ASIAN JOURNAL OF CIVIL ENGINEERING (BUILDING AND HOUSING) VOL. 5, NOS 3-4 (2004)
PAGES 121-143

A HYBRID GENETIC ALGORITHM FOR STRUCTURAL OPTIMIZATION PROBLEMS

Mohammad G. Sahab*, Civil and Environmental Engineering Department, Amirkabir University of Technology (Tafresh Branch) Tehran, Iran.
Vassili V. Toropov, Altair Engineering Ltd. Vanguard Centre, Sir William Lyons Road, Coventry, CV4 7EZ, UK.
Ashraf F. Ashour, School of engineering, Design and Technology, University of Bradford, Bradford, West Yorkshire BD7 1DP, UK.

ABSTRACT

This paper presents a hybrid optimization algorithm based on a modified genetic algorithm (GA). The algorithm includes two stages. In the first stage, a global search is carried out over the design search space using a modified GA. In the second stage, a local search is executed that is based on GA solution using a discretized form of Hooke and Jeeves method. The modifications on basic GA includes dynamically changing the population size throughout the GA process, utilizing variable penalty multiplier and the use of a square root form of the penalty function in constraint handling. The hybrid algorithm and the modifications to the basic GA are examined on the design optimization of a well-known test problem (10 bar truss). The effect of different parameters and techniques of handling GA operators on the performance of the proposed algorithm is investigated. The hybrid algorithm is employed for the design optimization of a reinforced concrete flat slab building and the results are compared with those of using the GA only.

Keywords: structural optimization, genetic algorithm, hybrid optimization algorithm, flat slab building

INTRODUCTION

Genetic algorithms (GAs) belong to a family of algorithms called evolutionary algorithms. Other major algorithms of this family include evolutionary programming, evolutionary strategies and genetic programming. Evolutionary algorithms have been inspired from the principles of evolution in the nature. Genetic algorithms were developed by Holland, his colleagues and students in the 1960s and 1970s [1].

GA has been described as a robust optimization technique [2]. GA can not guarantee finding the optimum solution but it is able to determine a near-optimal solution. As a GA is

* E-mail address of the corresponding author: sahab@cic.aut.ac.ir

a time consuming optimization technique it can be used in conjunction with other optimization techniques to provide an efficient hybrid optimization algorithm. These hybrid algorithms based on GA are usually called hybrid GA [3]. Also many attempts were made to modify GA operators and tuning of GA parameters to increase its efficiency [4-6]. In this paper a hybrid optimization algorithm based on a GA is presented and examined on design optimization of a 10-bar truss test problem and a reinforced concrete (RC) flat slab building.

THE BASIC GA PROCEDURE

GA is an iterative procedure that is motivated by the principles of natural selection and survival of the fittest ones. A GA begins searching by a randomly created population of solutions that are represented using a string coding of fixed length (chromosome). The solutions are decoded and evaluated according to a criterion that is called fitness function. Every solution is assigned a fitness according to the obtained value of fitness function for that solution. To produce a new generation (new population) of solutions, some of the solutions are selected according to their fitness values to enter to the matting pool. Next, the matting pool is filled by cloning the individuals (solutions) in proportion to their fitness values using a randomized technique. Creation of a new population is implemented by crossover and mutation operations. In crossover stage, two individuals are selected as parents and then some segments of encoded string of parents are swapped to create two children. During the mutation, some random changes are applied to encoded string of some randomly selected children. The fitness values of individuals of new generation are evaluated. If the termination conditions are satisfied, the process is terminated. Otherwise the iteration process is repeated.

CONSTRAINT HANDLING

As a GA is an unconstrained optimization technique, it is necessary to transform the constrained optimization problem to an unconstraint one. Several methods for handling constraints by GAs have been proposed [7]. Among them rejecting strategy and the methods based on penalty approach can be named. In the rejecting strategy, any design that violates one or more constraints is not accepted to create a new population in the GA process. In a penalty method, a constrained optimization problem is converted to an unconstrained problem by adding a penalty for each constraint violation to the objective function, $C(\boldsymbol{x})$, as follows:

$$\widetilde{C}(\boldsymbol{x}) = C(\boldsymbol{x}) \; + \; r \sum_{i=1}^{m} \Phi_i(\boldsymbol{x}) \tag{1}$$

where $\widetilde{C}(\boldsymbol{x})$ is the penalised objective function, r is the penalty multiplier, m is the number of constraints and Φ_i is the i-th penalty function which can be expressed in a general form as follows:

$$\Phi_i(\boldsymbol{x}) = [\max(G_i(x), 0)]^n \tag{2}$$

where n is the power of penalty function and $G_i(\boldsymbol{x})$ is the value of the i-th constraint. Different forms for penalty function have been proposed in the literature [7]. Quadratic penalty function is one of the most common forms of the penalty function, which has been used by many researchers [2, 8-10].

Penalty multiplier is an important effective factor that influences the way of driving the bulk of population over the design search space. At the beginning of a GA process the population of designs is distributed all over the design search space either feasible or infeasible region. As the GA progresses and the design population evolves, applying a penalty to infeasible designs forces the population to move towards the feasible region of the design search space. Two strategies for applying a variable penalty multiplier are proposed and examined in this paper.

ENCODING

In the basic GA encoding is carried out using binary strings [2]. Traditional binary coding for function optimization is known to have a weakness due to the large change of a real parameter value arising from changing a single bit in the binary string of the parameter. For example, the binary strings 011111 and 111111 are equal to decimal numbers of 63 and 31, respectively, while they are only different in one bit.

Gray coding is another way of coding parameters into bits, which has the property that an increase by one step in the value of a design variable corresponds to change of a single bit in the binary string of the design variable. The conversion from Gray coding to binary coding is given by:

$$\beta_k = \sum_{i=1}^{k} \gamma_i \tag{3}$$

where β_k is the k-th binary code bit and γ_i is the i-th Gray code bit. Bits are numbered from 1 to n starting on the left and summation is done in binary mode [11]. These two different coding methods have been implemented in the developed optimization algorithm.

SELECTION

Fitness proportional selection is a common method in GAs [2]. Its basic idea is that the selection probability for each individual is proportional to the fitness value. In another popular type of selection, tournament selection, some number of individuals are randomly selected from the population. The best individual of this group is then selected as a parent. The process is repeated to select one more parent to form a couple. The number of selected individuals for tournament competition is called tournament size, which is often two [9].

Whitley [12] pointed out two weaknesses for the fitness proportional selection, which are

stagnation and premature convergence of the search. When the relative difference between fitness value of the individuals is small the search process stagnates. On the other hand, when the relative difference between the fitness value of the individuals is large, the fittest individuals dominate the creation of the next generation. Consequently, the search prematurely converges to a solution. In tournament selection all individuals have an equal chance of being selected for tournament competition, therefore this method is immune to the aforementioned shortcomings of the fitness proportional selection. These two selection schemes are compared in this study.

CROSSOVER

One-point crossover is the simplest type of crossover, but it has some shortcomings. This method combines the bit strings in a limited way. Sometimes only a few bits in the strings of parents need to be altered, but this method exchanges all bits after the crossover point. This effect is called positional bias [13,14]. This implies that the strings with long defining lengths could be often destroyed under one-point crossover [13,14]. The segments, exchanged between the parents, always include the end points of the strings. This is called endpoint effect [13]. In addition, crossover of parent strings that are identical after the crossover point has no effect, as the children will be identical to the parents.

In order to reduce the above mentioned problems of one-point crossover, two-point cross over often used [13]. In this method two crossover points are selected at random and the parts of strings between them are exchanged.

Uniform crossover exchanges every bit along the parent strings at random. Uniform crossover does not have any of the aforementioned shortcomings of the one-point crossover. However, it has been claimed that this method can be highly disruptive, especially in early generations [13,14]. These three methods of crossover are investigated.

ELITIST STRATEGY AND MUTATION

The elitist strategy introduced by DeJong [15] transfers some number of the fittest individuals of the current generation according to the assumed percentage of elite, P_e, to the next generation. Therefore, the best individuals are not lost due to crossover and mutation and the maximum fitness value in a generation can not decrease as compared with the previous generation. Since in the developed GA in this study mutation is applied on common (non-elite) individuals, the mutation rate is less than a typical value of around 0.1%. In this paper the effect of different elite percentages on the efficiency of GA is investigated.

THE PROPOSED HYBRID ALGORITHM

The hybrid algorithm includes two stages. In the first stage, a global search is carried out

over the design search space using a modified GA as explained in the next section. In the second stage, a local search is implemented based on GA solution using a discretized form of Hooke and Jeeves method [16]. At the beginning, the penalized objective function is calculated at a point obtained by a positive or negative increment in the direction of the first coordinate (design variable). If any of these new points gives a better design, this new point is considered as a new base point. Then exploration is continued in the specified direction of the first coordinate until no better design can be obtained by a further movement. This process is repeated for all coordinates until there is no point in the neighbourhood of a base point that gives a better design. If an improvement is achieved in the direction of any coordinate, the exploration is carried out by going through all other coordinates.

MODIFICATIONS TO THE BASIC GA

Two modifications have been applied to the basic GA and two other modifications are examined in the constraint handling by penalty approach.

The first modification to the basic GA is that GA starts by a large size, I_s, of randomly created designs over the design search space and then, a smaller number of best designs, S_s, are selected to carry on the rest of the GA process [5].

As the GA process progresses, the population becomes more uniform and fitness values of individuals become close to each other. One individual from each group of individuals with same fitness value, as a representative of the group, can be sufficient to transfer genetic information during the crossover. The second modification limits the number of copies of each group of designs with the same fitness to one. In this manner the population size is decreased during the process but it does not become smaller than a predefined minimum allowable population size, M_s [6].

Proximity of the GA solution to the optimum solution heavily depends on the values of the penalty multiplier. If the penalty coefficient is moderate, the algorithm may converge to an unfeasible solution. On the other hand, if the penalty coefficient is too large this case is equivalent to the rejecting strategy. Infeasible solutions may have some useful genetic information, which can be transferred to feasible solutions during the crossover. Therefore, the use of a moderate penalty multiplier can increase the chance of transformation of useful information from infeasible to feasible solutions. These facts suggest using a variable penalty multiplier in the GA process. The penalty multiplier may be increased by a positive increment or decreased by a negative increment. These two approaches of changing the penalty multiplier are examined in this paper. As Eq. (4) shows, the final penalty multiplier is limited by a certain value, r_f.

$$r_i = Max((r_{in} + I_r \times (N_{ge\,i} - 1)),\ r_f) \tag{4}$$

where r_i and $N_{ge\,i}$ specify the i-th penalty multiplier and generation number, respectively, r_{in} is the initial value of penalty multiplier and I_r is the penalty multiplier increment.

Since the constraints have been introduced in the normalized form, approaching to the

end of the GA process, the values of constraints for infeasible solutions become close to zero. In this case, the use of a power n lesser than one, in Eq. (2), for the penalty function can magnify a slight violation of the constraints, more so than in the traditional quadratic form. On the other hand, the use of a power greater than one for the cases when the value of constraint violation is greater than one may magnify the constraint violation and consequently using a smaller penalty multiplier can be possible. These cases may mostly occur at the beginning of search process when the solutions are distributed over the design search space either in feasible or in infeasible domain. In this paper, the efficiency of the optimization algorithm using quadratic, linear and square root forms for the penalty function is compared.

NUMERICAL RESULTS

To evaluate the efficiency of the proposed algorithm and its modifications, design optimization of a 10-bar truss and a RC flat slab building is carried out.

Example 1; 10-bar Truss: This test problem has been widely used by other researchers (e.g. Refs. [5], [17] and [18]). The geometry and loading of the truss is presented in Figure 1. Design variables, x_i, are the cross-section of truss members which can take a certain value from a predefined set as given in Eq. (5). The objective function is the total weight of the truss, $W(x)$. The total weight is minimized while the member stresses, σ_i, does not exceed $\pm 25ksi$ and the vertical displacements, δ_j, of nodes D and E, become less than $2in$. The mathematical formulation of the problem is as follows:

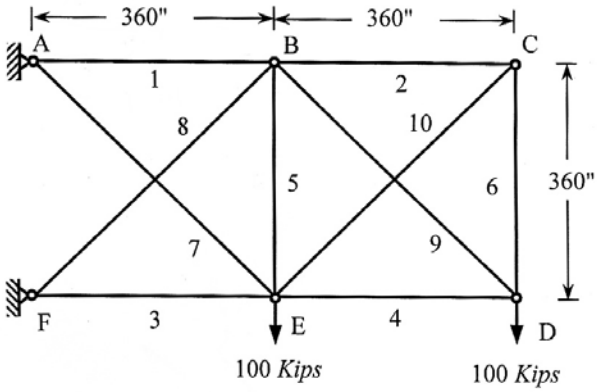

Figure 1. 10-bar truss

$$\text{Minimize } W(\boldsymbol{x}) = \rho \sum_{i=1}^{10} x_i l_i ,$$

$$\text{subject to: } \frac{\sigma_i(\boldsymbol{x})}{25(\text{ksi})} \le 1 \text{ , i = 1, 2,…10} \qquad (5)$$

$$\frac{\delta_j(\boldsymbol{x})}{2(\text{in})} \le 1 \quad \text{j = E, D,}$$

$x_i\,(in^2) \in \{1.62, 1.8, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09, 3.13, 3.38, 3.47, 3.55, 3.63, 3.84, 3.87, 3.88, 4.18, 4.22, 4.49, 4.59, 4.80, 4.97, 5.12, 5.74, 7.22, 7.97, 11.5, 13.5, 13.9, 14.2, 15.5, 16.0, 16.9, 18.8, 19.9, 22.0, 22.9, 26.5, 30.0, 33.5\}$ $i = 1, 2,\ldots 10$

All members are double angle aluminium profiles taken from the AISC manual [18] with the specific weight, ρ, of 0.1 lb/in^2 and modulus of elasticity, E, of 10^4 ksi. Table 1 compares the cross-sectional area of truss elements for the best solution given by Mahfouz [4], Rajeev et al. [17] and Galante [18] and the present study. The best solutions obtained by Mahfouz, Galante and the present study are the same. The current algorithm gives less weight truss than that by Rajeev et al. [17].

Table 1. Comparison of the best solutions given by other researchers and the present study

Design variables	**Rajeev *et al.***	**Galante, Mahfouz and present study**
$x_1\,(\text{in}^2)$	33.5	33.5
$x_2\,(\text{in}^2)$	1.62	1.62
$x_3\,(\text{in}^2)$	22	22.9
$x_4\,(\text{in}^2)$	15.5	14.2
$x_5\,(\text{in}^2)$	1.62	1.62
$x_6\,(\text{in}^2)$	1.62	1.62
$x_7\,(\text{in}^2)$	14.2	7.97
$x_8\,(\text{in}^2)$	19.9	22.9
$x_9\,(\text{in}^2)$	19.9	22
$x_{10}\,(\text{in}^2)$	2.62	1.62
Total weight (lb)	5613.8	5490.7

MULTIMODALITY OF THE PENALIZED OBJECTIVE FUNCTION

The penalized objective function in the above test problem can be a multimodal function of the cross-sections of the truss elements. Table 2 lists the values of the penalized objective

function, $\widetilde{C}(\boldsymbol{X})$, in the fifteenth best solutions listed from the best to the worst in terms of $\widetilde{C}(\boldsymbol{X})$ that can be local optima. These points have been obtained after a local search around GA solutions. There is no explicit expression for the penalized objective function in terms of design variables. Therefore, it can not be mathematically possible to prove these points are local minima of the penalized objective function. To show the variation of the penalized objective function around the best solution the values of the penalized objective function have been calculated at four points in the direction of each coordinate as shown in Figure 2.

Table 2. Values of the penalized objective function for the fifteen best solutions

Solution number	**Values of design variable (lb)**										$\widetilde{C}(\boldsymbol{X})$
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	
1	33.5	1.62	22.9	14.2	1.62	1.62	7.97	22.9	22.0	1.62	5490.738
2	33.5	1.62	22.9	15.5	1.62	1.62	7.97	22.0	22.0	1.62	5491.717
3	33.5	1.62	22.9	14.2	1.62	1.62	7.22	22.9	22.9	1.62	5498.375
4	33.5	1.62	22.9	15.5	1.62	1.62	7.22	22.9	22.0	1.62	5499.354
5	33.5	1.62	22.9	15.5	1.62	1.62	7.22	22.0	22.9	1.62	5499.354
6	33.5	1.62	22.0	13.9	1.62	1.62	7.97	22.9	22.9	1.80	5502.523
7	33.5	1.62	22.9	16.9	1.62	1.62	7.22	22.0	22.0	1.62	5503.934
8	33.5	1.62	22.0	14.2	1.62	1.62	7.97	22.9	22.9	1.62	5504.158
9	33.5	1.62	22.0	15.5	1.62	1.62	7.97	22.9	22.0	1.62	5505.138
10	33.5	1.62	22.0	15.5	1.62	1.62	7.97	22.0	22.9	1.62	5505.138
11	30.0	1.62	22.9	16.9	1.62	1.62	7.97	22.9	22.9	1.62	5507.758
12	33.5	1.62	22.0	16.9	1.62	1.62	7.97	22.0	22.0	1.62	5509.717
13	33.5	1.8	22.9	14.2	1.62	1.99	7.97	22.0	22.9	1.62	5510.538
14	33.5	1.62	22.9	13.5	1.62	1.62	7.97	22.9	22.9	1.62	5511.358
15	33.5	1.62	22.0	15.5	1.62	1.62	7.22	22.9	22.9	1.62	5512.775

These four points are two points before and two points after a presumed local optimum in the direction of a given coordinate. In this figure, each radial line specifies the direction of a

coordinate (design variable) numbered from one to ten for the ten design variables. As it is seen by moving to any point in neighbourhood of the best solution in the direction of each coordinate the value of the penalized objective function increases.

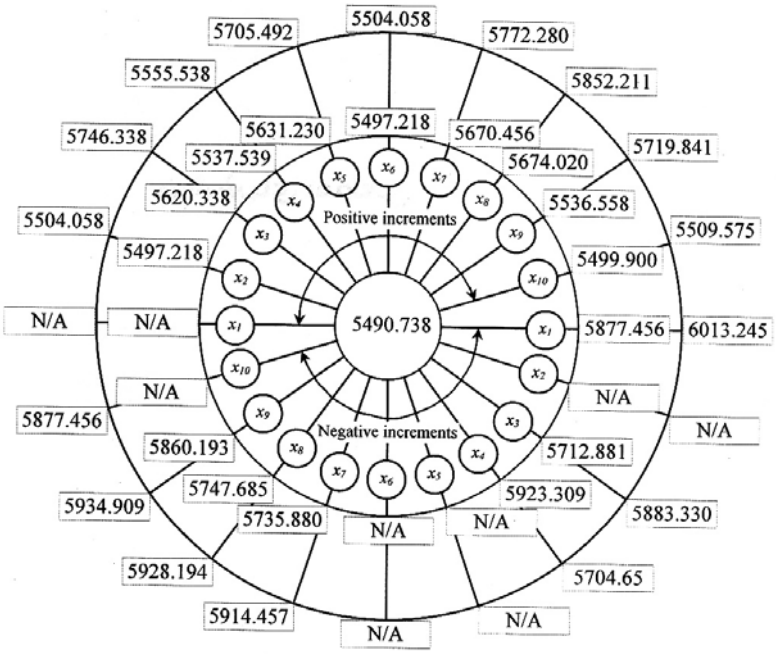

Figure 2. Variation of the penalized objective around the best solution (N/A indicates that a positive or negative increment is out of the defined range for the design variable)

COMPARATIVE STUDIES

The performance of GA is affected by parameters such as population size, probability of crossover and mutation, P_c and P_m, percentage of elite, P_e , method of selection, crossover and encoding and many other parameters. In the following the effect of some of these parameters and methods for the test problem is investigated. These comparative studies are as follows:

Case 1. Comparison of Gray and binary coding.
Case 2. Effect of population sizes.
Case 3. Effect of penalty multiplier.
Case 4. Effect of percentage of elite.
Case 5. Effect of power of penalty function.
Case 6. Comparison of three crossover methods: one-point, two-point and uniform crossover.
Case 7. Comparison of two selection methods: fitness proportional and tournament selection.
Case 8. Comparison of results obtained from GA only and the hybrid GA.

The specifications and some input data for these comparative studies are given in Table 3. In all cases the probability of mutation and crossover are kept fixed at $P_m = 0.01$, $P_c = 1$, respectively.

Table 3. Specifications of different comparative studies considered in Example 1

Case	I_s	S_s	M_s	r	n	P_e	Coding	Crossover	Selection
1	1000	100-600	50	0.75	0.5	10	G and B	ONC	FPS
2	500 and 1500	200-500	50 and 100-500	0.75	0.5	10	G	ONC	FPS
3	1000	100-600	50	0.65-0.9	0.5	10	G	ONC	FPS
4	1000	600	50	0.75	0.5	0-90	G	ONC	FPS
5	1000	600	50	0.75, 1.2 and 1E5	0.5, 1 and 2	10	G	ONC	FPS
6	1000	600 and 700	50, 550 and 600	0.75	0.5	10	G	ONC, TWC and UNC	FPS
7	1000	600	300 and 500	0.75	0.5	10	G	ONC	FPS and TOS
8	1000	300 and 600	50	0.75	0.5	10	G	ONC	FPS

Note: I_s is an initial population size, S_s is the number of the best selected individuals in the second generation, M_s is the minimum allowable population size, r is the penalty multiplier, n is the power of penalty function, P_e is the elite percentage. G and B stand for Gray and binary coding, respectively. ONC, TWC and UNC stand for one-point, two-point and uniform crossover, respectively. FPS and TOS stand for fitness proportional and tournament selections.

BINARY AND GRAY CODING

Table 4 presents the results of implementation of the proposed GA using binary and Gray coding. It can be observed that generally, the average number of function evaluations for Gray coding is larger than that for binary coding and the obtained results for Gray coding are better than those for binary coding. In cases where the average number of function

evaluations of two coding methods are similar, the results for Gray coding are still better than those for binary coding. The frequency of the best solution in 20 runs for Gray coding is either the same or higher than that for binary coding.

Table 4. Comparisons of the results of GA using Gray and binary coding

S_s	Coding method	The best solution (lb)	Frequency of the best solution (20 runs)	Average (lb)	Average number of function evaluations
100	Gray	5523.906	1	5580.279	9182
	Binary	5544.462	1	5670.311	7201
200	Gray	5490.738	1	5562.376	13065
	Binary	5534.742	1	5672.127	11081
300	Gray	5498.374	1	5556.827	17195
	Binary	5533.656	1	5600.956	16158
400	Gray	5491.717	2	5535.129	23361
	Binary	5491.717	1	5572.322	22943
500	Gray	5490.738	3	5519.741	29275
	Binary	5490.738	1	5571.035	28866
600	Gray	5490.738	4	5518.475	34705
	Binary	5491.717	1	5548.819	35732

Note: S_s is the number of the best selected individuals in the second generation.

POPULATION SIZE

In the proposed algorithm, the population size during the GA process, dynamically decreases. Three different values are specified for the population size; those are the initial population size, I_s, the size of the best selected individuals in the second generation, S_s, and the minimum allowable population size, M_s. Different population sizes have been tested. According to Table 3 (Case 2), I_s takes 500 and 1500, S_s changes from 200 to 500 with step size of 100 and M_s takes 50 and varies from 100 to 500 with intervals of 100. For each set of population sizes 20 runs have been carried out and the average penalized objective function

and the number of function evaluations of 20 runs, have been calculated. Figures 3a and 3b show the variation of the average penalized objective function versus M_s for different S_s and I_s.

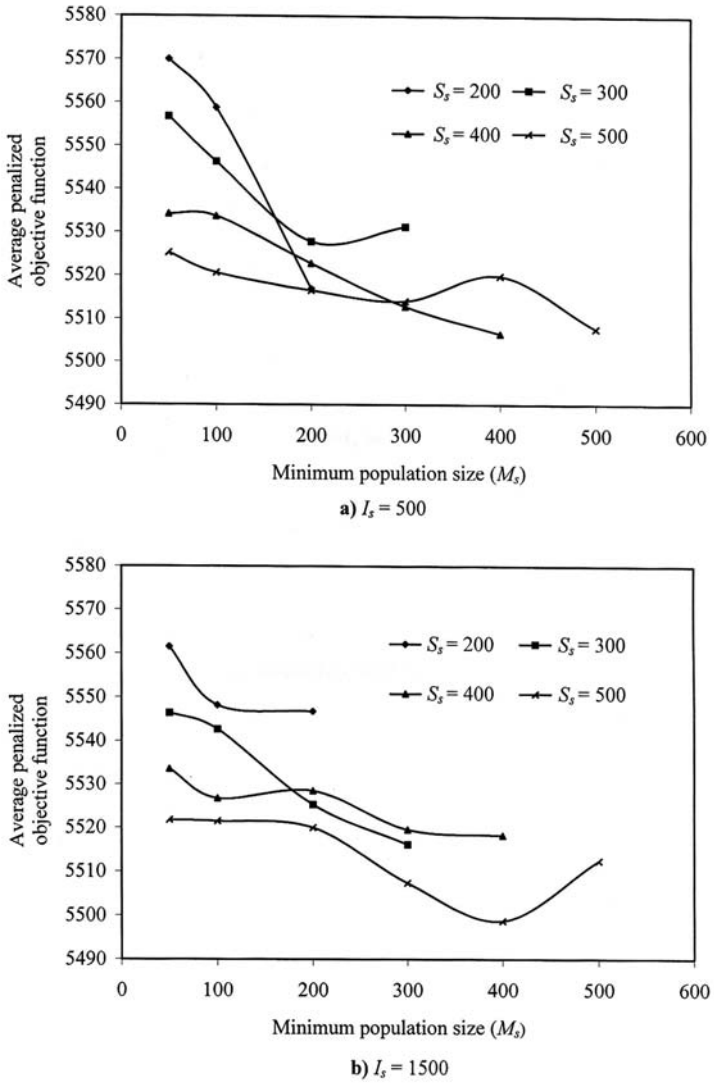

Figure 3. Variation of the average penalized objective function with respect to minimum population size for varying sizes of the best selected individuals

These curves do not present a regular change but, in general, they follow a decreasing trend toward a certain limit. An increasing in any of the population sizes increases variety of chromosomes in the population and as a result the diversity of the population. Figures 4a and 4b show the variation of the average number of function evaluations against M_s for different S_s and I_s.

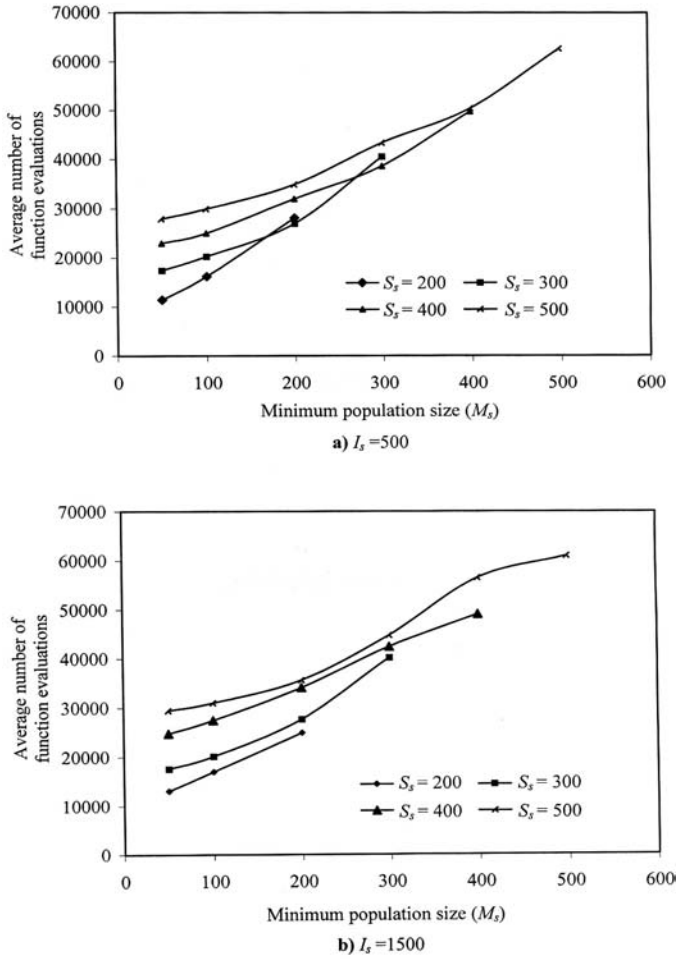

Figure 4. Variation of the average number of function evaluations with respect to minimum population size for varying sizes of the best selected individuals

These figures confirm that the increase of the population size leads to finding a better solution at the expense of greater number of function evaluations. By comparing these figures with Figures 3a and 3b, one can conclude that as the minimum size of population increases, the number of function evaluations increases and, at the same time, the average value of the penalized objective function decreases. However, some sets of population sizes with a relatively small number of function evaluations give better values for the average penalized function. But in general, it can be deduced that to have a good solution the population sizes must be large enough to allow an adequate number of function evaluations before the GA converges to the solution.

This study indicates that the initial population size does not have a significant effect on the optimum solution. With the increasing initial population size from 500 to 2000, for same set of S_s and M_s, the number of function evaluations and the average penalized objective function are not significantly changed. Accordingly, M_s and S_s have the most significant

effect on the performance of the GA, respectively.

PENALTY MULTIPLIER

Three different strategies for applying a penalty multiplier are examined on this test problem. First, a constant penalty multiplier is considered along the GA process. Figure 5 illustrates the effect of the penalty multiplier on the average penalized objective function of the solutions of 20 runs for varying sizes of S_s. It can be seen that in a certain range the increase in the value of the penalty multiplier leads to a dramatic decrease in the average penalized objective function up to a certain limit, beyond which there is almost no major change in the average penalized objective function.

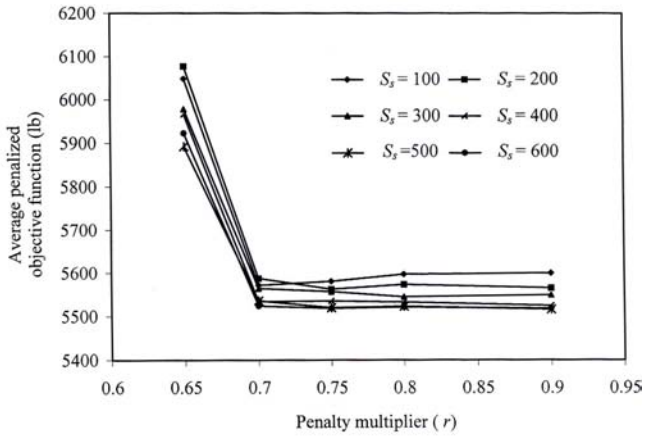

Figure 5. The average penalized objective function versus penalty multiplier for varying sizes of the best selected individuals

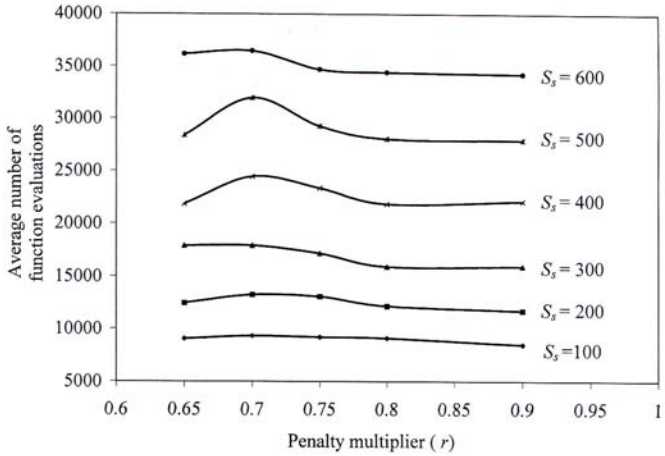

Figure 6. Average number of function evaluations versus penalty multiplier for varying sizes of the best selected individuals

Figure 6 shows the average number of function evaluations plotted against the penalty multiplier for varying S_s. It can be seen that in the range where the average penalized objective function does not dramatically change with the increase of the penalty multiplier, the variation of the number of function evaluations is nearly flat.

Two other approaches implementing of a variable penalty multiplier during a GA process have been examined. In the first approach, the penalty multiplier changes from a relatively large value and linearly decreases to a minimum allowable value (Eq. (4)). In the second approach, the penalty multiplier changes from a relatively small value and linearly increases to a maximum allowable value (Eq. (4)).

Table 5. Comparisons between the results of applying a fixed and variable penalty multipliers

Case	r_{in}	r_f	I_r	N_{fea}	N_{inf}	Average penalized objective function (lb)	Average number of function evaluations
Fixed penalty multiplier	0.75	0.75	0	20	0	5518.475	34705
Variable penalty multiplier	1	0.4	-0.01	7	13	5530.364	34923
	1	0.4	-0.0075	11	9	5510.581	35858
	1	0.4	-0.005	20	0	5522.624	34569
	1	0.4	-0.0025	20	0	5527.804	34043
Variable penalty multiplier	0.4	1	0.01	20	0	5529.139	42773
	0.4	1	0.0075	17	3	5534.776	41837
	0.4	1	0.005	15	5	5553.412	41623
	0.4	1	0.0025	11	9	5600.942	39249

Note: r_{in} is the initial value of penalty multiplier, r_f is the final value of penalty multiplier, I_r is the penalty multiplier increment, N_{fea} is the number of feasible solutions, N_{inf} is the number of infeasible solutions.

Table 5 presents the results for all three strategies. In this table, N_{fea} and N_{inf} specify the number of feasible and infeasible solutions out of 20 runs, respectively. It can be observed that there is no significant preference for a variable penalty multiplier over a fixed penalty multiplier. In most cases, the average number of function evaluations and the average penalized objective function for a fixed penalty multiplier is smaller than the corresponding values for a variable penalty multiplier.

POWER OF PENALTY TERM

Three different powers for the penalty term in Eq. (2) are studied; namely n = 0.5, 1 and 2. The results have been summarised in Table 6. In order to study the effect of the power of penalty function on the GA performance, results of the GA only (i.e. before any

improvement by a local search technique) are presented in this table.

Table 6. Comparisons between different powers for penalty function

n	*r*	The best solution (lb)	Maximum constraint value	Frequency of the best solution (20 runs)	Solutions violating constraints	Average (lb)	Average number of function evaluations
0.5	0.75	5490.738	0.9995	4	0	5519.985	34687
1	0.75	4695.669	1.1803	13	20	4709.700	31327
	1.2	5490.738	0.9995	12	5	5490.824	32733
2	0.75	4075.538	1.373	17	20	4080.228	34304
	1E5	5490.738	0.9995	1	2	5530.246	40533

Note: *n* is the power of penalty function, *r* is the penalty multiplier.

The results show that a power less than one in the penalty function allows using a relatively small penalty multiplier. To avoid any constraint violation, when the power in the penalty function increases, the penalty multiplier has to be dramatically increased. As shown in Table 6, for the power $n = 2$ in the penalty function, a penalty multiplier equal to 1E5 is still insufficient to prevent constraint violation; this has been occurred twice in 20 runs.

Overall, considering the number of solutions violating constrains and the averages of results for each power in the penalty function; it may be concluded that a power less than one in the penalty function performs better than that greater than one.

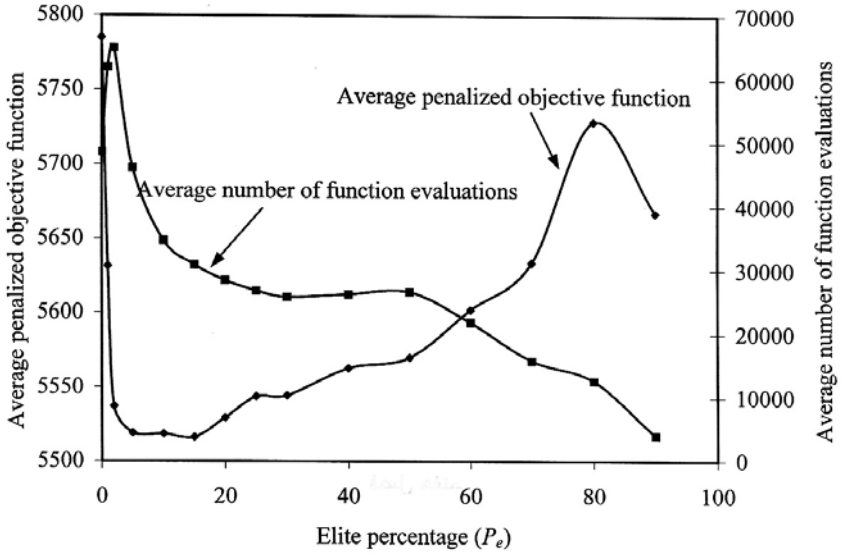

Figure 7. The average penalized objective function and the number of function evaluations against the elite percentage

EFFECT OF THE ELITE PERCENTAGE

Figure 7 shows the variation of the average penalized objective function and the average number of function evaluations against the elite percentage, P_e. By increasing the elite percentage from zero to 90%, the average penalized objective function, initially decreases, passes a minimum and, then, increases with some fluctuations. The average number of function evaluations follows a decreasing trend. In this test problem, a value of elite percentage between 5 to 20% gives better solutions while the number of function evaluations is relatively small.

Table 7. Comparisons of the results for three crossover methods

Crossover method	The best solution (lb)	Frequency of the best solution (20 runs)	Average (lb)	Average number of function evaluations
one-point	5490.738	4	5518.475	34705
Two-point	5490.738	3	5514.501	37877
Uniform	5490.738	8	5507.458	75156

Table 8. Comparisons of the results of one-point and uniform crossover considering the number of function evaluation

Crossover method	S_s	M_s	The best solution (lb)	Frequency of the best solution (20 runs)	Average (lb)	Average number of function evaluations
One-point	600	50	5490.738	4	5518.475	34705
	600	550	5490.738	9	5504.398	68130
	600	600	5490.738	10	5504.568	71877
	700	600	5490.738	11	5499.995	74100
Uniform	600	50	5490.738	8	5507.458	75156

Note: S_s is the number of the best selected individuals in the second generation, M_s is the minimum allowable population size.

EXAMINING THREE METHODS OF CROSSOVER

Table 7 shows the obtained results for three methods of crossover on this test problem. For

all three methods, S_s and M_s are equal to 600 and 50, respectively. It can be observed that the uniform crossover gives better solutions at the expense of much larger number of function evaluations.

Table 8 compares the obtained results for one-point crossover for varying S_s and M_s with that of the uniform crossover while S_s and M_s are equal to 600 and 50, respectively. This table shows that by increasing the population sizes and consequently increasing the average number of function evaluations the results for one-point crossover can be better or close to that for the uniform crossover. This means that for the same number of the function evaluations the uniform crossover does not have an evident preference over one or two-point crossover.

Table 9. Comparisons of two selection methods

M_s	Selection method			The best solution (lb)	Frequency of the best solution (20 runs)	Average (lb)	Average number of function evaluations
300	Fitness proportional			5490.738	11	5513.510	46447
	Tournament	size=2	A	5490.738	3	5527.895	30100
			B	5490.738	1	5546.871	28238
		size=10	A	5529.300	1	5573.311	20700
			B	5513.158	1	5589.317	19553
		size=20	A	5523.906	1	5619.042	19463
			B	5507.758	2	5572.003	21641
500	Fitness proportional			5490.738	7	5517.283	59248
	Tournament	Size=2	A	5490.738	2	5531.087	44495
			B	5490.738	3	5542.656	41514
		Size=10	A	5490.738	1	5555.846	33975
			B	5507.758	1	5551.524	31767
		Size=20	A	5498.375	1	5578.721	30883
			B	5490.738	1	5561.344	32643

Note: M_s is the minimum allowable population size. In Case A, the tournament selection is used in the mating process and the fitness proportional selection is used in the cloning stage. In Case B, the tournament selection is utilized for both mating and cloning processes.

FITNESS PROPORTIONAL AND TOURNAMENT SELECTION METHODS

The obtained results using fitness proportional and tournament selection in a GA are presented in Table 9. Two cases of the use of the tournament selection have been examined. In the first case, the tournament selection is used in the mating process and fitness proportional selection is used in the stage of cloning (Case A). In the second case, the tournament selection is utilized for both mating and cloning processes (Case B). Since a tournament between individuals in a population of a small size leads to a premature convergence, (as comparison of the number of function evaluations for M_s=300 and M_s=500 in the Table 9 confirms this), relatively large values of M_s have been considered (300 and 600). In the GA developed in this paper, all fitness values are scaled using a linear fitness scaling [7]. Therefore, the problem of premature convergence or slow of convergence reported for the fitness proportional selection is reduced. The results show that in this case the tournament selection does not have any advantage over the fitness proportional selection. It can be seen that with the increasing tournament size, the number of function evaluations decreases and the average value of the penalized objective function at the solution increases. This means that the increase in the tournament size may lead to premature convergence. Also, the results indicate that the use of the tournament selection for mating process only performs better than that when this selection scheme is used for both mating and cloning processes (fitness values are scaled).

Table 10. Comparisons of the results of the hybrid algorithm and the GA

Population sizes	GA only		Hybrid GA		Number of improved solutions out of 20 runs
	Average (lb)	Average number of function evaluations	Average (lb)	Average number of function evaluations	
I_s = 1000 S_s = 300 M_s = 50	5560.729	17175	5556.827	17195	4
I_s = 1000 S_s = 600 M_s = 50	5519.985	34687	5518.09	34705	3

Note: I_s is an initial population size, S_s is the number of the best selected individuals in the second generation, M_s is the minimum allowable population size.

COMPARISON OF THE GA AND THE HYBRID GA

The results of 20 runs of the program aimed at the comparison of the modified GA and the developed hybrid GA are illustrated in Table 10. This table indicates that using the hybrid GA can perform better than GA at the expense of only a few more function evaluations. In the following another example on a more practical problem is given to show the effectiveness of the hybrid GA. As it is illustrated, in this case the hybrid GA is more efficient as compared to the first example.

Example 2; Design Optimization of a Flat Slab Building: Design optimization of a three-storey flat slab building of height 4 *m* in the first and 3 *m* in the second and third floors is carried out. The span lengths are as shown in Figure 8. Live load is 5 kN/m^2 and dead load is 2.5 kN/m^2 plus the self-weight of the floor. The unit prices of concrete, reinforcement, formwork and excavation cost of foundations, including the cost of labour and material, are 55 $£/m^3$, 0.5 *£/kg*, 20 $£/m^2$ and 20 $£/m^3$, respectively. The characteristic strength of main and shear reinforcement and concrete are 460 and 250 and 35 N/mm^2, respectively. Four types of cross-sectional dimensions are considered for columns in each floor which are a corner column, an edge column in each direction of the building and an intermediate column. It is assumed that cross-sectional dimensions of columns for each type in the first and second floor are identical. Therefore, there are 16 design variables for column cross-sectional dimensions and 3 design variables for thickness of floors. The binary string of design variables includes 82 binary bits which has been formed from sixteen segments of four bits and three segments of six bits. The aim is to minimize the total cost of the structure subject to some constraints defined based on British Code of Practice [19].

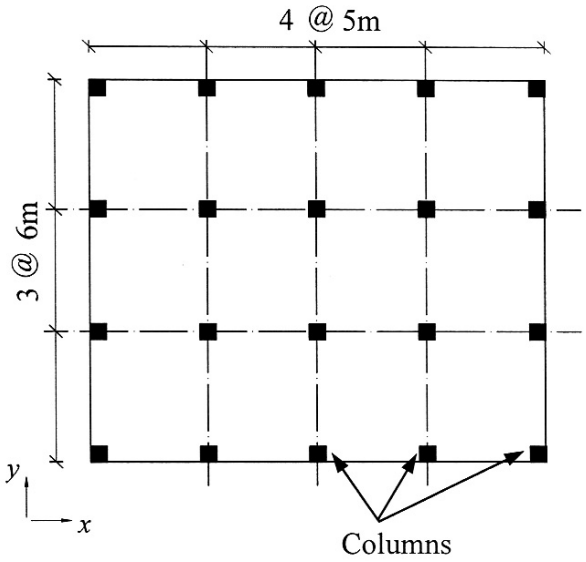

Figure 8. Typical plan of a three-storey RC flat slab building

$$\begin{aligned} &\text{Minimize:} \quad C(\boldsymbol{x}) = V_c\, u_c + W_r\, u_r + S_f\, u_f + V_e\, u_e \\ &\text{subject to:} \quad G_i(\boldsymbol{x}) \le 1 \qquad i = 1, 2, \ldots, m, \\ &\qquad\qquad x_j^l \le x_j \le x_j^u \qquad j = 1, 2, \ldots, n, \end{aligned} \tag{6}$$

where $C(\boldsymbol{x})$, is the total cost, V_c, W_r, S_f and V_e are the total volume of concrete, weight of reinforcement, area of formwork and foundation excavation for a quarter of the building, u_c, u_r, u_f and u_e represent the unit cost of labour and material for concrete, reinforcement, formwork and foundation excavation, respectively. $G_i(\boldsymbol{x})$ is the i-th constraint function as defined by BS 8110 [19].

The results of 10 runs using the GA only and the hybrid algorithm are illustrated in Table 11. As it is observed by spending a little time for complementary search the GA solution can be significantly improved. There are no identical solutions when GA only is used. But after modification of the GA solutions by a local search some identical solutions are found. Also, several runs have been carried out using GA only. Population sizes and some GA parameters were changed to increase the accuracy of GA solution. As a result by spending 3.77 times more of relative computing time an optimum solution of £12858, identical to the best result of hybrid GA, was achieved.

Table 11. Performance of GA only and a hybrid algorithm on design optimization of a RC flat slab building

Run	**GA only**		**Hybrid algorithm (GA and discretized form of Hooke and Jeeves method)**	
	Cost function (£)	**Relative computing time**	**Cost function (£)**	**Relative computing time**
1	13850	0.55	12971	0.68
2	14140	0.14	13503	0.2
3	13645	0.71	12992	0.89
4	14014	0.25	12858	0.29
5	13885	0.36	12858	0.46
6	13866	0.41	12978	0.45
7	13340	0.88	12926	1
8	14031	0.41	12906	0.5
9	13805	0.41	12976	0.48
10	14161	0.31	12906	0.36

CONCLUSIONS

A hybrid algorithm based on GA and some modifications on basic GA were proposed. The proposed algorithm and modifications were examined using a common test problem and then applied to design optimization of a reinforced concrete flat slab building. The main conclusions of this study can be drawn as follows:

- The number of function evaluations has to be considered as a criterion to judge the performance of a GA. Generally, better solutions are obtained at the expense of more function evaluations.
- On the test problem Gray coding performed better than basic binary coding.
- In order to obtain a good solution the population size of the best selected individuals in the second generation, S_s, and the minimum population size, M_s, should be large enough to provide an adequate number of function evaluations before GA converges to the solution.
- When the fitness values are scaled, the tournament selection does not show preference over the fitness proportional selection scheme. When the population size is relatively small the tournament selection may lead to premature convergence.
- When the constraints are introduced in the normalized form, a power less than one in the penalty function can considerably improve the efficiency of GA.
- For the same number of function evaluations, none of the one-point, two point or uniform crossover schemes shows any evident preference over others.
- A large elite percentage may lead to premature convergence. On the other hand a small elite percentage (less than 5% in the selected test problem) may cause losing the advantage of elitist strategy. On the basis of the obtained results an elite percentage between 5 to 15% is recommended.
- A complementary local search on the promising area of design space, found through the global search by GA, can improve GA solution at the expense of only a few more function evaluations.
- The performance of the proposed hybrid algorithm may change from a problem to another depending on the complexity of design search space.

REFERENCES

1. Holland, J. H., *Adaptation in natural and artificial systems*, University of Michigan, Ann Arbor, USA, 1975.
2. Goldberg, D. E., *Genetic algorithms in search optimization and machine learning*, Addison-Wesley Publishing Co. Reading, Massachusetts, 1989.
3. Gen, M. and Cheng, R., *Genetic algorithms and engineering optimization,* John Wiley and Sons, Inc., New York, 2000.
4. Chen, T. Y. and Chung, J. C., Improvement of simple genetic algorithm in structural design, *International Journal of Numerical Methods in Engineering*, **40**(1997)1323-1334.
5. Mahfouz, S.Y., "*Design Optimization of Structural Steelwork*", Ph.D. thesis, University of Bradford, U.K., 1999.

6. Sahab, M. G., Toropov V. V. and Ashour, A. F., Multilevel optimization of reinforced concrete flat slab buildings based on genetic algorithm, *Proc. of 3rd ASMO UK/ISSMO Engineering Design Optimization Conference*, Harrogate, UK, 9-10 July 2001, pp. 243-248.
7. Michalewicz, Z., A survey of constraint handling techniques in evolutionary computation methods, *Proc. 4th annual conference on evolutionary programming*, MIT Press, Cambridge, 1995, pp. 135-155.
8. Lin, C. Y. and Hajela, P., Genetic Algorithms in optimization problems with discrete and integer design variables, *Engineering Optimization*, **19**(1992)309-327.
9. Yang, J. and Soh, C.K., Structural optimization by genetic algorithms with tournament selection, *Journal of Computing in Civil Engineering*, ASCE, No. 3 **11**(1997)195-200.
10. Camp, C., Pezeshk, S. and Cao, G., Optimized design of two-dimensional structures using a genetic algorithm *Journal of Structural Engineering*, ASCE, No. 5, **124**(1998)551-559.
11. Wright, A. H., Genetic algorithms for real parameter optimization, Rawlins G. Ed., *Foundations of genetic algorithm*, California, Morgan Kaufmann Publishers, 1991, pp. 205-218.
12. Whitley, D., The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best. *Proc. of 3rd International. Conference. on Genetic algorithms*, Schaffer, J.D., Ed., Morgan Kaufmann Publishers, California, 1989, pp. 116-121.
13. Mitchell, M., *An introduction to genetic algorithms*, MIT Press, USA, 1998.
14. Coley, D. A. , *An introduction to genetic algorithms for scientists and engineers*, World Scientific Publishing Co. Pte. Ltd., Singapore, 1999.
15. DeJong, K. A., An analysis of the behaviour of a class of genetic adaptive system. Ph.D. thesis, University of Michigan, 1975.
16. Hooke, R. and Jeeves, T. A., Direct search solution of numerical and statistical problems, *Journal of Assoc. Computer Mach.*, **8**(1961)212-229.
17. Rajeev, S. and Krishnamoorthy, C. S. Discrete optimization of structures using genetic algorithms, *Journal of Structural Engineering*, ASCE, No. 5, **118**(1992)1233-1250.
18. Galante, M., Genetic algorithms as an approach to optimize real world trusses, *International Journal for Numerical Methods in Engineering*, **39**(1996)361-382.
19. British Standards Institution, Structural use of concrete, Part 1, Code of practice for design and construction, BS 8110, BSI, London, 1997.