Turk J Engin Environ Sci
25 (2001) , 561 – 568.

A Comparison of the Performances between a Genetic Algorithm and the Taguchi Method over Artificial Problems

Özgür YENİAY
Hacettepe University, Faculty of Science,
Department of Statistics, 06532, Beytepe, Ankara - TURKEY

Received 03.06.1999

Abstract

In the manufacturing industry, to produce the best quality product, it is important to define several levels of inputs. The Taguchi method is proposed for the solution of this problem and is widely used.

In this study, a steady-state genetic algorithm (GA), called Genmak, is developed for the solution of the experimental design problems. In order to compare the performance of the suggested algorithm with that of the Taguchi method, 3 sets with different characteristics are carefully designed. Each set has 1000 test problems. Each of these test problems is an experimental design problem having 4 factors with 3 levels. Significant effects and optimum solution are determinated by statistical methods for every problem generated. Two methods are applied to the problems and the number of problems in which the optimum solution is reached is recorded. Then, the methods are compared with respect to these records. The results show that the performance of the GA is as high as that of the Taguchi method. Another important result is that the performance of the methods decreases as the amount of interaction in a problem increases.

Overall, it is concluded that GAs are suitable for finding the optimum solution to this kind of problem and can be used as an alternative to the Taguchi method.

Key Words: Genetic algorithm, Experimental design, Taguchi method

Yapay Problemler Üzerinde Genetik Algoritma ile Taguchi Yönteminin Performanslarının Karşılaştırılması

Özet

Üretim sektöründe üretilen ürünün kalitesini en iyi yapabilmek için, etmenlerin çeşitli düzeylerinin belirlenmesi önemli bir problemdir. Taguchi yöntemi, bu problemin çözümü için önerilen yöntemlerden bir tanesidir ve yaygın olarak kullanılmaktadır.

Bu çalışmada, deney tasarım problemlerinin çözümü için Genmak olarak adlandırılan bir durağan durum genetik algoritma (GA) geliştirilmiştir. Önerilen algoritmanın performansını Taguchi yönteminin performansı ile karşılaştırmak amacıyla farklı özelliğe sahip 3 küme tasarlanmıştır. Her kümede 1000 test problemi vardır. Bu test problemlerinin herbiri, 3 düzeyli 4 faktöre sahip deney tasarım problemidir. Yaratılan her problem için önemli etkiler ve en iyi çözüm, istatistiksel yöntemler yardımıyla belirlenmiştir. İki yöntem problemlere uygulanmış ve en iyi çözüme ulaşılan problemlerin sayısı kaydedilmiştir. Daha sonra, yöntemler bu kayıt değerlerine göre karşılaştırılmıştır. Sonuçlar GA'nın performansının Taguchi yönteminin performansı kadar yüksek olduğunu göstermiştir. Diğer önemli bir sonuç ise, problemdeki etkileşim miktarı arttıkça yöntemlerin performansının düşmesidir.

Genel olarak, GA'nın bu tür problemlere en iyi çözüm bulmada uygun olduğu ve Taguchi yöntemine alternatif olarak kullanılabileceği sonucuna ulaşılmıştır.

Anahtar Sözcükler: Genetik algoritma, Deney tasarımı, Taguchi yöntemi

Introduction

Within the last decade, it has become apparent that Genetic Algorithms (GAs) are potent tools for solving a wide variety of difficult, real world problems. There have been a number of international conferences on the theory and applications of GAs. They have been used successfully in a wide variety of applications including packing (Falkenauer and Delchambre, 1992), scheduling (Fang et al., 1993), neural networks (Bornholdt and Graudenz, 1992), traveling salesman (Grefenstette, 1987), steiner tree (Kapsalis et al., 1993), and transport problems (Michalewicz, 1994). Despite this variety of applications, there have only been a few studies in the field of experimental design. Reeves and Wright (1995) demonstrated the relationship between GA and statistical methods.

Genetic Algorithms

A GA may be described as a mechanism that mimics the genetic evolution of a species (Goldberg, 1989; Holland, 1992). The basic principles of GAs were first proposed by John Holland at the University of Michigan in the late 1960s and early 1970s. Thereafter a series of papers and reports have become available.

In nature, competition among individuals for scant resources such as food and space and for mates results in the fittest individuals dominating over weaker ones. Only the fittest individuals survive and reproduce, a natural phenomenon called "survival of the fittest". Their characteristics, encoded in their genes, are transmitted to their offspring and tend to propagate into new generations (Patnaik and Srinivas, 1994).

GAs simulate, in a rather simplified way, the processes outlined above to get better solutions to a problem. They work with a population of individuals, each representing a possible solution to a given problem. Each possible solution must be encoded in a binary or non-binary string format such as Gray code, floating point representation or sequence representation (for details, see Reeves, 1993; Janikow and Michalewicz, 1991). These strings are analogous to chromosomes in nature. A chromosome is composed of genes, each of which can take on a number of values called alleles.

The simplest forms of GAs work according to the scheme shown in Figure 1.

1. initialise population (t)
2. determine fitness of population (t)
3. repeat until a stopping criterion is satisfied
   a) select parents from population (t)
   b) perform crossover on parents creating population (t+1)
   c) perform mutation on population (t+1)
   d) determine fitness of population (t+1)

Figure 1. Steps of a typical Genetic Algorithm

The initial population of individuals is generated at random or heuristically. Then, each individual is evaluated according to some fitness function. In other words, the fitness function is used to map the individual into a positive number which is called the individual's fitness. In step *a*, GA selects some individuals from the population. Individuals are selected with a probability which depends on their fitness. Population selection is based on the principle of survival of the fittest. GAs traditionally use two genetic operators (crossover and mutation) for generating new individuals (in steps *b* and *c*). Steps *a, b, c* and *d* are repeated until some specified stopping criterion is satisfied. This criterion can be set by the number of iterations (generations), the amount of variation of individuals between different generations or a predefined value of fitness. The three primary genetic operators focused on by most researchers are described below.

Selection: Selection represents a very important aspect of GAs. This is usually implemented as a weighted selection, which means that individuals with higher fitness have a better chance of being chosen. It is possible for an individual to be selected more than once, or not at all. Several selection methods may be used to determine the fitness of an individual. Proportional selection and ranking are the main selection methods used in GAs. In the most common type of selection, called fitness proportionate selection, individual i with fitness f_i is given a selection probability

$$P_i = \frac{f_i}{\sum_{j=1}^{N} f_i} \tag{1}$$

where f_i is the fitness of individual i, $\sum_{j=1}^{N} f_j$ is the total fitness of the population and N (population size) is the number of individuals in the population.

Crossover: Crossover is the most important mechanism of the algorithm. It makes two new indi-

viduals (offspring) by combining two old ones (parents). Several types of crossover operators have been proposed, such as one-point crossover, two-point crossover, multi-point crossover and uniform crossover. The simplest form of crossover (one-point crossover) proceeds as follows. First, the entire population is paired off at random to give N/2 sets of individuals. A random choice is made, where the likelihood of crossover being applied is typically between 0.6 and 1.0. If it is approved, one crossover position k is generated from the uniform distribution on [1, . . . ,n-1], n: number of bits of an individual, and the last n-k bits of each individual are exchanged. If it is not approved, offspring are produced simply by duplicating the parents.

Assume that the following two parents of length n=10 have been selected for one-point crossover:
P1: 0 0 1 0 0 1 1 0 1 0 and P2: 1 1 1 0 0 1 0 0 0 1.

9 (n-1) possible crossover positions exist. Suppose that position 6 has been selected. Then, one offspring produced by the concatenation of 001001 from P1 and 0001 from P2 is 0010010001. The second offspring produced by concatenation of 111001 from P2 and 1010 from P1 is 1110011010. Figure 2 illustrates this process (/ is the crossover position).

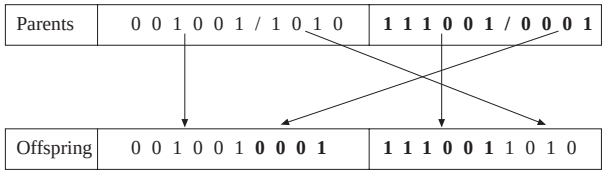

Figure 2. Example of one-point crossover

Mutation: Mutation is applied to offspring after crossover. The GA has a mutation probability, p_m, which dictates the frequency at which mutation occurs. For each bit in each offspring, the GA checks to see if it should perform a mutation. If it should, it changes the bit value to a new one. Mutation in a binary coded string is the changing of 0 to 1 or 1 to 0. For example, the GA decides to mutate bit position 4 in the offspring 0010010001 (see Figure 3).

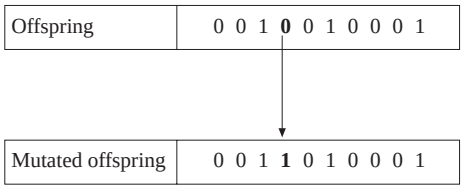

Figure 3. Bit mutation on the fourth bit

The resulting individual is 0011010001 as the fourth bit in the offspring is flipped.

The mutation probability should be kept very low (usually about 0.001) as a high mutation probability will destroy fit offspring and degenerate the GA into random walk.

Experimental Design Methods

Experimental design is a useful analytical method where a) a mathematical model of the system is not available, b) the system is not well understood or c) the system is described by a complex mathematical model (Rowlands, 1996). Experimental design methods first gained acceptance as a result of the work of Sir R.A. Fisher in agriculture. Today, they are used in application fields such as agriculture, medicine and chemistry and new methods have been developed. In addition to the experimental design methods which have been used in every field, there are also special experimental design methods developed for these fields.

Experiments where the effects of more than one factor on response are investigated are known as full factorial experiments. In a full factorial experiment, both the levels of every factor are compared with each other and the effects on the response of levels with each factor are investigated according to the levels of other factors. For example, in a 2^5 full factorial design problem, with 5 factors each at two levels, 32 experiments must be done in order to find the best combination of factors. When there are 7 factors and 10 factors, the number of total experiments become 128 and 1024 respectively. Thus, it is clear that as the number of factors and levels increases, the number of experiments geometrically increases. In addition, the consideration about the experimental cost and time problem make the determination of the best treatment combination harder. To reduce the number of experiments to a practical level, only a small set from all the possibilities is selected. The method of selecting a limited number of experiments which produces the most information is known as a fractional factorial experiment (Montgomery, 1997).

Taguchi (1987) first developed a method which simplifies the use of a fractional factorial experiment. This method uses a special set of arrays called orthogonal arrays. These standard arrays stipulate the way of full information of all the factors that affect the performance parameter. The crux of the orthogonal arrays method lies in choosing the level combinations of the input design variables for each

experiment. For example, the L4 orthogonal array consisting of 4 rows and 3 columns where each row corresponds to a particular experiment (treatment combination) and each column identifies settings of a design parameter is as follows:

Experiment number	Column 1	2	3
1	1	1	1
2	1	2	2
3	2	1	2
4	2	2	1

Figure 4. L4 orthogonal array

In the first run, for example, the three design variables are set at their low level (level=1). In the second run, the first parameter is set at level 1 and the remaining two variables are set to high level (level 2), and so on.

Generating the Test Problems and Their Specialties

In order to compare the performances of two methods, 3 problem sets which consist of 1000 problems were generated. Each problem in these sets has 4 factors each at 3 levels, i.e., it has a 3^4 factorial design. Therefore, the number of possible treatment combinations is 81.

While the problem sets were being generated, each of them was thought to have different properties. The problems were designed such that in the first set only the main effects are important, in the second set the main effects and interactions are important, and in the third set the main effects and interactions are important (interaction effects, however, are smaller than those in the second set). The purpose of changing the level of interactions in the second and third sets is to see the effects of the interactions on the methods.

By generating a problem in a set, we express obtaining $\mathbf{Y}_{81\times1}$ response vector of that set with the property described above. Before generating 1000 $\mathbf{Y}_{81\times1}$ vectors in the ith set, only one $\mathbf{Y}_{81\times1}$ vector for that set was produced. This was achieved in two stages. In the first stage, with the help of the designed properties of the set, for the β unknown parameters vector in the model given by (2), arbitrary coefficients are attained. For instance, for the set we want only the main effects to be important; the coefficients of the main effects in the model are attained larger than the coefficients of the other effects. In the second stage, by using this arbitrary coefficients vector and $\mathbf{X}_{81\times27}$ matrix in the model, $\mathbf{Y}_{81\times1}$ was calculated. Whether the calculated $\mathbf{Y}_{81\times1}$ vector possesses the property of the set it represents or not was tested by analysis of variance (ANOVA). If it was suitable for that set, a basic $Y_{81\times1}$ vector was obtained. If not, we returned to the first stage and the same operations were repeated by attaining a new arbitrary vector of unknown parameters.

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{2}$$

In matrix form, the model can be written as follows:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_{79} \\ y_{80} \\ y_{81} \end{bmatrix} = \begin{array}{c} \begin{matrix} 1 & X_1 & X_2 & X_1 & X_2 & & & X_4^2 X_3 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & . & . & . & 1 \\ 1 & 1 & 1 & 1 & . & . & . & 2 \\ 1 & 1 & 1 & 1 & . & . & . & 3 \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ 1 & 3 & 3 & 2 & . & . & . & 2 \\ 1 & 3 & 3 & 2 & . & . & . & 3 \\ 1 & 3 & 3 & 2 & . & . & . & 1 \end{bmatrix} \end{array} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ . \\ . \\ . \\ \beta_{24} \\ \beta_{25} \\ \beta_{26} \end{bmatrix}$$

$$+ \begin{bmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \varepsilon_2 \\ . \\ . \\ . \\ \varepsilon_{78} \\ \varepsilon_{79} \\ \varepsilon_{80} \end{bmatrix}$$

where

$\mathbf{Y}_{81\times1}$: the vector of response values,

$\boldsymbol{\beta}_{27\times1}$: the vector of unknown parameters,

$\mathbf{X}_{81\times27}$: a sub-orthogonal array produced by selecting the columns corresponding to the main effects, 2-factor linear interaction effects and 2-factor quadratic interaction effects and excluding the columns corresponding to the other interaction effects from the 81×40 dimensional L81 orthogonal array,

$\boldsymbol{\varepsilon}_{81\times1}$: the vector of random errors.

The reason for producing a new $\mathbf{X}_{81\times27}$ orthogonal array by excluding the columns of L81 orthogonal array corresponding to 3-factor and 4-factor interactions is that the interpretation of these in the model is hard and they are not used in practice. These interaction effects will be in the error term in the model.

The least squares estimates of $\boldsymbol{\beta}$ were found by using the $\mathbf{Y}_{81\times1}$ response vector and the $\mathbf{X}_{81\times27}$ matrix in the model given by (2). For each element of

$\boldsymbol{\beta}$, the variances were computed. With the aid of these variances, the 95% confidence intervals of each element of $\boldsymbol{\beta}$ were found.

In order to generate 1000 problems with the $\mathbf{Y}_{81\times1}$ vector found for the ith set, 1000 $\boldsymbol{\varepsilon}_{81\times1}$ random error vectors from normal distribution with mean 0 and variance 1 and 1000 $\boldsymbol{\beta}$ vectors from the confidence interval of the estimated $\boldsymbol{\beta}$ vector were produced by Minitab 9.2 for Windows. One thousand different $\mathbf{Y}_{81\times1}$ vectors were obtained by using the $\boldsymbol{\beta}_{27\times1}$ and $\boldsymbol{\varepsilon}_{81\times1}$ vectors produced in the model given by (2). For instance, the first element of the $\mathbf{Y}_{81\times1}$ vector (y_1: the response value obtained for the first treatment combination) has the contribution of the following effects:

$$\begin{aligned} y_1 = X\boldsymbol{\beta} + \varepsilon = \quad & \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \beta_4 X_1^2 + \beta_5 X_2^2 + \beta_6 X_3 + \beta_7 X_4 + \beta_8 X_3^2 + \beta_9 X_4^2 + \\ & \beta_{10} X_1 X_3 + \beta_{11} X_1 X_4 + \beta_{12} X_2 X_3 + \beta_{13} X_2 X_4 + \beta_{14} X_3 X_4 + \beta_{15} X_1^2 X_2 + \beta_{16} X_1^2 X_3 + \\ & \beta_{17} X_1^2 X_4 + \beta_{18} X_2^2 X_1 + \beta_{19} X_2^2 X_3 + \beta_{20} X_2^2 X_4 + \beta_{21} X_3^2 X_1 + \beta_{22} X_3^2 X_2 + \beta_{23} X_3^2 X_4 + \\ & \beta_{24} X_4^2 X_1 + \beta_{25} X_4^2 X_2 + \beta_{26} X_4^2 X_3 + \varepsilon \end{aligned} \tag{3}$$

where

y_1: the first element of $\mathbf{Y}_{81\times1}$ vector, i.e., the fitness value of the first treatment combination,

$\boldsymbol{\beta}_{27\times1}$: a produced $\boldsymbol{\beta}$ vector,

$\mathbf{X}_{1\times27}$: a vector composed of the elements of the first row of the $X_{81\times27}$ matrix,

$\boldsymbol{\varepsilon}$: the first element of the $\boldsymbol{\varepsilon}_{81\times1}$ vector.

Neither the GA nor the Taguchi method guarantees an optimum solution to the problem considered. It is not appropriate to compare the two methods according to the results obtained from only one test problem. Therefore, problems which have the same characteristic were included in the same set and 1000 problems were generated for each set.

Although a GA can find good solutions for difficult problems, it does not give additional information about the problem like experimental design methods do. Therefore, each method must be applied to each problem, and the number of problems in which the optimum is reached must be recorded. This is the best way to follow if the results obtained using the two methods are to be compared.

Therefore, firstly the optimum solution for each problem must be found by a statistical method. By using the experimental design submodule of Statistica 5.0 for Windows, the best treatment combinations (optimum solutions) for the problems in 3 sets were found. The results are shown in Table 1.

The Newman-Keuls range test (for details of this test, see Montgomery, 1997) was used to determine the treatment combinations which were not different from the best one in each set. Table 2 shows treatment combinations that had no difference from the best treatment combinations.

Table 1. The best treatment combination in each set

Problem set	Best treatment combination
I	1 1 1 1
II	2 2 2 1
III	2 2 2 2

Table 2. The best and other combinations having no difference from the best treatment combinations in each set

Problem set	The best and other combinations that are not different from the best one	
I	1 1 1 1	2 1 1 1
II	2 2 2 1	2 2 2 3
III	2 2 2 2	

As seen from Table 2, there are second solutions for both set I and set II. It has been observed that the differences between these solutions and their corresponding optimum solutions are insignificant. As a result, it was decided to make two controls according to whether the result obtained when a problem in the i^{th} set was solved by GA and the Taguchi method was one of the solutions in Table 1 and Table 2. In this way the number of cases in which the methods reach the optimal solution, the second best solution and a different solution (unsuccessful cases) can be found and the interpretation will be detailed. The stages of the first control can be summarised as follows:

Step 1: Take the k^{th} problem from the i^{th} set.

Step 2: Solve the problem by GA and the Taguchi method.

Step 3: Check whether the result is the same as the one given in Table 2.

Step 4: Add one to the success frequency of the method with which the same result is obtained.

Step 5: Go to Step 1 for the solution of a new problem. If k=1000 go to the next set and repeat the first four steps. If i=3 and k=1000 end.

A Steady-State Genetic Algorithm: Genmak

The simple GA creates an entirely new population from an existing population in each iteration. GAs that replace the entire population are called generational GAs. GAs that replace only a small fraction of strings in each iteration are called steady-state GAs.

Genmak is the name we gave to the GA we developed as an alternative to the Taguchi method by taking experimental design conditions into account and was programmed with Visual Basic 3.0 for Windows. Only two new strings are obtained in each iteration of Genmak and these are replaced by two strings of the existing population. Thus, Genmak is a steady-state GA (see selection step of Genmak).

Genmak can conveniently be used for experimental design problems with different numbers of factors and levels other than the 3^4 experimental design problems explored in this study. Parameter values are not constant. They can be changed according to the problem. Each iteration can be monitored on the screen and can be altered whenever needed (changing a string in the initial population produced randomly, for example) (Yeniay, 1999).

There are some similarities between the concepts of GA and the design of the experiment such as:

chromosome - treatment combination

gene - factor

allele - level

offspring - new candidate for treatment combination

generation - iteration

population - a subset of all probable treatment combinations

While Genmak was developed, these similarities were used.

The Specialties of Genmak:

Initial population: By using Genmak, the initial population can be generated both at random and by the researcher. In this study, the initial population was generated at random. The number of fitness evaluations is limited and using a small population is important in our study. Therefore, the population size is taken as 13.

Selection: Genmak uses steady-state selection. Thus treatment combinations in the population are ordered by their fitness values. Two individuals that have the highest fitness values are crossed over. By this crossing we have offspring that replace two individuals having the lowest fitness value. In this way, only two new individuals are obtained in each iteration. In other words, the total number of fitness evaluations decreases by 2 in each iteration.

Coding: Integer coding is used for Genmak. In this study, as it concerns 3^4 design, chromosomes are formed by 4 integers, and each of these integers corresponds to each gene (factor). For example, 1 1 2 3 chromosome means that the first and second factors have low levels, the third factor has medium level and the fourth factor has high level.

Crossover: In each generation of Genmak, two chromosomes that have the highest fitness values of the population are recombined by one point crossover operator. In Genmak, there is a restriction that the offspring created by crossover should be different from their parents. Unless this rule is satisfied, the crossover is repeated by changing the crossover point or one of the parents.

Mutation: In Genmak, when any level of a factor is chosen for mutation, this level value is changed by one of the possible level values of the related factor. For example, if the first gene of 1 1 2 3 chromosome is chosen for mutation, the value of 1 is replaced with 2 or 3 randomly. In our study, the mutation operator is not used.

Stopping Criteria: In order to compare GA and the Taguchi method under the same conditions, it was decided to use L27 orthogonal array corresponding to 1/3 fraction of the 3^4 factorial design. Thus, we had 27 fitness evaluations. As 2 fitness evaluations are carried out each generation, (because 2 offspring were created), 14 new offspring are created during 7 generations. Furthermore, as there are 13 individuals' fitnesses that should be evaluated in the initial population, we will have a total of 27 individuals' evaluations.

The reason why the number of function evaluations is limited to 27 is that fitness values have to be obtained by experiment in a real life application (chemistry, textiles etc.). To set up and run these experiments needs time and money in real life.

Solution By Genmak

Genmak was run with 7 iterations for each 1000 problems in 3 sets. The frequency of identification of the solutions in Tables 1 and 2 and the number of iterations required to find them were recorded. The frequencies of discovery of those solutions by Genmak and the Taguchi method are shown in Tables 3 and 4 respectively.

Solution By The Tauguchi Method

The frequency of success obtained by using L27 orthogonal array is given in Tables 3 and 4. From these tables, we see that when L27 orthogonal array was used, the best solutions were found in 100% of the 1000 problems in all sets. Since Genmak required, on average, only 17 iterations to find the best solution (see Tables 3 and 4), we wished to see the performance by using less treatment in the Taguchi method and we used 1/9 fraction of the 3^4 factorial, i.e., L9 orthogonal array.

Table 3. Frequency of identification of the best solution in 3 sets by Genmak and the Taguchi method

	Problem set		
Method	I	II	III
GA	780 (17)*	370 (17)*	750 (17)*
Taguchi (L_{27})	1000	1000	1000
Taguchi (L_9)	480	160	1000

*average number of iterations required to find the best solutions given in Table 1 by Genmak.

Table 4. Frequency of identification of the best or second best solution by Genmak and the Taguchi method

	Problem set		
Method	I	II	III
GA	840 (17)*	770 (17)*	750 (17)*
Taguchi (L_{27})	1000	1000	1000
Taguchi (L_9)	1000	160	1000

*average number of iterations required to find the solutions given in Table 2 by Genmak.

When L9 array was used instead of L27, 27-9=18 experiments decreased, but the chance of finding the best solution decreased 52% in the first set and 84% in the second set.

With Genmak, the performance of finding the optimum solution was 78% in the first, 37% in the second and 75% in the third set of problems. It should be noted that this level of success was obtained in about 17 iterations and while the interaction effect was increasing, the success of Genmak was decreasing. The result was valid for L9 trials as well.

As Genmak found the best solution for most of the problems in the first and third sets, it is seen from Table 4 that the level of success of Genmak did not change much in these sets. Genmak generally found the second best solution because of the high interaction effect in the second set. Therefore, its performance is better in Table 4. When L9 array was used, the frequency of identification of the second best solution was slightly higher than that of the optimal solution in the first set (in 1000-480=520 problems). However, frequency of the second best solution was exactly zero by using the same array in the second set.

Conclusion

In this study, we dealt with problems where fitness evaluations are expensive and time-consuming. However, for many applications of GAs, a fitness function is known and the time in which the fitness function is evaluated is not so important. Although all possible solutions and the best solution were known in the study, to prepare and carry out all of these experiments is expensive in the real world. Therefore, the number of the function evaluations was decided as 27 for the upper bound.

The results of the study are as follows:

i) Because of the reasons described above, it may not be suitable to use a large population size in solving the experimental design problems by GA, because obtaining and processing the experimental data requires time and money. In other words, there is a linear relation between the number of experiments and cost in this type of application.

ii) Experimental design methods give more information about the process (for example, the regions of good and bad results revealed and interaction between the factors).

iii) The purposes should be specified before deciding which method can be used. If finding the best solution is adequate, GAs may be suitable. However if the purpose is having information about the process, the design of experiment methods are better, because they give more information.

iv) In the large dimensional problems, GAs need less human effort than the experimental design methods.

Acknowledgments

Parts of this paper are based on my Ph.D. thesis at Hacettepe University. I would like to thank Prof. Dr. Zehra Muluk and Prof. Dr. Gülsüm Hocaoğlu for their help in carrying out the experiments and for their valuable suggestions. I also thank the anonymous reviewers and the editor for their valuable suggestions on improving this paper.

References

Bornholdt, S. and Graudenz, D., General Asymmetric Neural Networks and Structure Design by "Genetic Algorithms", Neural Networks, 5, 327-334, 1992.

Fang, H.L., Ross, P. and Corne, D., "A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Re-Scheduling and Open-shop Scheduling Problems", Proc. of 5^{th} International Conference on Genetic Algorithms, S. Forrest (ed.) Morgan Kaufmann, San Mateo, CA, 375-382, 1993.

Falkenauer, E. and Delchambre, A., "A Genetic Algorithm for Bin Packing and Line Balancing", Proc. IEEE International Conference on Robotics and Automation, 1992.

Goldberg, D.E., Genetic Algorithms in Search, Optimisation and Machine Learning, Addison-Wesley, Reading, MA, 1989.

Grefenstette, J. J. 1987), Incorporating Problem-Specific Knowledge into Genetic Algorithms, Genetic Algorithms and Simulated Annealing, L. Davis ed., Los Altos, CA, 42-60.

Holland, J.H., Adaptation in Natural and Artificial Systems an Introductory Analysis with Applications to Biology, Control and Artificial Intelligence, Cambridge, Mass: MIT Press, 1992.

Janikow, C. and Michalewicz, Z., An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms, Proc. of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, Los Altos, CA, 31-36, 1991.

Kapsalis, A., Smith, G.D. and Rayward-Smith, V.J., "Solving the Graphical Steiner Tree Problem Using Genetic Algorithms", JORS, 44, 397-406, 1993.

Michalewicz, Z., Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Berlin, 1994.

Montgomery, D.C., Design and Analysis of Experiments, John Wiley & Sons, NY, 1997.

Reeves, C.R. and Wright, C.C., "Genetic Algorithms and Statistical methods: A Comparison", Proc. 1st IEE / IEEE International Conference on Genetic Algorithms for Engineering Systems: Innovations and Applications, Sheffield, UK, 1995.

Reeves, C., Modern Heuristic Methods for Combinatorial Optimisation, Blackwell Scientific Publications, Oxford, 1993.

Rowlands, H., "A Hybrid Approach for Optimum Design Using a Genetic Algorithm, A Neural Network and the Taguchi Method", Second International Conference on Adaptive Computing in Engineering Design and Control, University of Plymouth, 205-211, 1996.

Srinivas, M. and Patnaik, L.M., Genetic Algorithms: A Survey, Computer, 6, 17-26, 1994.

Taguchi, G., System of Experimental Design: Engineering Methods to Optimize Quality and Minimise Costs, UNIPUB / Kraus International Publications, Two Volumes, 1987.

Yeniay, Ö., A Genetic Algorithm Approach to the Problems of Taguchi's Experimental Design, Doctoral dissertation, Hacettepe University, 1999.