on non-stationary penalty function). Algorithm 1 has poor performance for problem #4 and problem #7 because two problems have equality constraints. Rejecting the unfeasible solution causes algorithm 1 premature when the ratio, feasible solution / search space, is very small. Algorithm 2 can overcome the drawback of algorithm 1 but the parameters depend on problem. Our FCGA approach is actually a modification of GAs and ESs, thus we compare our FCGA approach with various GA-based approaches and EP method according to Table 3 and Table 4. Our FCGA approach outperforms various GA-based methods and EP.

Table 3 The results were obtained from [18] (The violated tolerance is 0.001)

method #1 uses GA and level-based penalty function[11]
method #2 uses GA and non-stationary penalty function[15]
method #3 uses GA and non-penalty function approach
method #4 uses GA and penalty function
method #5 uses GA and the assumption of feasible solution better than an unfeasible solution
method #6 uses EP and rejects the unfeasible solution

pro #	method #1	method #2	method #3	method #4	method #5	method #6
1	-15.002*(b) -15.002*(m) -15.001*(w)	-15.00(b) -15.00(m) -14.999(w)	-15.00(b) -15.00(m) -14.998(w)	-15(b) -15(m) -15(w)	-15(b) -15(m) -14.999(w)	-15.00(b) -14.999(m) -13.616(w)
2	2282.723* 2449.789* 2756.679*	3117.242* 4213.497* 6056.211*	7485.667 8271.292 9752.412	7377.976 8206.151 9652.901	2101.367* 2101.411* 2101.551*	7872.948 8559.423 8668.648
3	680.771 681.262 689.660	680.787 681.111 682.798	680.836 681.175 685.540	680.642 680.718 680.955	680.805 682.682 685.738	680.847 681.826 689.417
4	0.084 0.955 1.0	0.059 0.812 2.542	#	0.054 0.064 0.557	0.067 0.091 0.512	#
5	24.690 29.258 36.060	25.486 26.905 42.356	#	18.917* 24.418* 44.302	17.388* 22.932* 48.866	25.653 27.116 42.477

*: the solution violates some constraints
#: the method was not applied to the problem.

Table 4 Experimental results based on 10 runs

algorithm 1: based on the assumption that feasible solutions are always better than unfeasible solutions
algorithm 2: based on non-stationary with multi-stage penalty function
vv: the sum of the value of violated constraints.
note1: The violated tolerance is 0.00001 for algorithm 2 and 0 for algorithm 1

		algorithm 1			algorithm 2		
pro #	optimal solution	best solution	mean	Std. Dev.	best solution(vv)	mean(vv)	Std. Dev.
1	-15	-15	-15	0	-15 (0)	-15 (0)	0
2	7049.3309	7383.589	7488.04	78.699	7176.176 (0)	7243.535 (0)	49.90
3	680.6300	680.6312	680.6335	0.0022	680.631 (0)	680.6333 (0)	0.003165
4	0.053949	0.9875	1.0	0.01677	0.054 (0)	0.054 (0)	0
5	24.3062	24.3161	24.3239	0.0132	24.319 (0)	24.33673 (0)	0.02240
6	1.3934651	1.393464	1.39346	0	1.393465 (0)	1.393464 (0)	1.39346
7	8953.44	8989.897	9008.011	15.1520	8857.34 (0.0001)	8879.184 (0.000075)	14.2518
8	-30665.538	-30665.5	-30665.5	0	-30665.538 (0)	-30665.538 (0)	0

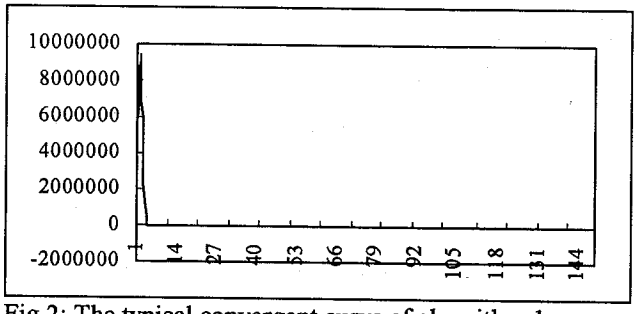

Fig 2: The typical convergent curve of algorithm 1

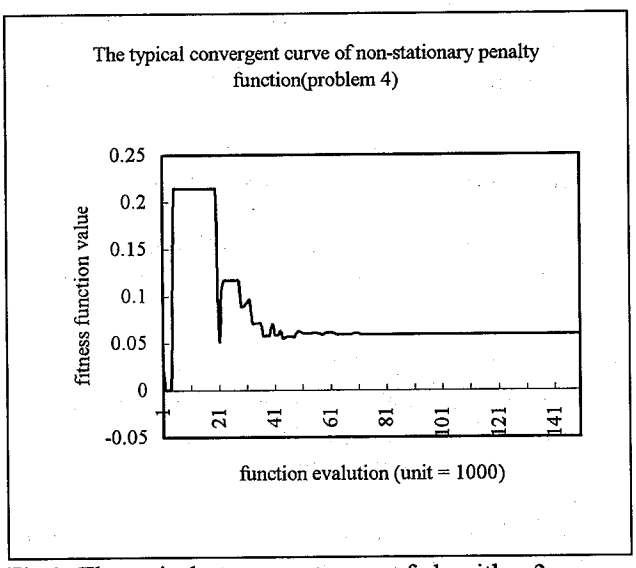

Fig 3: The typical convergent curve of algorithm 2

5 Discussions and conclusions

We obtain some important facts according to the experimental results.

1). FCGA outperforms other evolutionary algorithms: The FCGA technique combines the philosophy of ESs,GAs and family competition. As demonstrated with the use of eight general constrained global optimization problems, the FCGA outperformed other evolutionary algorithms, including the EP, ESs, traditional GA and real-coded GA. These results indicate that the FCGA is a powerful optimization technique.

2). The non-stationary and level penalty strategy is very useful to search the exact and feasible global optimization solution and has better flexibility to tune the parameter.

3). Penalty function is very important for constrained optimization with equality constraints: Algorithm 1 has very poor performance on problem #4 and problem #7 because the two problem have equality constraints. This indicates that rejecting the unfeasible is not fit the equality constraint and optimizing equality constrained problems must rely on the information of unfeasible solution. Oppositely, algorithm 2 can get very closed to global optimization solution.

4). Generally, strategy with non-stationary penalty function is better than strategy with assumption that feasible solutions are always better than unfeasible solutions: Is very obvious that algorithm 2 is better than algorithm 1 in all test problems. Above all, the algorithm 1 can not find good performance of the problem #2 that does not have any equality constraints, but algorithm 2 can find better solution.

5). Tuning the parameters of penalty function is troublesome because they depend on characteristics of the problem.

The main search power of FCGA comes from four sources as follows:

1).The family competition preserves the advantages of ESs, reduces the computing complexity, keeps the diversity of population, and avoid premature trouble.

2). The family competition can be viewed as local competition and selection viewed as global competition. Therefore, FCGA has two level competitive power.

3). The crossover operator searches with a larger jump than Gaussian mutation so that it can search the approximate optimal area, yet it is not suitable for tuning solution when the solution is near the optimal.

4). The Gaussian mutation operator has excellent ability for tuning the solution.

The traditional GAs rely on crossover and inversion operators. These operators usually destroy the link between the parents and children. In other words, these operators focus more on exploration and less on exploitation. This policy may causes GAs that do not suit to local search. The EP focuses on controlling the step size of Gaussian mutation only. This policy implies that EP emphasizes more on exploitation and less exploration. FCGA is able to balance the exploration and exploitation dynamically. Table 5 describes the behavior and main operators of FCGA

Table 5 The behavior and main operators between exploration and exploitation in FCGA search

Search Time	Early time		Advanced time	
	behavior	operators	behavior	operators
Exploration	High	1 big step size Gaussian mutation	Low	1 small step size Gaussian mutation
Exploitation	Low	2 crossover on low information and high diversity population	High	2 crossover on high information and low diversity population

References

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]. 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.

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

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

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

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

[10]. J.M. Yang and C.Y. Kao, "A Combined Simulated Evolutionary Algorithm for Real Parameter Optimization," Proceeding of the IEEE ICEC 1996 , pp 732-737.

[11] A. Homaifar, A.H.-Y Lai and X. Qi "Constrained Optimization via genetic algorithms," Simulation, vol.62, no. 4, 1994, pp242-254.

[12] Z. Michalewicz, "A survey of constrain handing techniques in evolutionary computation methods," proceeding of the 3rd annual conference on Evolutionary programming, 1994 ,pp98-108

[13] C.A. Floudas, C.A. and P.M. Pardalos, " A collection of test problems for constrained global optimization algorithm," Springer-Verlang, Lecture Notes in Computer Science, 1987, vol. 455.

[14] W. Hock and K. Schittkowski, "Test examples for Nonlinear Programming Codes," Springer-Verlag, Lecture Notes in Economics and Mathematical systems, 1981, vol. 187.

[15] J.A. Joines and C.R. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs, Proceeding of the IEEE ICEC 1994, pp. 579-584.

[16] D. Powell and M.M. Skolnick, "Using genetic algorithms in engineering design optimization with non-linear constraints," ICGA, 1993, pp. 424-430.

[17] M. Schoenauer and S. Xanthakis, "Constrained GA optimization,", ICGA 1993, pp 573-580

[18] Z. Michalewicz, "Genetic Algorithms, Numerical Optimization, and Constrains," ICGA, 1995, pp.151-158.

[19] R.G.Reynolds, Z. Michalewicz and M.Cavaretta, "Using cultural algorithm for constraint handing in Genocop," proceedings of the 3rd annual conference on Evolutionary programming 1995 ,pp289-305.

[20]. J.H. Holland, Adaptation in natural and artificial systems, MIT press, 1992.

[21]. J.T. Richardson, M.R. Palmer, G. Liepins and M. Hilliard, "Some Guidelines for Genetic Algorithms with Penalty Functions," ICGA 1989, pp 191-197.

[22] D. Orvosh, and L. Davis, " Shall We Repair? Genetic Algorithms, Combinatorial Optimization, and Feasibility Constraints," ICGA 1993, pp 650.

[23] J. Paredis, "Co-evolutionary Constraint Satisfaction," PPSN 1994, Springer-Verlag, pp 46-55.

[24] Z. Michalewicz and C. Janikow, "Handling Constraints in Genetic Algorithms," ICGA 1991, pp 151-157.

[25] D.M. Himmelblau, Applied nonlinear programming , Mcgraw-Hill, New York, 1972.

[26]. H. Myung and J.H. Kim, "Constrained Optimization Using Two-Phase Evolutionary Programming," Proceeding of the IEEE ICEC 1996 , pp 262-267.

[27]. J.M. Yang and C.Y. Kao, "A Family Competition Genetic Algorithm for high dimensional Function Optimization ," prepared to submit.

[28] D.B.Fogel, "A Comparison of Evolutionary Programming and Genetic Algorithm on selected Constrained Optimization Problems," Simulation 1995, June, pp 397-404.