Structural Optimization by Real-Coded Probabilistic Model-Building GA

Tomoyuki Hiroyasu, Mitsunori Miki, Hisashi Shimosaka,Yusuke Tanimura

Department of Knowledge Engineering and Computer Sciences, Doshisha University
1-3 Tatara Miyakodani Kyotanabe, Kyoto, 610-0321, Japan
Email:tomo@is.doshisha.ac.jp

Yasunari Mimura,Shinobu Yoshimura

Institute of Environmental Studies Graduate School of Frontier Sciences The University of Tokyo
Eng. Building Nr.12,Asano Campus, 7-3-1 Hongo, Bunkyo, Tokyo 113-8656
Email:mimura@garlic.q.t.u-tokyo.ac.jp

Abstract— **In this paper, probabilistic model-building genetic algorithm (PMBGA) is applied to structural optimization problems. PMBGA has high searching ability but it sometimes converges to the local minimum. To avoid this problem, the concept of distributed GA is applied to PMBGA. To deal with constraints, penalty function and pulling back methods are also applied to PMBGA. Using the proposed methods, a truss structure is designed to minimize its volume as a numerical example. Through the numerical example, the comparison between PMBGA and conventional DGA shows the effectiveness of PMBGA. The penalty function and pulling back methods are also effective in the example.**

Keywords—**Real-Coded Genetic Algorithm,Structual Optimization, Probabilistic Model-Building GA**

I. Introduction

Genetic algorithm (GA) [1][2] that simulates creatures' heredity and evolution is one of optimization tools. GA is a stochastic searching method and is also one of multi searching point methods. It is said that GA is good at finding a global optimum and it is easy to apply to several types of problems. In this paper, structural optimization problems are tried to be solved by GA.

Because of the rapid progress in computers, computer simulation of structures such as cars, planes, buildings and so on helps structural designers. Optimization is another advanced tool for designers. When designer formulate their problems with objective function, constraints and design variables, designers can make their decision by computational optimization. In structural optimization problems, the problems have the following four characteristics. Firstly, object function has not only a global optimum but also several local optimums. Secondly, they have several types of constraints. Thirdly, usually there are a lot of design variables. Finally, because of a large number of design variables and constraints, the feasible region is very narrow compared to design field. Therefore, for structural optimization problems, optimization method that has high searching ability and has mechanisms to deal with constraints should be prepared.

Recently, a new type of GA that is called "Probabilistic Model-Building GA (PMBGA)" has been focused [8]. In simple GA, children are generated from the parents and these parents are selected randomly. However, in PMBGA, the good characteristics of parents are forced to inherit to children using statistical information. Since children must have parents' characteristics, the effective searching is expected.

GA is an algorithm that cannot treat constraints explicitly. Therefore, some mechanisms of treating constraints have to be implemented into GAs. Some GAs have been introduced for constrained problems[10], [11], [12], [13], [14], [15]. Usually, when constraints are violated, some penalty values are reduced from the fitness function. This method is called a penalty function method. However, it sometimes happens that almost all the initial individuals violate the constraints when the feasible region is very narrow. In this case, the effective search cannot be expected. In this paper, the other method is introduced. In this method, when an individual violates the constraints, this individual is moved to the point that satisfies the constraints. This method is called "Pulling Back Method".

Penalty method and pulling back method is implemented into PMBGA and PMBGA is applied to solve truss structure problems. Truss structure is a simple structure, but the tendency of the results holds true for the other complicated structure problems. In the numerical example, the results of PMBGA are compared with those of sequential quadratic programming (SQP) and conventional distributed GA (DGA). Through the effectiveness of PMBGA, SQP and DGA in structural optimization problems is discussed. At the same time, capability of penalty function and pulling back methods are demonstrated.

II. Real-Coded Probabilistic Model Building Genetic Algorithm

In this paper, we introduce a real-coded probabilistic model-building GA (Real-coded PMBGA). In this model, PCA transformation is utilized. Then, we pro-

pose a distributed population model of PMBGA.

A. Outline of PMBGA

In this section, PMBGA is explained briefly.

In this paper, the concept of probabilistic model-building GA (PMBGA)[8]is used. Fig. 1 show the concept of PMBGA.

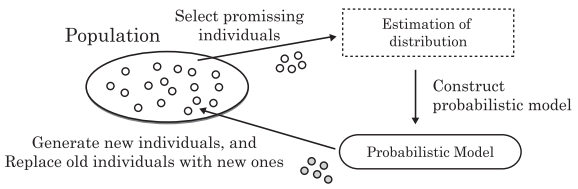

Fig. 1. PMBGA

In PMBGA, some of individuals are chosen at first. Usually, the individuals whose fitness values are high are chosen. Using these individuals, the probabilistic model is prepared. By this probabilistic model, children are generated stochastically. Then these new individuals are evaluated and some new individuals are substituted for old individuals. The procedure is returned to the first step. Finally, the optimum solutions can be found after the iterations. This is the general flow of PMBGA. Since the probabilistic model is constructed with the good parents, it is expected that the generated children are also good. Because of this mechanism, the optimum solution can be derived quickly.

B. Real-coded Probabilistic Model Building GA

In this paper, real-coded probabilistic model building GA (Real-coded PMBGA) is applied.

In a simple GA, design variables are coded by binary coding, gray coding and so on. The searching point before the coding is called a phenotype and the searching point after the coding is called a genotype. Then, an optimum solution is searched by genetic operations using these coded design variables. However, it is reported that it is difficult to find the high accuracy solution in continuous problems[3]. At the same time, the landscape of objective function is totally different from the one before the coding and the one after the coding. Thus, when users are searching optimum and the problems are continuous, it is easy to understand the landscape of the problem in a real value space. This is one of the motivations that users choose GAs where design variables are not coded but treat real values. These types of GAs are called "Real-coded GAs" and several real-coded GAs have been introduced[4], [5].

In real coded GAs, different crossover and mutation operations are applied to generate children from parents. In GAs, to find a good solution, it is important to produce children that inherit the good parts of their parents. To produce good children, it is significant to design an excellent crossover mechanism. In a crossover operation that is one of genetic operations, usually one point crossover, two point crossover and uniform crossover are applied. These crossovers can generate children who have good parts of their parents. At the same time, there is a possibility for these crossovers to break the good parts of the parents and inherit them to the children[6]. In this paper, PMBGA is applied to real-coded GA. The following steps are real-coded PMBGA.

Step1: Initialization: Set the generation $t = 0$ and generate initial population $P(0)$ randomly. Evaluate all of the initial individuals.

Step 2: Probabilistic model construction: The probabilistic model $M(P)$ is constructed This model is explained precisely later.

Step 3: Generation new individuals: Using the probabilistic model M, the new individuals $O(t)$ are generated. $O(t) = M(P(t))$

Step 4: Mutation: Each design variable is determined to change as mutation with along to the mutation rate. When the design variable is decided to be changed, the value of design variable is changed randomly.

Step 5: Evaluation: Evaluate $O(t)$.

Step 6: Alternation of generations: Substitute some of $O(t)$ for $P(t)$ on the basis of the value of evaluation. This becomes a new population $P(t+1)$. Set $t = t+1$. In this algorithm, any generation model can be applied. In the following structural optimization problems, all of individuals in $O(t)$ are substitute into the individuals of $P(t)$ whose evaluation values are low.

Step 7: Terminal condition: When the terminal condition is satisfied, the searching is terminated. If the condition is not satisfied, return to Step 2.

C. Probabilistic model using PCA

In Real-coded PMBGA, a solution candidate is expressed as a real-coded vector. Therefore, there is no difference between genotype and phenotype. Because of this reason, it is easy to grasp of the distribution of individuals in real value space. The most important part of real-coded PMBGA is the mechanism of constructing probabilistic model and generating children.

In Fig.2, an example of the distribution of individuals is shown in the design field.

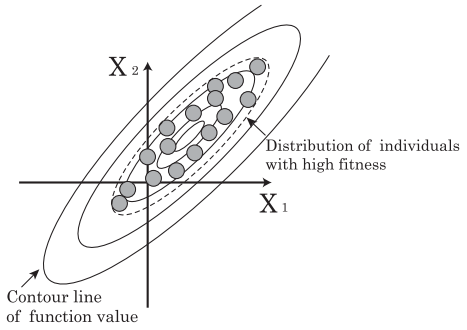

Fig. 2. Distribution of each induvidual

In this problem, the optimization function should be minimized and there are two design variables. Like this way, the landscape of the objective function can be illustrated with the individuals who have high fitness value.

When the objective function is influenced by design variables independently, the axis of the distribution of the individuals is parallel to the axis of the design variable. On the other hand, the objective function is influenced by design variables and those are related each other, the axis of the distribution of the individuals is not parallel to the axis of the design variable. Children should be generated with along to this distribution of parents.

In this paper, we construct the probabilistic model and generate children in the following way. In this procedure, Principal Component Analysis (PCA) is applied.

Step 1: Extract $A\%$ of the elite individuals from the population $P(t)$, where t indicates the generation. The extracted population is $E(t)$. When there are n design variables and m data, these data can be shown as $n \times m$ matrix $\mathbf{X} = \{x_{ij}\}\{i = 1...n, j = 1...m\}$.

Step 2: Derive the average of each design variable of $E(t)$. These are $\mathbf{X_{ave}} = \{x_{ij}^{ave}\}$. Then, subtract the average value from the data, $\mathbf{X} = \mathbf{X} - \mathbf{X_{ave}}$.

Step 3: Derive the covariance $\mathbf{S}$ of $\mathbf{X}$. $\mathbf{S} = \{s_{ij}\}$ can be derived from the following equation,

$$s_{ij} = \frac{1}{m-1}\Sigma_{k=1}^{m} x_{ik}x_{jk}$$

.

Step 4: Derive eigen values and vectors of $\mathbf{S}$. PCA transfer matrix $\mathbf{V}$ is consisted of the eigen vectors. The data $\mathbf{X}$ can be changed into no interrelated data $\mathbf{Y}$ using PCA transfer matrix $\mathbf{V}$ as follows,

$$\mathbf{Y} = \mathbf{V}\mathbf{X}$$

.

Step 5: In each design variable, random number is generated with accordance with the normal distribution. Then new data $\mathbf{Y'}$ are generated. The variance of used normal distribution is derived by the product of the variable B and the variance of each design variable of $\mathbf{Y}$. When the number of generated individuals is p, matrix $\mathbf{Y'}$ becomes $n \times p$.

Step 6: The generated data $\mathbf{Y'}$ reversed into the original field. At first, $\mathbf{X'}$ is deribed by the multiple of the inverse of $\mathbf{V}$ and $\mathbf{Y'}$.

$$\mathbf{X'} = \mathbf{V}^{-1}\mathbf{Y'}$$

Then, new data $\mathbf{X_{new}}$ are derived by the addition of $\mathbf{X'}$ and $\mathbf{X_{ave}}$.

These individuals become new population $O(t)$.

D. Distributed PMBGA

One of the disadvantages of PMBGA is a possibility of the convergence to the local minimum. To avoid this situation, the concept of distributed genetic algorithm (DGA) [7] is used. In DGA, a population is divided into sub populations. These sub populations are called islands. In each island, normal genetic operations are performed for several iterations. After some iterations, some individuals of islands are chosen and moved to the other island. This operation is called a migration.

Because the evolution can be proceeded in each island, DGA can keep the high diversity. It is reported that DGA has high searching ability compared to simple GA [7].

Therefore, when the concept of DGA is introduced into PMBGA, it is expected that the convergence to local minimum can be avoided.

In DGA, there are three variables in addition to the variables that are used in a simple GA; number of islands, migration interval and migration rate. The interval generation between the migration is called migration interval. At the migration operation, the number of individuals that moved to the other island is determined by the multiple of the number of the population in an island and the migration rate. When the DGA is applied to PMBGA, a population is divided into sub populations. In each sub population, PMBGA is performed for the migration interval. After the migration interval, some individuals are chosen randomly and moved to the other island. The topology of the island is a ring topology. This is shown in Fig. 3 and the topology is determined randomly at every migration chance.

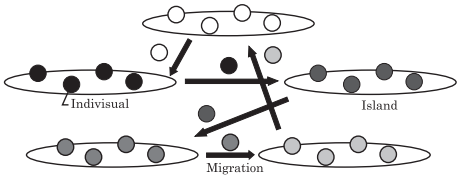

Fig. 3. Migration Topology

III. Structural Optimization Problem by GA

In this paper, structural optimization problem is solved by GA. Structural optimization problem has an objective function and several constraints. These are formulated as follows,

$$\begin{aligned} &minimize \quad f(\mathbf{x}) \\ &such\ that \quad g_j(\mathbf{x}) \leq 0 \quad j = 1..m \end{aligned}$$

.

Normally, compared to the design field, the feasible region is very small because of the constraints. Since GA cannot deal with constraints explicitly, the mechanism which deals with the constraints is important. In this paper, two methods are introduced and compared them through the numerical example.

A. Penalty function methods

GA is an optimization method that can not treat constraints explicitly. Therefore, many methods dealing with the constraints are proposed [10], [11], [12], [13], [14], [15]. Among them, penalty function method is the simplest method.

In this method, the objective function is modified when the constraint condition is violated. The modified function can be express as follows,

$$minimize \quad F_\rho(\mathbf{x}) = f(\mathbf{x}) + \rho(\Sigma_{i=1}^{m} max\{0, g_j(\mathbf{x})\})$$

where ρ is penalty parameter and ρ>0 should be satisfied.

Since this method is same as the normal GAs besides the fitness function, it is very easy to apply.

B. Pulling back method

Penalty function method is easy to apply for GAs. However, when the feasible region is very narrow compared to the design field, it sometimes happens that almost all the individuals violate the constraints. Usually a value of penalty function is greater than the objective function. Thus, the effective search cannot be performed. In this case, some individuals should be satisfied the constraints.

In this section, pulling back method is introduced. In Fig. 4, the concept of this movement is illustrated. In this method, when a searching point X_{out} violates constraints, the searching point is moved to the point X_{opt} that satisfies the constraints. In this figure, X_{opt} means the closest point from X_{out} that satisfies constrains. Therefore, if new individuals violate the constraints, they will satisfy the constrains by this operation. That is the reason this operation is called "Pulling back".

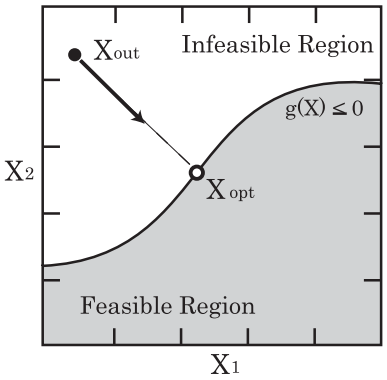

Fig. 4. Pulling back operation

To find the searching point X_{opt} that satisfies the constraints from the point X_{out}, the following optimization problem should be solved.

$$\begin{aligned} &minimize \quad f(\mathbf{x}) = \sqrt{(\mathbf{x}-\mathbf{x_{out}})^2} \\ &such\ that \quad \nabla g_j(\mathbf{x_{out}})(\mathbf{x} - \ -\mathbf{x_{out}}) + g_j(\mathbf{x_{out}}) \le 0 \\ &\qquad\qquad\qquad\qquad\qquad\qquad j = 1, ..., m \end{aligned}$$

In this problem, the objective function is the distance between the new and old points. The constraints are the original constraints but they are linearized. This is a quadratic problem and it can be solved by several methods, such as a sequential quadratic programming.

This pulling back operation is performed after the crossover and mutation. All the individuals that violate the constraints are renewed to satisfy the constraints.

IV. Numerical Example

In this section, a truss structure is designed by the proposed distributed PMBGA and handling methods of constraints. In this problem, the volume of the truss structure is minimized within the certain value of stress, displacement and buckling conditions. The comparison of the results by conventional DGA, the effectiveness of distributed PMBGA is described. At the same time, the characteristics of penalty function method and pulling back method are discussed.

A. Truss structure

The designed truss structure is shown in Fig. 5.

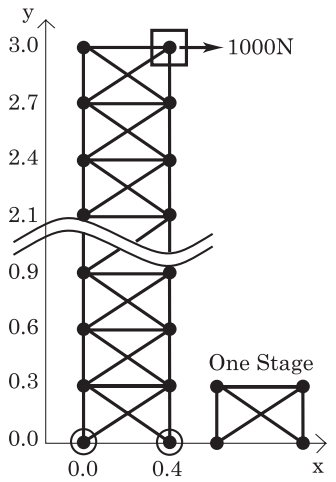

Fig. 5. 10 Stage Truss

This truss structure is consisted of 22 nodes and 50 members. Two nodes ((0.0,0.0) and (0.4,0.0)) at the ground are fixed. There is a load $N = 1000N$ at the top node ((0.4,3.0)). All the elements are linear elasticity and Young's modules is $1.000GPa$. The cross section of each element is a circle.

In this problem, the total volume of this structure is minimized. Design variables are areas of cross section of the element. There are three constraints. The displacement at the node that has the load should be less than a certain value (0.003 - 0.0015m). The maximum stress should be less than 2.000 MPa. Buckling should not be occurred in the structure.

B. SQP and conventional DGA

The results of PMBGA are compared with a gradient method and conventional DGA.

In this paper, sequential quadratic programming (SQP) is used as a gradient method. DOT code [9] is used for the SQP. The used parameters are summarized in Table I.

TABLE I
PARAMETER OF DOT

Method	SQP
RPRM[0]	-0.05 ?-0.108
(other) RPRM Array	ALL 0
IPRM Array	ALL 0
Maximum Value of Design Variables	0.1
Minimum Value of Design Variables	0.000001

TABLE II
PARAMETER OF CONVENTIONAL DGA

Number of Individuals	240
Tournament Size	4
Crossover Rate	1
Mutation Rate	0.000645
Number of Elite Individuals	1
Migration Rate	0.5
Migration Interval	1 Generation
Number of Islands	8
Terminal Criterion	1000 Generations
Maximum Value of Design Variables	0.1
Minimum Value of Design Variables	0.000001
Penalty Parameter (ρ)	1e+06

In conventional DGA, a bit string is used to express a design variable. 31 bits used for a design variable and a total length of gene is 1550 bits. The coding method is a gray coding. For selection, crossover and mutation, tournament selection, two point crossover and bit flip mutation are applied respectively. The migration method is same as distributed PMBGA. For conventional DGA, only the penalty function method is applied. The used parameters are summarized in Table II.

Penalty function method and pulling back method are applied to MBGA and results are compared. The used parameters are summarized in Table III.

C. Results

The truss structure is designed by DOT(SQP), conventional DGA with penalty function method, PMBGA with penalty method and PMBGA with pulling back method. Here, the value of displacement constraint is changed 0.0015, 0.0020, 0.0025 and 0.0030.

The volumes of the derived structures are summarized in Fig. 6. The number of function call of each method is summarized in Table IV. The transion of the total volume with respect to the generation is shown in Fig. 7. All the results are the average of 5 trials.

From these results, the following facts are made clarified.

When the constraints are loose, there are no big differences between the result of SQP and GAs. On the other hand, when the constraints are tight, the results of GAs are better than those of SQP. However, GA needs a lot of iterations. Therefore, GA is useful for applying the complicated and difficult problems.

Pulling back method needs more function calls than penalty function method. When an individual violates the constraints, it moves to the point that satisfies the constraints. At this moment, some function calls are needed. Therefore, this operation needs a lot of function calls.

Comparison between PMBGA and conventional DGA shows that PMBGA has a high searching ability. At the same time, PMBGA is a robust method to find the optimum since the proposed PMBGA has the distributed

TABLE III
PARAMETER OF PMBGA

Number of Individuals for pulling back method	80
Number of Individuals for penalty function method	240
PMBGA (A) for pulling back method	30%
PMBGA (A) for pulling back method	10%
PMBGA (B)	2
Mutation Rate for pulling back method	0.02
Mutation Rate for pulling back method	0.002
Number of Elite Individuals	1
Migration Rate	0.5
Migration Interval	1 Generation
Number of Islands	8
Terminal Criterion	1000 Generations
Maximum Value of Design Variables	0.1
Minimum Value of Design Variables	0.000001
Penalty Parameter (ρ)	1e+06

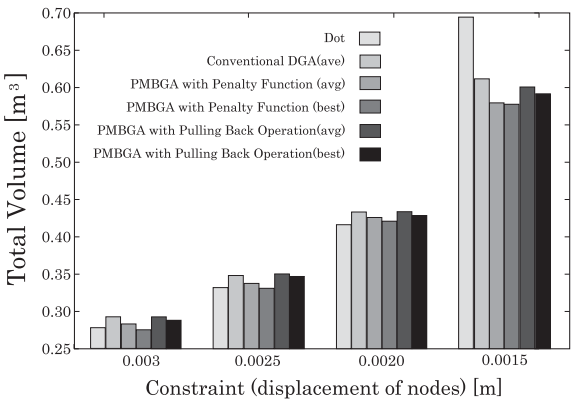

Fig. 6. Comparison of PMBGA with Pulling Back Operation,PMBGA with Penalty Function,Conventional DGA and DOT (Total Volume($m3$))

TABLE IV
COMPARISON OF PMBGA,CONVENTIONAL GA AND DOT
(NUMBER OF FUNCTION CALLS)

Constraints(m)	0.003	0.0025	0.002	0.0015
DOT	1214	1226	1214	472
Conventional DGA	240000	240000	240000	240000
PMBGA / Penalty	240000	240000	240000	240000
PMBGA / Pulling	1481017	1966673	1884030	2197135

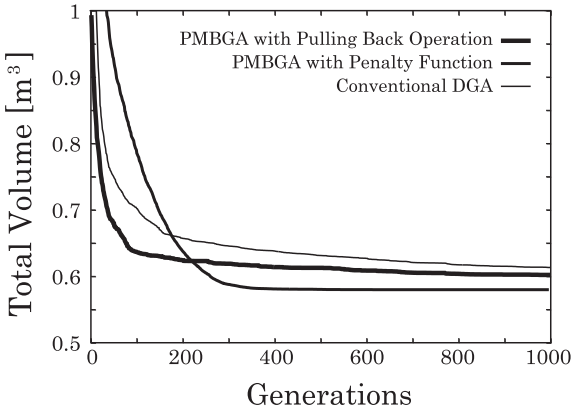

Fig. 7. Search History of PMBGA with Pulling Back Operation,PMBGA with Penalty Function and Conventional DGA

population concept.

In this problem, the result of penalty method is better than that of pulling back method. This is different from the expectation. In this problem, some of the initial individuals satisfy the constraints. Therefore, the search can be proceeded even by penalty function. It is still expected that pulling back method is superior to penalty function in the problems where all the individuals of initial population violate the constraints.

V. CONCLUSION

In this paper, probabilistic model-building genetic algorithm (PMBGA) is applied to structural optimization problems. For constraint optimization problems, penalty function method is usually applied to genetic algorithms. However, when the feasible region is very narrow compared to the design field, most of the individuals violate constraints and GA can not search an optimum effectively. To solve this problem, we introduce the pulling back method. In this method, the individual that violates constraints is moved to the point that satisfies the constraints. Since all of the points satisfy the constraints by this operation, the effective searching can be expected.

PMBGA has high searching ability but it sometimes converges to the local minimum. To avoid this problem, the concept of distributed GA is implemented PMBGA. In distributed GA, a population is divided into sub population and it can maintain the diversity.

The proposed method is applied to design a truss structure as a numerical example. Through the numerical example, the comparison of PMBGA with conventional DGA shows the effectiveness of PMBGA. However, the result of the pulling back method is not superior to that of penalty function method. This may come from the fact that the problem is rather easy to find an optimum. Solving much complicated problem is a future work.

Acknowledgments

This work was supported by a grant to RCAST at Doshisha University from the Ministry of Education, Science Sports and Culture, Japan.

REFERENCES

[1] Goldberg,D.E. *Genetic Algorithms in Search Optimization and Machine Learnig*, Addison-Wesley, 1989.

[2] Holland,J.H. *Adaptation In Natural and Artificial Systems*, University of Michigan Press,1975.

[3] Davis, L., The Handbook of Genetic Algorithms, Van Nostrand,1990.

[4] Larry J. Eshelman and Keith E. Mathias and J.David Schaffe, *Crossover Operator Biases: Exploiting the Population Distribution*, Proc. of the 7th International Conference on Genetic Algorithms,1997.

[5] Ono, I. and Kobayashi, S., *A Real-coded Genetic Algorithm for Function Optimization Using Unimodal Normal Distribution Crossover*, Proc. of the 7th ICGA,1997.

[6] Annie S. Wu and Robert K. Lindsay and Rick L. Riolo, *Emprical observation on the roles of crossover and mutation*, Proc. of the 7th International Conference on Genetic Algorithms,1997.

[7] Tanese,R. *Distributed Genetic Algorithms*, Proc. of the 3rd International Conference on Genetic Algorithms,1989.

[8] Pelikan,M. and Goldberg,D.E. and Fernando Lobo, *A Servey of Optimization by Building and Using Probabilistic Models*, illiGAL Report No.99018, 1999.

[9] DOT -Design Optimization Tools- , http://www.vrand.com/dot.htm

[10] Michalewicz,Z. and Janikow,C.Z. *Handling Constraints in Genetic Algorithms*, Proc. of the International conference on Genetic Algorithms 4,pp.151-157

[11] Hajela,P. and Yoom,J. *Constraint Handling in Genetic Search - A Comparative Study*, AIAA/ASME/ASCE/AHS Structures,Structural Dynamics and Materials Conference - Collection of Technical Papers 4,1995,pp.2176-2186

[12] Surry,P.D. and Radcliffe,N.J. *The COMOGA Method: Constrained Optimization by Multi-Objective Genetic Algorithms*, Control and Cybernetics 26(3),1997,pp.391-412

[13] Wright,J.A. and Farmani,R. *Genetic Algorithms: A fitness formulation for constrained minimization*, Proc. of the Genetic and Evolutionary Computation Conference,2001,pp.725-732

[14] Koziel,S. and Michalewicz,Z. *Evolutionary Algorithms, Homomorphours Mappings, and Constrained Parameter Optimization*, Evolutionary computation 7(1),1999,pp.19-44

[15] Tahk,M.-J. and Sun,B.-C. *Coevolutionary Augmented Lagragian Methods for Constrained Optimization*, Evolutionary computation 4(2),2000