Dumlupınar Üniversitesi Fen Bilimleri Dergisi

Sayı : 4 Ekim 2003

GENETIC ALGORITHMS IN ENGINEERING DESIGN OPTIMIZATION

Mühendislik Tasarım Optimizasyonunda Genetik Algoritmalar

Hamit SARUHAN*

ÖZET

Bu makale mühendislik tasarımında uygun bir optimum tasarım methodu olarak genetik algoritmaların kısa bir tanıtımını ve uygulamasını verir. Bu çalışmanın asıl amacı, genetik algoritmaların bir optimizasyon tekniği olarak potansiyelini ve uygulanabilme kabiliyetlerini bilyalı rulmanlara uyarlamakla göstermektir. Bilyalı rulmanlar için elde edilen sonuçlarla bu tekniklerin kabiliyeti gösterilmiştir. Genetik algoritmalar tabii seleksiyon (seçim) tekniğini kullanarak tanımlanan sınırlar içinde tarama yapan ve genetik fikrine dayalı uygun araştırma teknikleridirler. Gün geçtikçe genetik algoritmalar daha iyi tanınmakta ve bir çok alanda uygulanmaktadırlar.

Key Words: *Genetic algorithms, engineering design, optimization*
Anahtar Kelimeler: *Genetik algoritmalar, mühendislik tasarımı, optimizasyon*

1. INTRODUCTION

In engineering design, the goal is either to maximize or to minimize design objectives and to satisfy corresponding constraints. Many numerical optimization methods have been developed and used for design optimization of engineering problems. The development of faster computers has allowed development of more

* Abant İzzet Baysal Üniversitesi,Teknik Eğitim Fakültesi, Makine Eğitimi Bölümü, Düzce.

robust and efficient optimization methods. One of these methods is the genetic algorithms. Genetic algorithms are search procedures based on the idea of natural selection and genetics (Goldberg, 1998). Genetic algorithms can be applied to conceptual and preliminary engineering design studies. Interested reader can refer to the studies by (Saruhan et al., 2001) and (Saruhan et al., 2002). This paper shows how genetic algorithms search trough a design space to find the maximum value of the objective function for engineering design problems.

2. DESCRIPTION OF GENETIC ALGORITHMS

In this section of the paper, it is given the fundamental intuition of genetic algorithms and how they process. Genetic algorithms maintain a population of encoded solutions, and guide the population towards the optimum solution (Goldberg, 1989). Thus, they search the space of possible individuals and seek to find the best fitness string. Rather than starting from a single point solution within the search space as in traditional optimization methods, genetic algorithms are initialized with a population of solutions. Viewing the genetic algorithms as optimization techniques, they belong to class of zero-order optimization methods (Dracopoulos, 1997), (Louis, 1997).

The description of a simple genetic algorithm is outlined in Figure 1. An initial population is chosen randomly in the beginning. Then an iterative process starts until the termination criteria have been satisfied. After the evaluation of each individual fitness in the population, the genetic operators, selection, crossover and mutation, are applied to produce a new generation. Other genetic operators are applied as needed. The newly created individuals replace the existing generation, and reevaluation is started for fitness of new individuals. The loop is repeated until an acceptable solution is found.

Genetic algorithms differ from traditional search techniques in the following ways (Goldberg, 1989):

-Genetic algorithms work with a coding of design variables and not the design variables themselves.

-Genetic algorithms use objective function or fitness function information. No derivatives are necessary as in more traditional optimization methods.

-Genetic algorithms search from a set of points not a single point.

-Genetic algorithms gather information from current search points and direct them to subsequent search.

-Genetic algorithms can be used with discrete, integer, continuous, or a mix of these three design variables.

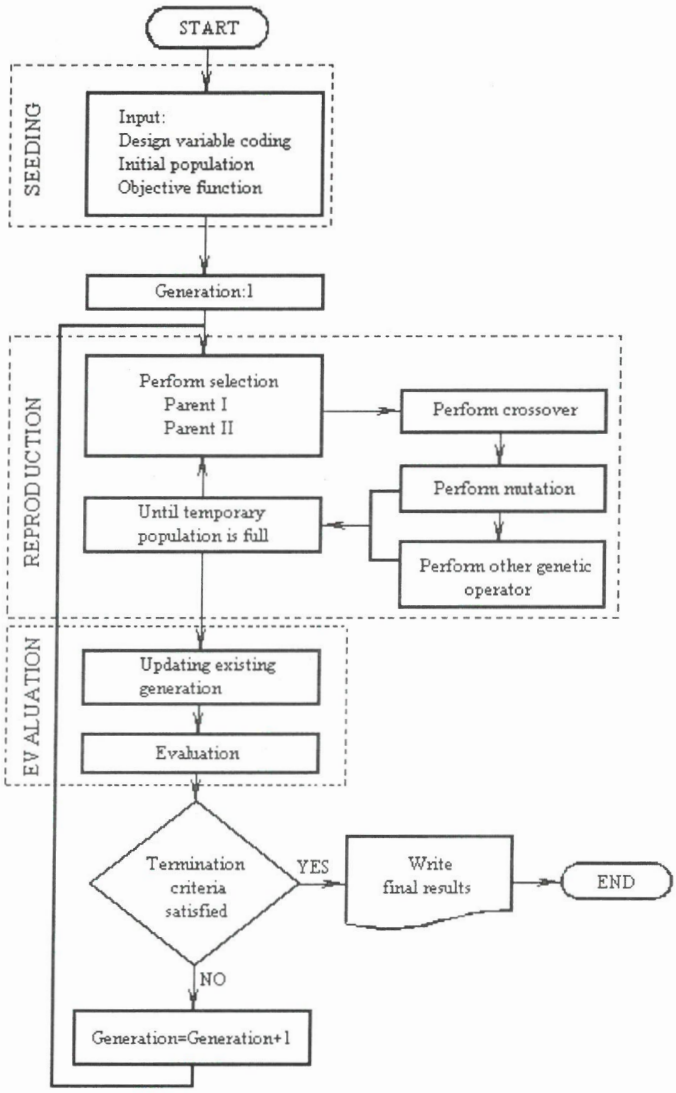

Figure 1. Flow chart for a simple genetic algorithm

3. PROBLEM STATEMENT

With the objective function given, a genetic algorithm is employed to find the maximum value of geometry factor for assigned range of diameter ratio, R_d, in the design of a ball bearing shown in Figure 2.

Selecting values of geometry factor, F_g, verses diameter ratio, R_d, is promising by third-order equation to fit tabulated results, reported in the publication (ANSI, 1978). This regression equation is suggested by (Wilson, 1997).

$$F_{Objective} = F_g(x) = \left(83940x^3 - 85850x^2 + 23710x + 2597\right) 0.013166 \quad (1)$$

where x is diameter ratio. $0.05 \le x \le 0.40$

Diameter ratio:

$$R_d = \frac{D \cos\alpha}{d_m} \quad (2)$$

where D is ball diameter; d_m is mean race diameter ($d_m = 0.5(d_o + d_i)$); d_o is outer race diameter (at ball contact); d_i is inner race diameter (at ball contact); α is normal contact angle (measured from the load line to the plane of the bearing).

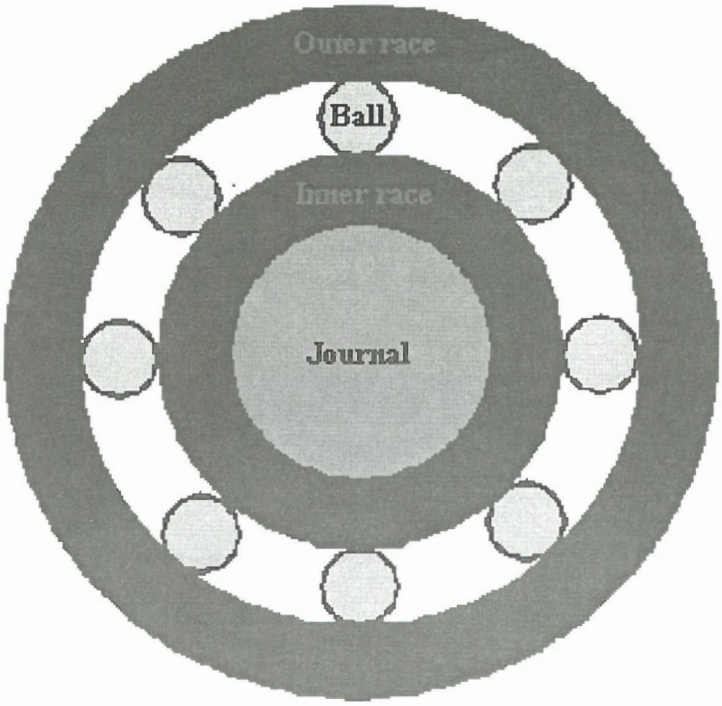

Figure 2. Schematic diagram of a ball bearing

Since this problem is an unconstrained maximization problem the objective function can be directly treated as the fitness function. Fitness function is one of most important aspects of genetic algorithms. Fitness function measures and rates the coded variable vectors in order to select the fittest strings that lead the solution.

$$Fitness_{Objective} = F_g(x) \tag{3}$$

3.1 Construction of Design variables and Genetic Algorithm Parameters

In optimization problem, a design of variables, $x(i)$, represents a solution that minimizes or maximizes an objective function. The first step for applying the genetic algorithms to assigned design problem is encoding of the design variables.

Genetic algorithms require the design variables of optimization problem to be coded as a finite length strings. These strings are represented as chromosomes. Each design variable has a specified range so that $x(i)_{lower} \le x(i) \le x(i)_{upper}$. The continuous design variables can be represented and discretized to a precision of ε ($\varepsilon = 0.01$). The number of digits in the binary strings, l, is estimated from the following relationship (Lin, 1992).

$$2^l \ge \frac{x(i)_{upper} - x(i)_{lower}}{\varepsilon} + 1 \tag{4}$$

where $x(i)_{lower}$ and $x(i)_{upper}$ are the lower and upper bound for design variable respectively. The design variables are coded into binary digit $\{0,1\}$.

The physical value of design variables, $x(i)$, can be computed from the following relationship (Wu, 1995):

$$x(i) = x(i)_{lower} + \frac{x(i)_{upper} - x(i)_{lower}}{2^l - 1} d(i) \tag{5}$$

where $d(i)$ represents the decimal value of string for design variable which is obtained using base-2 form.

To start the algorithm, an initial population is randomly generated. This set of initialized population is a potential solution to the problem. For example, the binary string representation for the design variable, x, in Figure 3 gives an example of a chromosome that represents diameter ratio.

Variable (x)						

Figure 3. The binary string representation of the variable

Figure 3 shows string of 6-bit string length represents one of 2^6 alternative individual solution existing in the design space. The genetic algorithm then proceeds by generating new solutions with bit operations utilizing genetic algorithm operators such as selection, crossover, and mutation. The solutions yield toward the optimum.

3.2 Genetic Algorithm Operators

In a simple genetic algorithm, there are three basic operators for creating the next generation. Each of these operators is explained and demonstrated in following:

The selection operator shown in this work is a tournament selection. Tournament selection approach works as follows: a pair of individuals from mating pool is randomly picked and the best-fit two individuals from this pair will be chosen as a parent. Each pair of parent creates two Child as described in the method of uniform crossover shown in Figure 4.

Crossover mask	**1**	**0**	**0**	**1**	**0**	**1**
Parent I	1	0	1	0	0	0
Parent II	0	1	0	1	0	1
Child I	1	1	0	0	0	0
Child II	0	0	1	1	0	1

Figure 4. Uniform crossover

A uniform crossover operator is used in this study. A uniform crossover operator probability of 0.5 is recommended in many works such as (Syswerda, 1989; Spears et al., 1991). Crossover is very important in the success of genetic algorithms. This operator is the primary source of new candidate solutions and provides the search mechanism that efficiently guides the evolution through the solution space towards the optimum. In uniform crossover, every bit of each parent string has a chance of being exchanged with corresponding bit of the other parent string.

The procedure is to obtain any combination of two parent strings (chromosomes) from the mating pool at random and generate new Child strings from these parent strings by performing bit-by-bit crossover chosen according to a randomly generated crossover mask (Beasly et al., 1993). Where there is a 1 in the crossover mask, the Child bit is copied from the first parent string, and where there is a 0 in the mask, the Child bit is copied from the second parent string. The second Child string uses the opposite rule to the previous one as shown in Figure 4. For each pair of parent strings a new crossover mask is randomly generated.

Preventing the genetic algorithm from premature convergence to a non-optimal solution, which may lose diversity by repeated application of selection and crossover operators, a mutation operator is used. Mutation operator is basically a process of randomly altering a part of an individual to produce a new individual by switching the bit position from a 0 to a 1 or vice versa as seen in Figure 5.

Before	1	0	1	0	**0**	0
After	1	0	1	0	**1**	0

Figure 5. Mutation operator

The mutation rate suggested by (Bäck, 1993) is:

$$\frac{1}{population\ size} < P_{mutation} < \frac{1}{chromosome\ length} \qquad (6)$$

Population size influences the number of search points in each generation. A guideline for an appropriate population size is suggested by (Goldberg, 1985). The guideline for optimal population size depends on the individual chromosome length, which is valid up to 60 expressed as follows:

$$population\ size = 1.65 * 2^{0.21 * l} \qquad (7)$$

For a string length of 6 bits, an optimal population size of 5 may be used (Goldberg, 1985). For this study, a randomly selected set, 10 strings, of potential solution is used as shown in Table 1.

Table 1. A set of starting population

Individual number	Randomized binary string x
1	0 1 1 0 1 1
2	0 1 0 1 0 1
3	1 1 0 1 0 1
4	0 1 0 1 0 0
5	1 1 0 1 1 1
6	0 1 0 0 0 1
7	1 1 0 1 0 1
8	0 1 0 1 0 0
9	0 1 1 1 0 1
10	1 1 0 1 0 1

A specialized mechanism, *elitism*, is added to genetic algorithm. Elitism forces the genetic algorithm to retain the best individual in a given generation and proceed unchanged into the following generation (Mitchell, 1997). The parameters of genetic algorithm for this study have chosen as follows:

Table 2. Genetic search algorithm parameters

Genetic algorithm parameters	
Chromosome length	6
Population size	10
Number of generation	300
Crossover probability	0.5
Mutation probability	0.01
Elitism	Yes

There are many different ways to determine when to stop running the genetic algorithm. One method is to stop after a preset number of generation or a time limit. Another is to stop after the genetic algorithm has converged. Convergence is the progression towards uniformity. A string is said to have converged when 95 % of the population share the same value (De Jong, 1975). Thus, most or all strings in the population are identical or similar when population is converged.

4. RESULTS

The distribution of normalized fitness function values for the objective function in generation number 1, 100, and 300 is given in Figure 6. As can be seen from plot, the normalized fitness function of individuals in a population improves over generations. Figure 7 shows the plots of the normalized average and best fitness function values in each generation as optimization proceeds. In Figure 7, the overall results show that the best design rapidly converge over the first several generations and refine the design over remaining generations. Thus, the selected parameters set have converged to stable solutions with similar values.

Substituting the diameter ratio of 0.19 for the geometry factor in Equation 1 yields to the value of 60.24. A plot of geometry factor versus diameter ratio obtained from closed form of Equation1 is shown in Figure 8. As shown in Figure 9, the peak of the graph has a significant share of the population strings; they have converged uniformly through generations for maximum value, 60.24, of geometric factor for diameter ratio of 0.19. As can be seen from the results of this example, genetic algorithms are efficient techniques able to find the maximum value of the design objective.

5. CONCLUSIONS

The main goal of this work is to motivate and give an idea to designers who are not currently exposed to the potentials of these algorithms. In this regard, the efficacy of genetic algorithm optimization techniques is demonstrated by employing an engineering design problem. It can be concluded that the genetic algorithms can be successfully used for conceptual and preliminary design optimization of engineering problems.

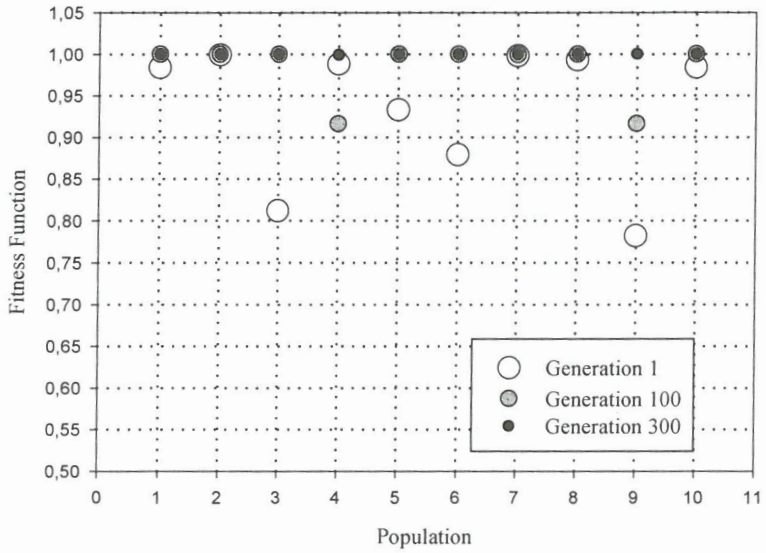

Figure 6. Distribution of normalized fitness function values for generation number one, one-hundred, and three-hundred

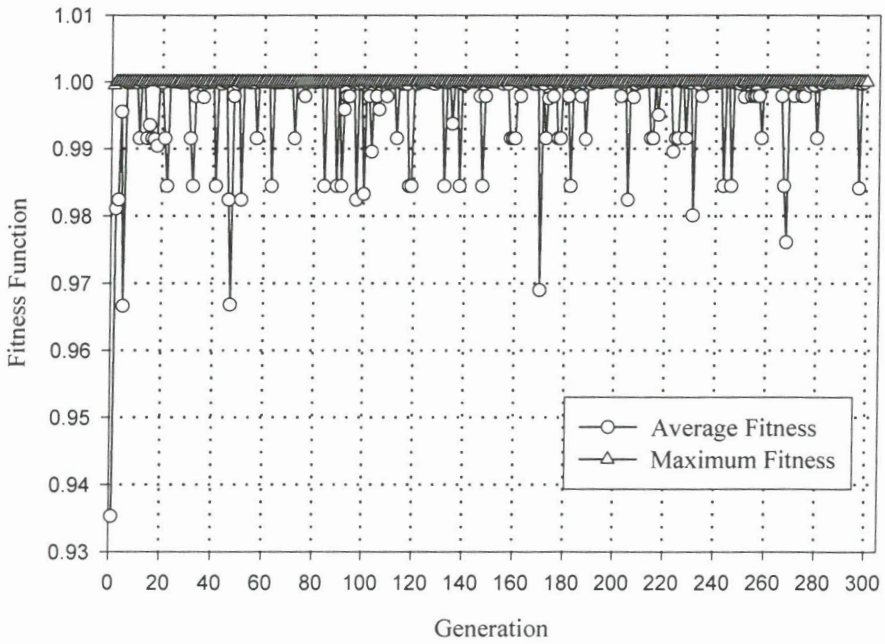

Figure 7. Convergence process of genetic algorithm for average and best fitness function of the objective function

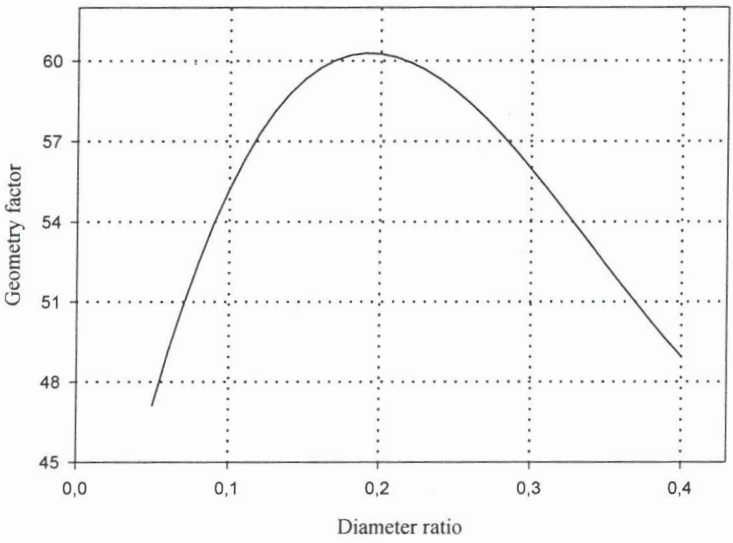

Figure 8. Geometric factor for basic loa rating

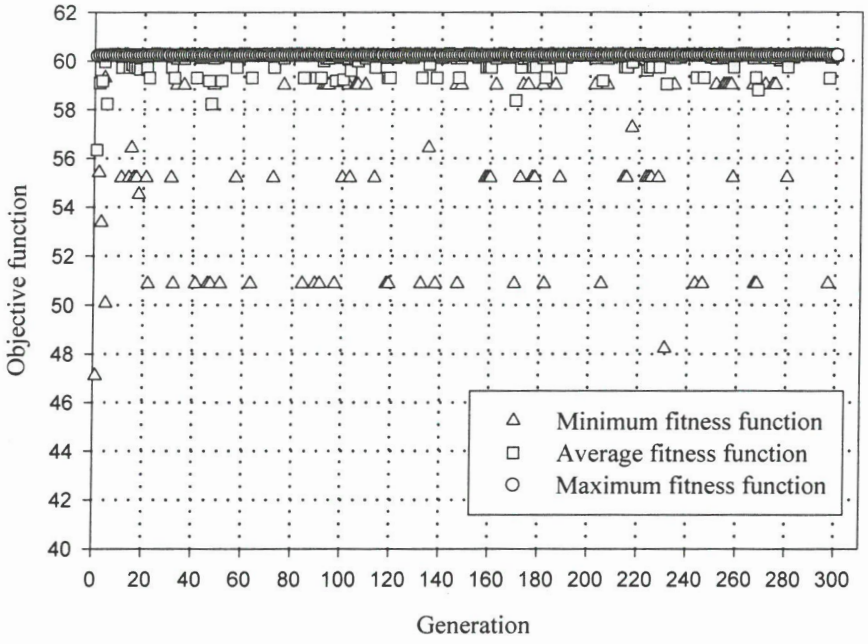

Figure 9. The distribution of fitness function

6. REFERENCES

1. Anti-Friction Bearing Manufacturers Association. ANSI-AFBMA Standard 9-1978. New York: American National Standards Institute, 1978.

2. Bäck, T., 1993, "Optimal Mutation Rates in Genetic Search," Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann, Los Angeles, 2-8.

3. Beasly, D., Bull, D.R., and Martin, R.R., 1993, "An Overview of Genetic Algorithms: Part2, Research Topics," University Computing, 15 (4), 170-181.

4. DeJong K., 1975, "The Analysis and Behavior of Class of Genetic Adaptive Systems," Ph.D. Thesis, University of Michigan.

5. Dracopoulos, D.C., 1997, "Evolutionary Learning Algorithms for Neural Adaptive Control," Springer-Verlag, London.

6. Goldberg, D.E., 1985, "Optimal Initial Population Size for Binary Coded Genetic Algorithms," The Clearinghouse for Genetic Algorithms, University of Alabama, TCGA Rept. 85001, Tuscaloosa.

7. Goldberg, D. E., 1989, "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley, Reading.
8. Goldberg, D.E., 1998, "The Design of Innovation: Lessons from Genetic Algorithms, Lessons for the Real World," University of Illinois at Urbana-Champaign, IlliGAL Report: 98004, Urbana, IL

9. Lin, C.Y. and Hajela, P., 1992, "Genetic Algorithms in Optimization Problems with Discrete and Integer Design Variables," Engineering Optimization, 19, 309-327.

10. Louis, S.J., Zhoo, F., and Zeng, X., 1997, "Flaw Detection and Configuration with Genetic Algorithms," Evolutionary Algorithms in Engineering Applications, Springer-Verlag.

11. Mitchell, M., 1997, "An Introduction to Genetic Algorithms," The MIT Press, Massachusetts.

12. Saruhan, H., Rouch, K.E., and Roso, C.A., 2001, "Design Optimization of Fixed Pad Journal Bearing for Rotor System Using a Genetic Algorithm Approach," The 1st International Symposium on Stability Control of Rotating Machinery, ISCORMA-1, Lake Tahoe, Nevada.

13. Saruhan, H., Rouch, K.E., and Roso, C.A., 2002, "Design Optimization of Tilting-Pad Journal Bearing Using a Genetic Algorithm Approach," The 9th of International Symposium on Transport Phenomena and Dynamics of Rotating Machinery, ISROMAC-9, Honolulu, Hawaii.

14. Spears, W.M., and De Jong, K.A., 1991, "On the Virtues of Parameterized Uniform Crossover," Proceedings of the 4th International Conference on Genetic Algorithms, Morgan Kaufman, 230-236.

15. Syswerda, G., 1989, "Uniform Crossover in Genetic Algorithms," Proceedings of the 3rd International Conference on Genetic Algorithms, Morgan Kaufman, 2-9.

16. Wilson, C.E., 1997, "Computer Integrated Machine Design," Prentice Hall, New Jersey.

17. Wu, S.J. and Chow, P.T., 1995, "Genetic Algorithms for Nonlinear Mixed Discrete-Integer Optimization Problems via Meta-Genetic Parameter Optimization," Engineering Optimization, 24, 137-159.