# DESIGN OPTIMIZATION OF MECHANICAL SYSTEMS USING GENETIC ALGORITHMS

**Hamit SARUHAN, İlyas UYGUR**

*Abstract*-**This paper presents an algorithm for the design of minimum weight of speed reducer, gear train, subject to a specified set of constraints. The study is primarily aimed to expose the potential of genetic algorithms, to discuss their application capabilities, and to show the concept of these algorithms as optimization techniques and their scope of application by implementing them to the speed reducer. Results obtained for the minimum weight of speed reducer are presented to provide insight into the capabilities of these techniques. Genetic algorithms are efficient search techniques which are inspired from natural genetics selection process to explore a given search space.**

*Keywords*- **Genetic algorithms, design, optimization**

*Özet*-**Bu makalede sınır şartları verilen bir hız redüktörünün minimum ağırlığını hesaplayan bir algoritma tanıtılmaktadır. Bu çalışmanın asıl amacı genetik algoritmaların potansiyellerini ve uygulama kabiliyetlerini, bir optimum hız redüktörü tasarımında göstermektir. Bu tasarım için elde edilen sonuçlar bu tekniklerin uygunluğunu göstermektedir. Genetik algoritmalar tabii seleksiyon (seçim) teknikleri kullanarak tanımlanmış sınırlar içinde tarama yapan ve genetik fikrine dayalı uygun araştırma teknikleridir.**

*Anahtar kelimeler*- **Genetik algoritmalar, tasarım, optimizasyon**

## I. INTRODUCTION

Many numerical optimization algorithms have been developed and used for design optimization of engineering problems. Most of these optimization algorithms solve engineering problems for finding optimim design. Solving engineering problems can be complex and a time consuming process when there are large numbers of design variables and constraints. Thus, there is a need for more efficient and reliable algorithms that solve such problems. The development of faster computers has allowed development of more robust and efficient optimization methods. One of these methods is the genetic algorithms. The genetic algorithms are search procedures based on the idea of natural selection and genetics [1]. Genetic algorithms can be applied to conceptual and preliminary engineering design studies. Genetic algorithms have been increasingly recognized and applied in many applications. Interested reader can refer studies by [2], [3]. This paper shows how genetic algorithms search through a design space to find the minimum value of the objective function for engineering design problems.

## II.GENETIC ALGORITHMS

In this section of the paper, the fundamental intuition of genetic algorithms and how they process are given. Genetic algorithms maintain a population of encoded solutions, and guide the population towards the optimum solution [4]. 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 the class of zero-order optimization methods [5], [6].

The description of a simple genetic algorithm is outlined in Figure 1. An initial population is chosen randomly at 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 the 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 [4]:

-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.

H.Saruhan, İ.Uygur; Abant İzzet Baysal Üniversitesi, Teknik Eğitim FakültesiMakine Eğitimi Bölümü, 14550, Düzce,TURKEY

-Genetic algorithms search from a population of points not a single point.
-Genetic algorithms gather information from current search points and direct them to the subsequent search.
-Genetic algorithms can be used with discrete, integer, continuous, or a mix of these three design variables.

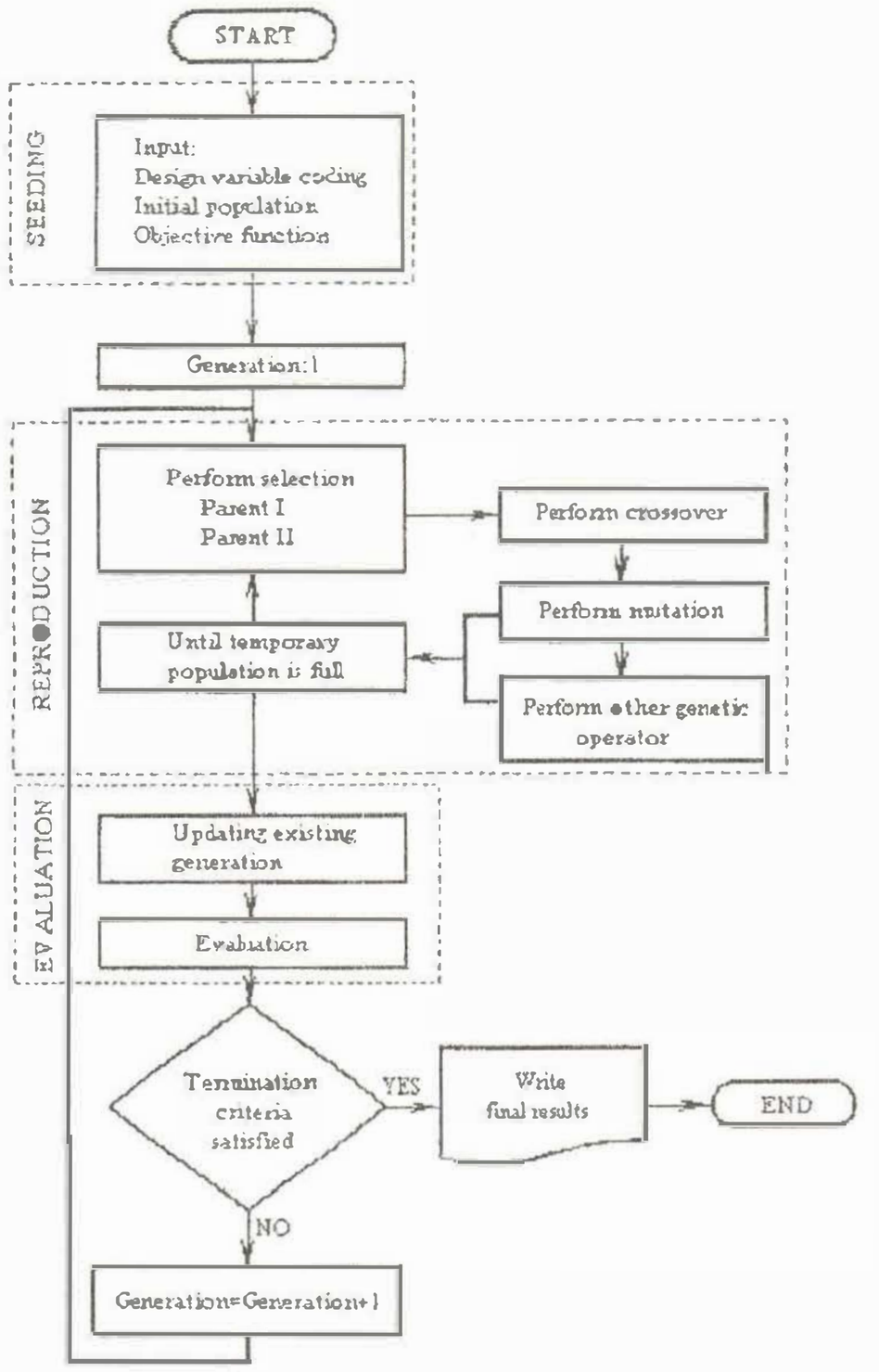



**Figure 1 Flow chart for a simple genetic algorithms.**

## III. PROBLEM STATEMENT

Figure 2 shows the configuration of a compound gear train which was taken from Rao [7]. It is desired to obtain the lowest weight of the gear train subject to a set of constraints. The statement of the design optimization of the problem is formulated as:

Objective function

$$\text{Minimize } F(X)$$

$$\text{Subject to} \qquad g_j(X) \le 0 \qquad j = 1,\cdots\cdots NIC$$

(number of inequality constraints)

$$X_i^{lower} \le X_i \le X_i^{upper}$$

where

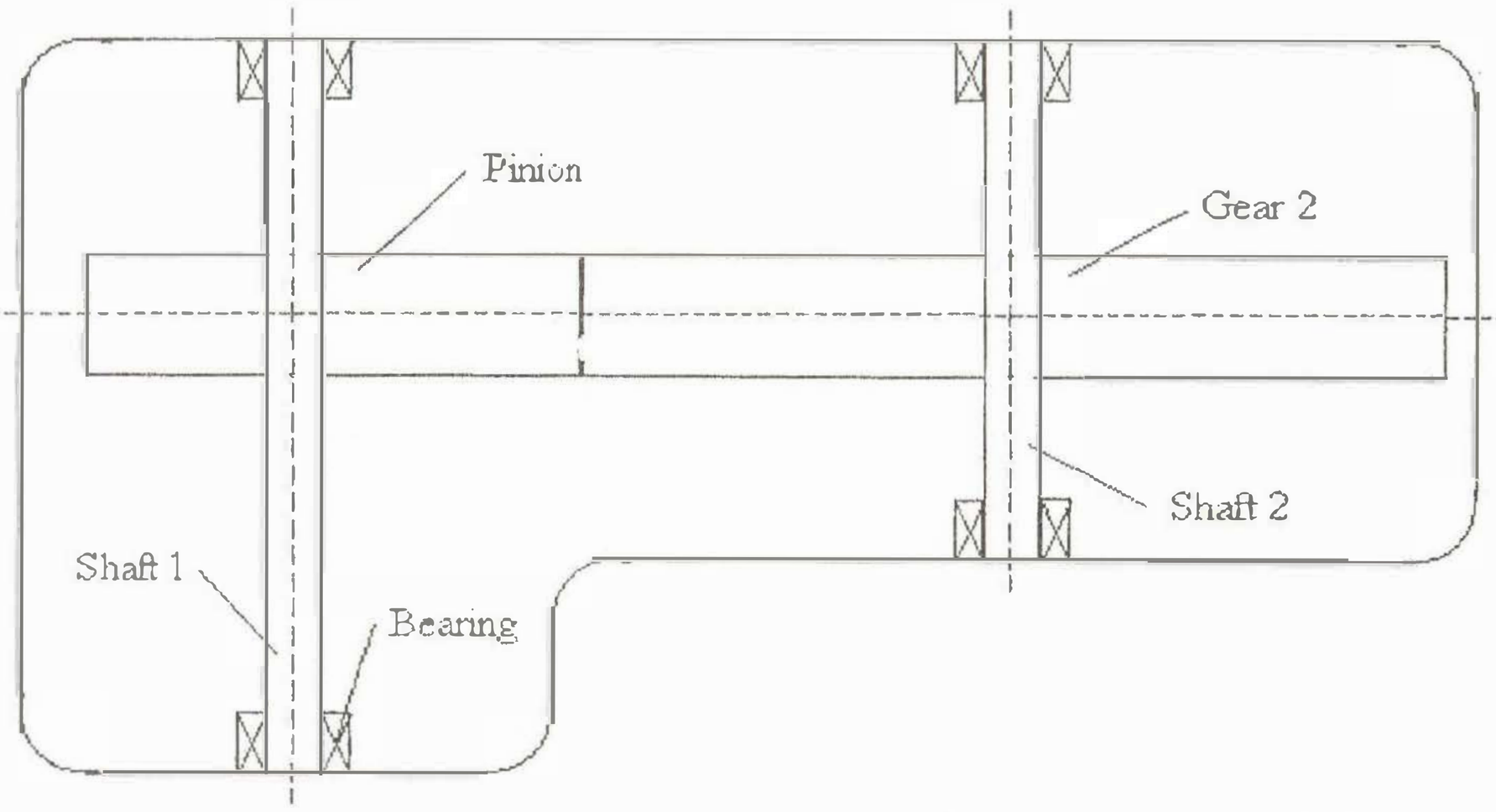
$$X_i = \{X_1, X_2, \ldots\ldots, X_n\} \qquad i = 1,\ldots\ldots n$$

$$\begin{aligned} F_{Objective} = F(X) = {} & 0.7854 X_1 X_2^2 (3.3333 X_3^2 + \\ & 14.9334 X_3 - 43.0934) - 1.508 X_1 (X_6^2 + \\ & X_7^2) + 7.477 (X_6^3 + X_7^3) + \\ & 0.7854 (X_4 X_6^2 + X_5 X_7^2) \end{aligned} \qquad (1)$$

**Figure 2 Schematic Diagram of Speed Reducer.**

### III.1. Design Variables

The design variables used for finding the minimum weight of the gear train include:



$X_1$ is the face width.

$$2.6 \le X_1 \le 3.6 \quad (2)$$

$X_2$ is module of teeth.

$$0.7 \le X_2 \le 0.8 \quad (3)$$

$X_3$ is number of teeth on pinion.

$$17 \le X_3 \le 28 \quad (4)$$

$X_4$ is length of shaft 1 between bearings.

$$7.3 \le X_4 \le 8.3 \quad (5)$$

$X_5$ is length of shaft 2 between bearings.

$$7.3 \le X_5 \le 8.3 \quad (6)$$

$X_6$ is daimeter of shaft 1.

$$2.9 \le X_5 \le 3.9 \quad (7)$$

$X_7$ is daimeter of shaft 2.

$$5.0 \le X_5 \le 5.5 \quad (8)$$

### III.2. Constraints

Constraints are conditions that must be met in the optimum design and include restrictions on the design variables value and optimum design of the function. These constraints define the boundaries of the feasible and infeasible design space domain. The constraints considered for the optimum design of gear train include the following:

$$g_1 = 2.7X_1^{-1}X_2^{-2}X_3^{-1} \le 1 \quad (9)$$

$$g_2 = 3.975X_1^{-1}X_2^{-2}X_3^{-2} \le 1 \quad (10)$$

$$g_3 = 1.93X_2^{-1}X_3^{-1}X_4^{-3}X_6^{-4} \le 1 \quad (11)$$

$$g_4 = 1.93X_2^{-1}X_3^{-1}X_5^{-3}X_7^{-4} \le 1 \quad (12)$$

$$g_5 = \left[\left(\frac{745X_4}{X_2X_3}\right)^2 + (16.9)10^6\right]^{0.5} \Big/ 0.1X_6^3 \le 1 \quad (13)$$

$$g_6 = \left[\left(\frac{745X_5}{X_2X_3}\right)^2 + (157.5)10^6\right]^{0.5} \Big/ 0.1X_7^3 \le 1 \quad (14)$$

$$g_7 = X_2X_3 \le 40 \quad (15)$$

$$g_8 = (1.5X_6 + 1.9)X_4^{-1} \le 1 \quad (16)$$

$$g_9 = (1.1X_7 + 1.9)X_5^{-1} \le 1 \quad (17)$$

Genetic algorithms are unconstrained optimization procedure. Therefore, the constrained optimization problem has been transformed into an unconstrained optimization problem by penalizing the objective function value with the quadratic penalty function. In case of any violation of a constraint boundary, the fitness of corresponding solution is penalized, and thus kept within feasible regions of the design space by increasing the value of the objective function when constraint violations are encountered. The penalty coefficients, $r_j$, for the $j$-th constraint have to be judiciously selected. The fitness function provides a measure of the performance of an individual, which is used to bais the selection process in favor of the most fit members of the current population.

$$Fitness_{Objective} = F - (F(x) + P) \quad (18)$$

$$P = \sum_{j=1}^{NCON} r_j \left( \max [0, g_j] \right)^2 \quad (19)$$

where $F$ is an arbitrary large enough that is greater than $F(X) + P$ to exclude negative fitness function values and $P$ is the penalty function.

### III.3 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 the optimization problem to be coded. Binary coding, as a finite length strings, is generally used although other coding schemes have been used. 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 $\varepsilon$. Genetic algorithms have ability to deal with integer and discrete design variables. The number of the digits in the binary strings, $l$, is estimated from the following relationship [8].

$$2^l \ge \frac{x(i)_{upper} - x(i)_{lower}}{\varepsilon} + 1 \quad (20)$$

where $x(i)_{lower}$ and $x(i)_{upper}$ are the lower and upper bound for design variables respectively. The design variables are coded into the binary digit $\{0,1\}$. The physical value of the design variables, $x(i)$, can be computed from the following relationship [9]:

$$x(i) = x(i)_{lower} + \frac{x(i)_{upper} - x(i)_{lower}}{2^{l} - 1} d(i) \qquad (21)$$

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

Design variables are represented in different level of precision. Table 1 gives descriptions of these mapping.

**Table 1 Design variables mapping.**

| Design Variables | Lower Limit | Upper Limit | Precision | String Length |
|---|---|---|---|---|
| The face width | 2.6 | 3.6 | 0.01 | 7 |
| Module of teeth | 0.7 | 0.8 | 0.1 | 2 |
| Number of teeth on pinion | 17 | 28 | 1 | 4 |
| Length of shaft 1 between bearings | 7.3 | 8.3 | 0.01 | 7 |
| Length of shaft 2 between bearings | 7.3 | 8.3 | 0.01 | 7 |
| Daimeter of shaft 1 | 2.9 | 3.9 | 0.01 | 7 |
| Daimeter of shaft 2 | 5.0 | 5.5 | 0.01 | 6 |

To start the algorithm, an initial population set is randomly assigned. This set of initialized population is a potential solution to the problem. For example, the binary string representation for the design variables, $x_i$, in Table 2 gives an example of a chromosome that represents design variables accordingly. This design string is composed of 40 ones and zeros.

**Table 2 The binary string representation of the variables.**

| Design Variables | | | | | | |
|---|---|---|---|---|---|---|
| $x(1)$ | $x(2)$ | $x(3)$ | $x(4)$ | $x(5)$ | $x(6)$ | $x(7)$ |
| 0001010 | 00 | 0000 | 000011● | 0001100 | 1000000 | 100100 |
| Concatenated Variables Head-to-Tail | | | | | | |
| 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 | | | | | | |

In Table 2, the string of 40-bit string length represents one of $2^{40}$ alternative individual solutions existing in the design space. For running genetic algorithms, an initial population is need to be assigned randomly at the beginning. Population size influences the number of search points in each generation. A guideline for an appropriate population size is suggested by Goldberg [10]. The guideline for optimal population size depends on the individual chromosome length, which is valid up to 60 expressed as follows:

$$population\ size = 1.65 * 2^{0.21 * l} \qquad (23)$$

For a string length of 40 bits, an optimal population size of 558 may be used [10]. Considering computation time, a randomly selected set, 10 strings, of potential solution is used in this study since there was seen to not have a significant improvement in results. See Table 3. The genetic algorithm then proceeds by generating new solutions with bit operations utilizing genetic algorithm operators such as selection, crossover, and mutation.

**Table 3 A set of starting population.**

| Individual number | Randomized binary string |
|---|---|
| 1 | 10010100001000100010100110010000001010 |
| 2 | 00010101000000000110100100010000000100000 |
| 3 | 11000100100100010110000110001010001111101 |
| 4 | 00011100000000100011000111011001001111100 |
| 5 | 01000100000100001110100010010010010100101 |
| 6 | 10010110011001000110000111000100101001010 |
| 7 | 00010100100010000100010001110010000010000 |
| 8 | 00010010010000000110001110111010001001000 |
| 9 | 10100101000010000110000110010000101001010 |
| 10 | 00010100000000100010000110010010000000000 |

### III.4. 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 the 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 the parent creates two Child as described in the method of uniform crossover shown in Table 4. A uniform crossover operator is used in this study. A uniform crossover operator probability of 0.5 is recommended in many works such as [11] and [12]. Crossover is very important in the success of genetic algorithms. This operator is the primary source of the 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 the 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 [13]. 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 Table 4. For each pair of parent strings a new crossover mask is randomly generated. Preventing the genetic algorithm from the 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 Table 5.

**Table 4 Uniform crossover.**

| **Crossover mask** | **00010100000000001100001100100000100100** |
|---|---|
| Parent I | 11010100010001000010000010010001001100000 |
| Parent II | 01010000001010001101001100000100010010101 |
| Child I | 01010100001010000010100010010010001000001 |
| Child II | 11010000010001000110000110000000100110100 |

**Table 5 Mutation operator.**

| | |
|---|---|
| Before | 10010100001000000110000010010001001001000 |
| After | 10010100001000100110000010010001001001000 |

The mutation rate suggested by Bäck (14) is:

$$1/population\ size < P_{mutation} < 1/chromosome\ length \quad (22)$$

A specialized mechanism, *elitism*, is added to the genetic algorithm. Elitism forces the genetic algorithm to retain the best individual in a given generation and proceed unchanged into the following generation [15]. The parameters of genetic algorithm for this study have chosen as in Table 6.

**Table 6 Genetic search algorithm parameters.**

| Genetic algorithm parameters | |
|---|---|
| Chromosome length | 40 |
| Population size | 10 |
| Number of generation | 200 |
| Crossover probability | 0.5 |
| Mutation probability | 0.01 |

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 which is used in this study 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 [16]. Thus, most or all strings in the population are identical or similar when population is converged.

## IV. RESULTS

Figure 3 shows the plots of the normalized minimum, average, and best fitness function values in each generation as optimization proceeds. As can be seen from Figure 3, the normalized fitness function of individuals in a population improves over generations. 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 has converged to a stable solutions with similar values. The results and their comparison with numerical method used by Rao [7] are shown in Table 7. As can be seen from these results, the genetic algorithms produced much better results than that the numerical method.

**Table 7 The problem design variables and objective function results.**

| **Speed Reducer** | Optimization Method | |
|---|---|---|
| **Design variables** | Numerical Optimization | Genetic Algorithm |
| The face width | 3.5 | 2.6 |
| Module of teeth | 0.7 | 0.7 |
| Number of teeth on pinion | 17.0 | 17.0 |
| Length of shaft 1 between bearings | 7.3 | 7.3 |
| Length of shaft 2 between bearings | 7.3 | 7.3 |
| Diameter of shaft 1 | 3.35 | 3.40 |
| Diameter of shaft 2 | 5.29 | 5.28 |
| **Design Objective** | | |
| **Minimum weight of gear train** | **2985.22** | **2654.19** |

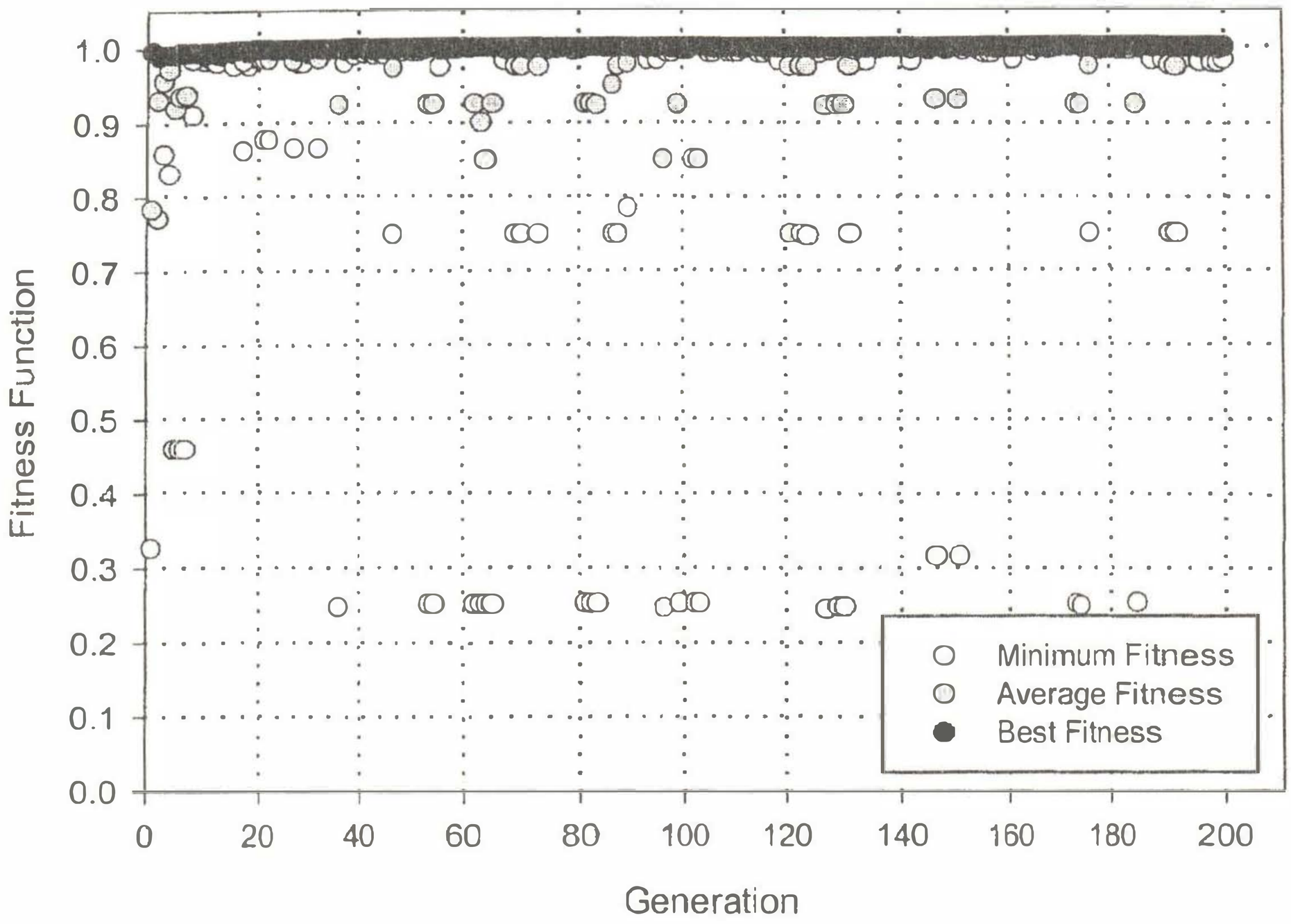



**Figure 3 Convergence process of genetic algorithms for normalized minimum, average, and best fitness function.**

## V. CONCLUSIONS

A genetic algorithm technique was used to generate the minimum weight of the gear train. The design variables were selected as those that influence the optimum design. The results show that genetic algorithm provides good solutions when compared to a numerical optimization method. 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 the engineering problems.

## VI. REFERENCES

1. Goldberg, D.E., The Design of Innovation: Lessons from Genetic Algorithms. Lessons for the Real World, University of Illinois at Urbana-Champaign, IlliGAL Report: 98004, Urbana, IL 1998.
2. Saruhan, H., Rouch, K.E., and Roso, C.A., 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, 2001.
3. Saruhan, H., Rouch, K.E., and Roso, C.A., 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, 2002.
4. Goldberg, D. E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, 1989.
5. Dracopoulos, D.C., Evolutionary Learning Algorithms for Neural Adaptive Control Springer-Verlag, London, 1997.
6. Louis, S.J., Zhoo, F., and Zeng, X., Flaw Detection and Configuration with Genetic Algorithms, Evolutionary Algorithms in Engineering Applications, Springer-Verlag, 1997.
7. Rao, S.S., Engineering Optimization Theory and Practice, New Age International (P) Limited, Pub., New Delhi, 1999.
8. Lin, C.Y. and Hajela, P., Genetic Algorithms in Optimization Problems with Discrete and Integer Design Variables, Engineering Optimization, 19, 309-327, 1992.
9. Wu, S.J. and Chow, P.T., Genetic Algorithms for Nonlinear Mixed Discrete-Integer Optimization Problems via Meta-Genetic Parameter

Optimization, Engineering Optimization, 24, 137-159, 1995.

10. Goldberg, D.E., Optimal Initial Population Size for Binary Coded Genetic Algorithms, The Clearinghouse for Genetic Algorithms, University of Alabama, TCGA Rept. 85001, Tuscaloosa, 1985.
11. Syswerda, G., Uniform Crossover in Genetic Algorithms, Proceedings of the 3rd International Conference on Genetic Algorithms, Morgan Kaufman, 2-9, 1989.
12. Spears, W.M., and De Jong, K.A., On the Virtues of Parameterized Uniform Crossover, Proceedings of the 4th International Conference on Genetic Algorithms, Morgan Kaufman, 230-236, 1991.
13. Beasly, D., Bull, D.R., and Martin, R.R., An Overview of Genetic Algorithms: Part2, Research Topics, University Computing, 15 (4), 170-181, 1993.
14. Bäck, T., Optimal Mutation Rates in Genetic Search, Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann, Los Angeles, 2-8, 1993.
15. Mitchell, M., An Introduction to Genetic Algorithms, The MIT Press, Massachusetts, 1997.
16. DeJong K., The Analysis and Behavior of Class of Genetic Adaptive Systems, Ph.D. Thesis, University of Michigan, 1975.