Direct index coding for discrete sizing optimization of structures by genetic algorithms

A. Kaveh[1,2] **and M. Shahrouzi** [2]

[1]Professor, Iran University of Science and Technology, Narmak, Tehran, Iran
E-mail: alikaveh@iust.ac.ir

[2] International Institute of Earthquake Engineering and Seismology, Tehran, Iran

Abstract: Genetic Algorithm is known as a generalized method of stochastic search and has been successfully applied to various types of optimization problems. By GA's it is expected to improve the solution at the expense of additional computational effort. One of the key points which controls the accuracy and convergence rate of such a process is the selected method of coding/decoding of the original problem variables and the discrete feasibility space to be searched by GA's. In this paper, a direct index coding (DIC) is developed and utilized for the discrete sizing optimization of structures. The GA operators are specialized and adopted for this kind of encoded chromosomes and are compared to those of traditional GA's. The well-known 10-bar truss example from literature is treated here as a comparison benchmark, and the outstanding computational efficiency and stability of the proposed method is illustrated. The application of the proposed encoding method is not limited to truss structures and can also be directly applied to frame sizing problems.

Keywords: *genetic algorithm, structural size optimization, direct index coding, direct uniform crossover, Direct Index Mutation, Adjacent Index Mutation, binary expanded mutation.*

1. INTRODUCTION

Despite most deterministic optimization methods, genetic algorithms take the advantage of starting from a random set of initial search points and seek the remainder of such a discrete feasibility space by their special walks. Such movements are performed by the GA operators; initiation and reproduction (selection, crossover and mutation). Sharing the genetic memory of parent chromosomes through crossover and selecting the fitter parents, it is aimed to produce fitter children during the iterative generations, while avoiding being trapped in local optima by employing random operators (mutation).

In order to apply such a search algorithm to real world problems, it is first necessary to define a mapping between the main optimization feasibility space and the GA discrete search space. Each solution alternative is called Phenotype in the problem space and Genotype when mapped (coded) to the genetic space. A number of encoding methods have already been employed by various investigators, such as Binary or Bit-string coding [1,2], Gray coding [3,4], Integer coding [5,6], Real coding [7,8,9] and also mixed schemes [10]. John Holland in his innovative article on GA, emphasized that the binary coding corresponds to the largest number of schemata for any given parameter resolution [11]. However, it does not mean the most powerful GA, since it can principally be inspired by any other types of alphabetic characters or coding schemes. In addition, binary coding often causes variable Hamming distances between the decoded values (genotypes) of adjacent phenotypes. This causes the preference of some directions as opposed to others during the genetic search. Though this affects the convergence rate of GA in a number of problems [12], it is

not always the case [13,14]. Gray coding removes Hamming cliffs at the expense of adding extra conversion to the binary representation of each individual. The reason is that in gray coding any movement from the mapped genotype of the corresponding phenotype to its next variable is performed only by 1 bit-exchange, thus preserving the constant Hamming distance of 1 between the adjacent numbers in both spaces.

Another problem is the approximation involved while decoding genotypes to the real parameters. Rajeev and Krishnamoorthy [15] suggested a two-phase coding method meanwhile improving the speed of GA convergence. In the first phase, the design domain is divided into a number of subdivision groups to enforce an explicit coding/ decoding scheme, but it may be exploited in the second phase when a filler index is required to associate any decoded substring to the corresponding desired index within each group. Such a filler index may be selected by heuristics or even reasonably approximated from the results of the structural analysis by force method, as introduced by Kaveh and Kalatjari [16].
Real value or floating-point coding is a systematic attempt to reduce such an approximation, specially for continuous design spaces, however, it may not lead to an accurate correspondence between the phenotypes and genotypes, esp. while decoding.

Additionally, in the mentioned coding schemes extra imposed bias to GA through encoding may limit its search to a special manner. A Direct Index Coding is proposed in this paper not only to remove such a bias, but also to preserve an explicit one-to-one correspondence between the discrete list of design section indices and GA characters. Consequently, the proper genetic operators are adopted for this type of chromosome and applied to the well-known ten-bar truss example as a comparison benchmark.

2. BASIC DEFINITIONS AND CONCEPTS

Genetic Algorithms deal with a set of design trials called population of individuals and aim to improve desirability (fitness) of the best individual through iterations of its search. The population in every iteration is called a generation and its new individuals (children) are produced from selected previous individuals (parents) by GA operators. An individual consists of a chromosome, i.e.; a string of genes or characters. The gene value is called allele. As shown in Fig. 1, the flowchart of a simple GA process includes initiation and reproduction. The latter itself consists of selection and mating.

First, by associating random values to genes, an initial population of chromosomes is generated and their corresponding fitness values are evaluated. In this stage, the objective function of the problem model should be calculated (e.g. through structural analysis) for any decoded chromosome as a design alternative. Then the same number of fitter individuals are selected and put to an intermediate population called mating pool. Mating consists of crossover and mutation. In crossover, a number of genes in two individual chromosomes (parents) are exchanged in order to generate new ones (children or offsprings). Then some characters of these children chromosomes are randomly changed according to the mutation procedure. The whole process is repeated for more generations until the convergence criteria is satisfied and ended by announcing the fittest individual as the optimal design

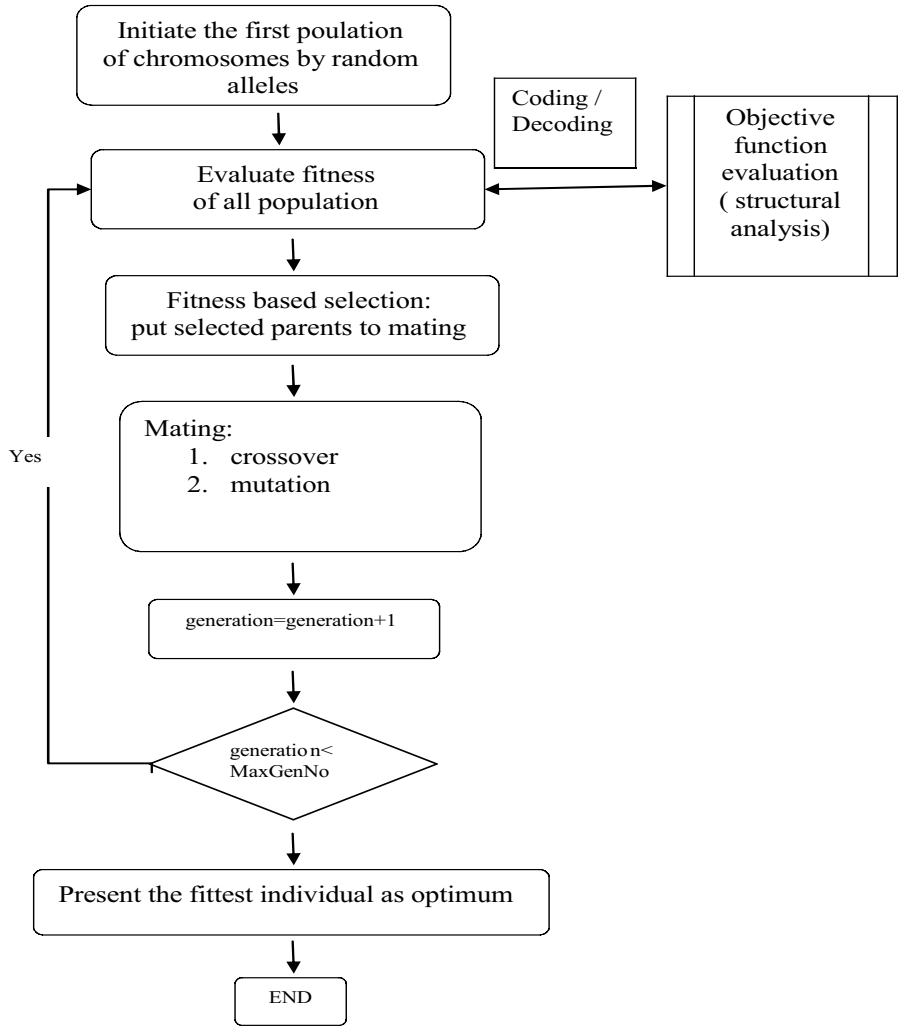


Figure 1. Flowchart of (structural) optimization by standard genetic algorithm.

solution of the problem.

The simplest and most common representation of a gene value (allele) is binary representation (zero/one); however, considering the described genetic process, GA is by no means limited to a special coding method. Since each gene in a binary string may represent only *2* alternative values (zero or one), a chromosome with *m* genes will address 2^m variants; *0* to 2^{m-1}. Therefore, to represent an *n*-alternate variable when *n* is not a power of *2,* *m*-digit string will be needed so that 2^m is more than *n*. The greater the design parameter upper-bound (*n*) is, the more the difference between it and the coded value alternatives (2^m) may arise. This may disaffect the accuracy and convergence rate of the genetic algorithm. Hajela [1] suggested that these out-of-bound values be penalized during fitness evaluation. Some other researchers [17], recommend the following formulation for associating a real parameter to any binary alternative dominated by bit-strings:

$$R = R_{\min} + (R_{\max} - R_{\min}).\frac{B}{2^m}$$

where R denotes the floating-point equivalent of the binary value, B, and varies between R_{min} and R_{max} bounds. The number of bits required for binary decoding is denoted by m. Obviously, the accuracy in evaluating such a formula is limited to the working precision of the computer hardware.

3. DIRECT INDEX CODING AND RELATED GENETIC OPERATORS

The following facts led the authors to utilize direct index coding for a more efficient genetic search in discrete problems:

1. To maximize the effectiveness of a stochastic search, it is desirable to avoid any unnecessary bias or priority transfer from the problem feasibility space to the GA coded search space. As the information transfer is only required during the fitness evaluation stage of a genetic algorithm (for example by analysis of the structural model), the coding/decoding method should not enforce any dependence to the physical parameters of the problem.

2. Since the gap between the decoded value and the corresponding discrete parameter will increase for greater parameter resolution, it is desirable to use a character which can address design parameters with a minimal range of variations.

Suppose that the unsorted list of available discrete parameter alternatives has n items, for example n section properties for a structural sizing problem. Disregarding the physical values of such a parameter variants, it can be minimally addressed by n-indices; 1 to n. When such indices for each design parameter are directly used as alternative values for the corresponding gene, a minimal length chromosome for the problem will be obtained. Here, such a method is called the Direct Index Coding (DIC).

The discrete variations of any parameter j is addressed by n_j indices, then in the initiation process the corresponding gene may be assigned a random character between 1 and n_j. Certainly, these bounds for each parameter may not be violated during gene exchange between two distinct chromosomes in the adopted crossover process. Figure 2

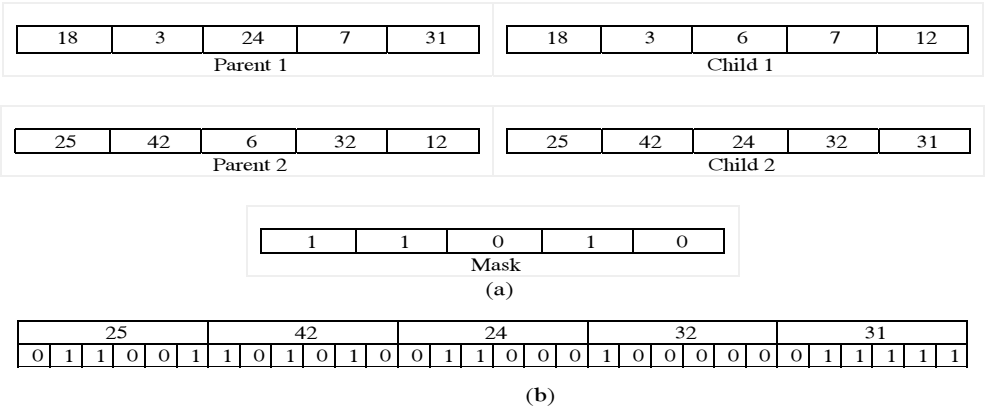


Figure 2. a) Direct uniform crossover applied to sample direct index coded chromosomes (here =42, 2^5<42<2^6), b) Binary expansion of child-2 and possible sites of exchange during direct uniform crossover. Also note that all 6 bits should be exchanged to move from binary representation of adjacent values 32 and 33 meanwhile only 1 bit-change is needed between 25 and 24 (Variable Hamming distance problem).).

shows how traditional types of crossover are adopted to these chromosome types in order to obtain the desired children. By expanding a character value to its binary string, one may note that a *Direct Uniform Crossover* corresponds to a specialized type of binary n-point crossover in which sites of exchanges are preserved at the interconnections of binary substrings.

Despite the crossover operators, the binary mutation process can not be directly extended to such index coded chromosomes, because now any character of a gene will address n_j distinct conditions instead of only two values (0 or 1). One way is to apply the traditional mutation to the binary expansion of each gene. This is so-called, *Binary Expanded Mutation*, BEM, Fig. 3. It should be mentioned that such an intermediate binary coding/decoding may intoduce undesirable values greater than for any jth gene in the chromosome. The following formula may be employed for probable out-of-bound values, however, binary coding adds the problem of Hamming cliffs bias:

$$D = \frac{n_j}{2^m}(B)_{10} \quad , \quad \text{if} \quad (B)_{10} > n_j$$

$$D = (B)_{10} \quad , \quad \text{otherwise.}$$

in which D is the equivalent index value for the mutated binary representation, B, of the original j^{th} gene value.

The second method is defined as *Direct Index Mutation* (DIM), in which if a randomly generated number for every gene falls below a pre-assigned value,P_m . The corresponding allele index is randomly exchanged with another index in the range [1,n_j]. Direct Index Mutation is somewhat like the Jump Mutation operator for real coding [18], but it has the additional feature of preventing violation of the index range from bounds of each phenotype.

Since, DIM may cause considerably large allele jumps, the third method of encoding in DIC is developed. It is so-called *Adjacent Index Mutation* (AM), and defined through steps of the following sub-algorithm:

1.For each gene in the chromosome, generate a random value, r_1, in the range [0 , 1].

D_1= **25**						D_2= **42**						D_3= 24						D_4= 32						D_5= **31**					
0	1	1	0	**0**	1	1	**0**	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	**0**	1	1	1	1	**1**

(a)

1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0

(b)

D_1= **27**						D_2= **58**						D_3= 24						D_4= 32						D_5= **62**					
0	1	1	0	**1**	1	1	**1**	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	**1**	1	1	1	1	**0**

(c)

Figure 3. a) Binary expansion of a sample direct indexed chromosome, b) corresponding mutation mask with 20% uniformity, c) The mutated and contracted chromosome after BEM. In this example binary mutation has caused out-of-bound values (D_2=58, D_5=62 >n_j =42).

Chromosome before mutation	25	42	24	32	31
Mutation mask (60% uniformity)	0	0	1	1	0
Chromosome after sample Direct Index Mutation	17	6	24	32	11
Chromosome after sample Adjacent Index Mutation	26	41	24	32	32

Figure 4. Example of Direct Mutation and Adjacent Mutation applied onto the sample chromosome of Fig.3.

2. If r_1 is less than a pre-assigned mutation probability, P_m , generate another random value r_2 , otherwise retain the gene value unchanged.

3. If $r_1 < P_m$ and $r_2 < 0.5$, then shift the character index value, D, to its next lower alternative and otherwise to its upper neighbor index; i.e.

$$D = \max(1, D-1) \quad , \quad r_2 < 0.5$$

$$D = \min(n_j, D+1) \quad , \quad r_2 \geq 0.5$$

It should be mentioned that by Adjacent Index Mutation, the [1, n_j] index range for any parameter is never violated, while the randomness of the process is also preserved, Fig. 4. This type of mutation leads to smooth shifting between adjacent placements of each parameter in its corresponding list. The following treated example shows the efficiency of this method.

Elitist Chromosome Strategy is also employed in this paper to avoid the loss of the best genetic material during further generations. According to this strategy, the fittest chromosome of any generation takes the place of the least fit individual in the next generation. The constrained optimization problems are changed to an unconstrained form to allow more individuals to be generated in the GA search. Although, in order to obtain only feasible solutions as the optimum GA does, iterate until at least one chromosome satisfies all the problem constraints at any generation. By means of the elitist strategy, such an iteration will no more require extra generations as soon as the first elitist individual satisfies all the constrained.

In each iteration of the GA, the mating pool of parents are selected based on the fitness competition of the previous generated chromosomes by various methods such as *Tournament Selection, Proportionate Selection*, etc.

According to the Proportionate Selection based on the *Roulette Wheel Rule*, in any population of the size, PopSize, the chance of a k^{th} individual with fitness value of F_k , to be selected for the mating pool is:

$$\frac{F_k}{\sum_{k=1}^{PopSize} F_k}$$

In the tournament method, each parent will be the fitter of two randomly selected individuals out of the population. The process is repeated until the mating pool is filled with the same number of individuals as in the current population.

4. AN ILLUSTRATIVE EXAMPLE

The well-known example of 10-bar truss is selected from literature as a benchmark. Boundary conditions, dimensions and applied loads (P = 445.4kN) are as depicted in Fig. 5. The allowable limits for stress and

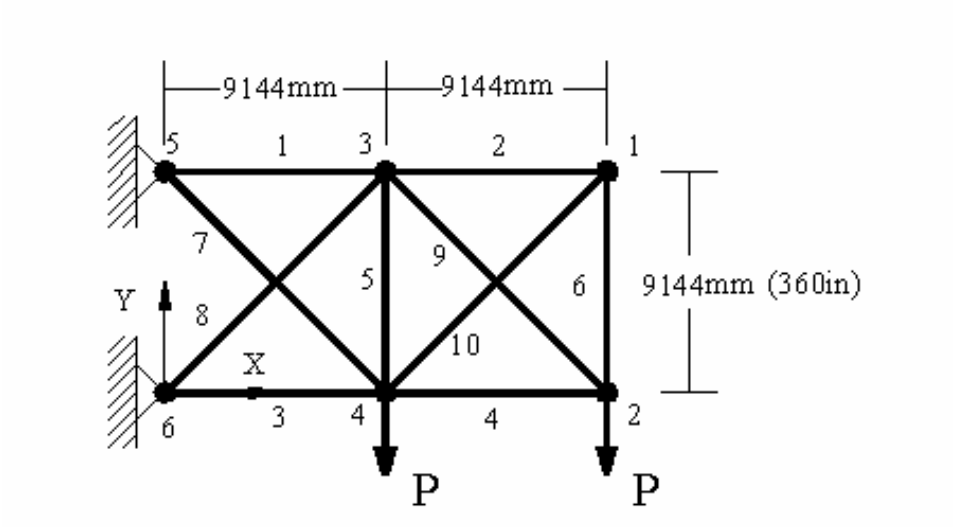


Figure 5. Boundary conditions, and dimensions of the 10-bar truss.

displacement are set to σ_a= *172MPa* and Δ_a= *50.8mm*, respectively.

Material density is 2770 kg/m^3 for all sections in Table 1. The truss is statically indeterminate, so the stress or displacement responses are dependent to the selected member properties in each solution individual and should be obtained by structural analysis.

The cross-sectional area for any of the 10 truss members can be selected from the 42-fold list of Table 1, thus in this case, the index range for all 10 genes of the corresponding direct indexed chromosome will be the same (n_j = 42 for j=1,2,…,10). As a result, the whole discrete search space will consist of 42^{10} combinations.

The objective of this problem is to determine minimal weight truss satisfying stress and displacement constraints. The fitness of each chromosome (as a design condition), is evaluated by the following formula with the penalty coefficient K=10 as given by Rajeev and Krishnamoorty [19]:

$$F = F_0 - f \times (1 + K \times C) \quad , \quad F_0 = 0$$

Where f denotes the total weight of the truss evaluated for each individual. C stands for normalized constrained violation and is computed for critical i,j in the following relations:

$$C = \sum_h \max(0, C_h) \ ,$$

$$C_1 = |\frac{\sigma_i}{\sigma_a}| - 1 \quad , \quad C_2 = |\frac{\Delta j}{\Delta_a}| - 1$$

σ_i, σ_a denote the computed stress for the i^{th} member and its allowable limit, respectively. The corresponding j^{th} node's displacement

Table 1. The indexed available section list for 10-bar truss example [7].

Index	Area (mm^2)	Index	Area (mm^2)	Index	Area (mm^2)
1	1045	15	2342	29	7419
2	1161	16	2477	30	8710
3	1284	17	2497	31	8968
4	1374	18	2503	32	9161
5	1535	19	2697	33	10000
6	1690	20	2723	34	10323
7	1697	21	2897	35	10903
8	1858	22	2961	36	12129
9	1890	23	3097	37	12839
10	1993	24	3206	38	14193
11	2019	25	3303	39	14774
12	2181	26	3703	40	17097
13	2239	27	4658	41	19355
14	2290	28	5142	42	21613

Table 2. Genetic parameters applied to the 10-bar truss example.

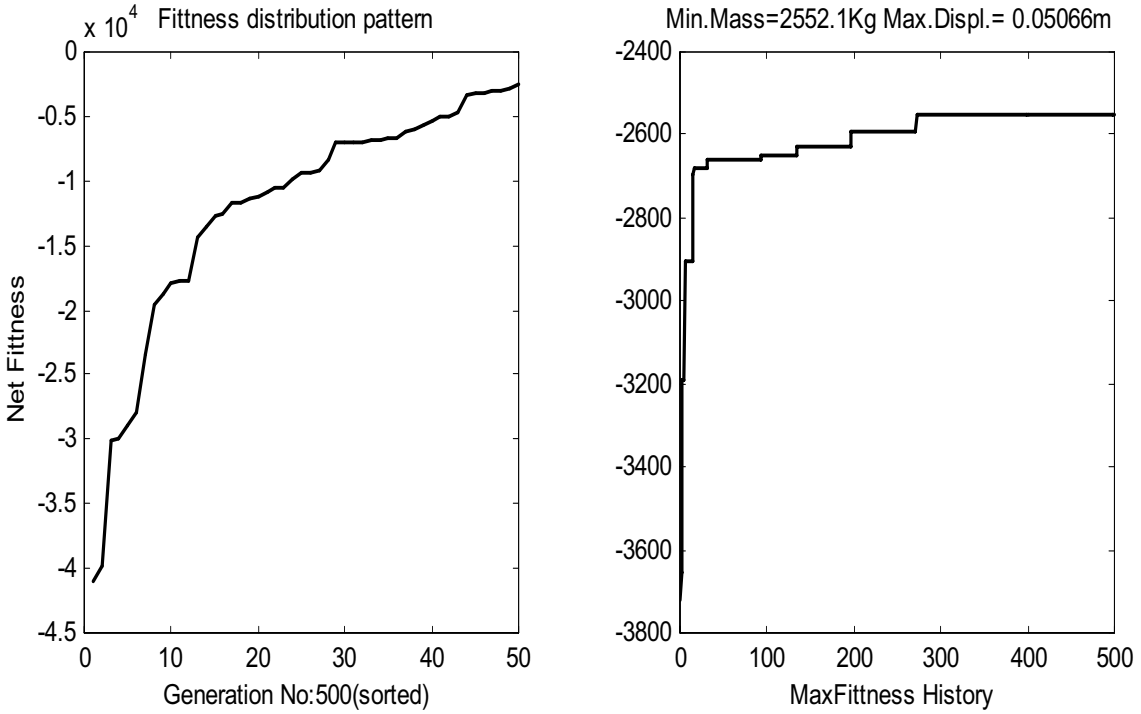
Work	Method of encoding	Selection Method	PopSize by Generation Number	Crossover Type	Pc	Mutation Type	Pm
Rajeev and Krishnamoorty (1992) [19]	Binary	Proportionate	40	2-point	<0.85	Binary	<0.05
Turkkan (2003) [7]	Real	Tournament	300 by 2000	Whole-linear	0.85	Non-uniform	0.05
Present Work-1	DIC	Tournament	50 by 500	Uniform (40%)	0.85	BEM	0.05
Present Work-2	DIC	Tournament	50 by 500	Uniform (40%)	0.85	DIM	0.05
Present Work-3	DIC	Tournament	50 by 500	Uniform (40%)	0.85	AIM	0.05
Present Work-4	DIC	Tournament	50 by 200	1-point	0.85	BEM	0.10
Present Work-5	DIC	Tournament	50 by 200	1-point	0.85	DIM	0.10
Present Work-6	DIC	Tournament	50 by 200	1-point	0.85	AIM	0.10

Table 3. Comparison of the obtained results for the 10-bar truss example.

Work	Method	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	Total mass (kg)	Max. nodal displacement (m)
Rajeev and Krishnamoorty (1992) [19]	Binary	42	1	38	33	1	1	32	37	37	6	2546	0.0505
Cai and Thierauf (1993) [20]	Binary	42	1	39	32	1	1	28	39	38	1	2490	0.0508
Coello (1994) [21]	Binary	41	1	39	30	1	1	31	38	38	1	2538	0.0505
Galante (1996) [22]	Binary	42	1	38	32	1	1	28	39	38	1	2475	0.0511
Ghasemi et al. (1999) [23]	Binary	42	1	38	31	1	1	28	39	39	1	2491	0.0512
Turkkan (2003) [7]	Real	42	1	39	32	1	1	28	39	38	1	2490	0.0508
Present Work 1	DIC+ BEM	42	9	39	31	2	1	29	38	37	9	2552	0.0507
Present Work 2	DIC+ DIM	41	2	39	32	1	1	30	37	39	6	2538	0.0508
Present Work 3	DIC+ AIM	41	1	39	35	1	1	28	39	39	1	2498	0.0507
Present Work 4	DIC+ BEM	38	25	42	29	2	18	34	37	38	26	2740	0.0502
Present Work 5	DIC+ DIM	41	1	39	33	1	3	29	40	36	1	2551	0.0505
Present Work 6	DIC+ AIM	42	1	39	32	1	1	28	39	38	1	2490	0.0508

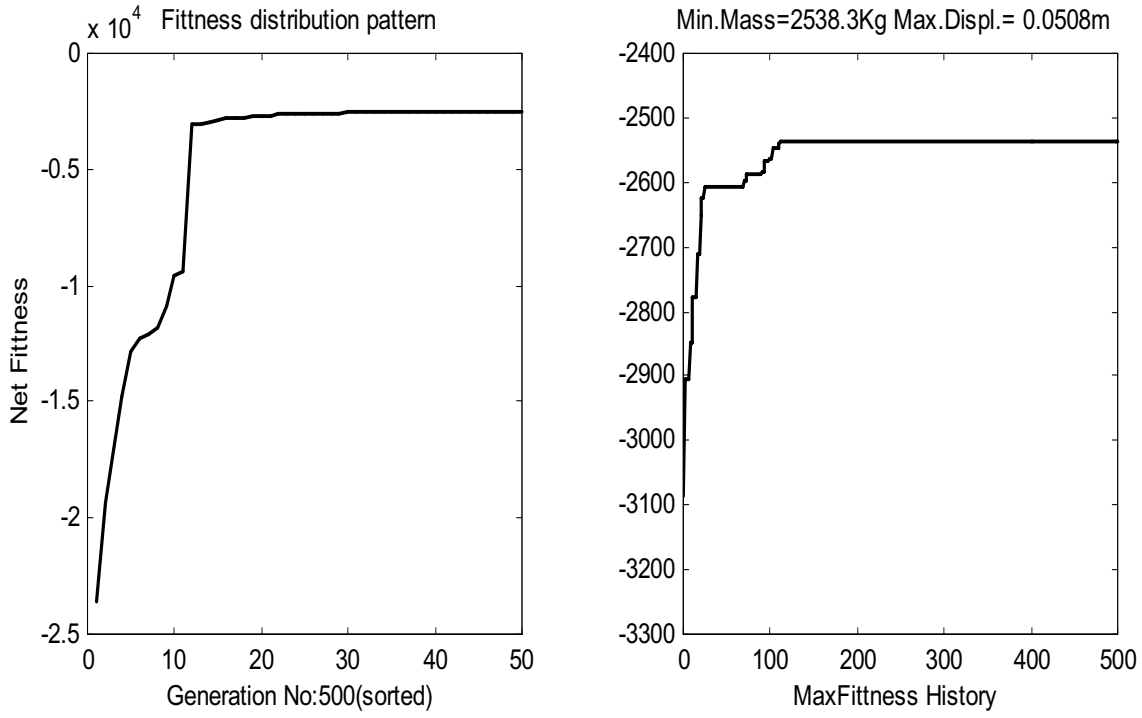
Table 4. Effect of increasing mutation rate meanwhile reducing number of genereations on the results.

Work	Method of encoding	PopSize by Generation Number	Crossover Type	Pc	Mutation Type	Pm	Total mass (kg)	Max. Fitness Improvement Lag
Present Work-7	DIC	50 by 100	Uniform (40%)	0.85	BEM	0.10	2807	28
Present Work-8	DIC	50 by 100	Uniform (40%)	0.85	DIM	0.10	2567	24
Present Work-9	DIC	50 by 100	Uniform (40%)	0.85	AIM	0.10	2549	12



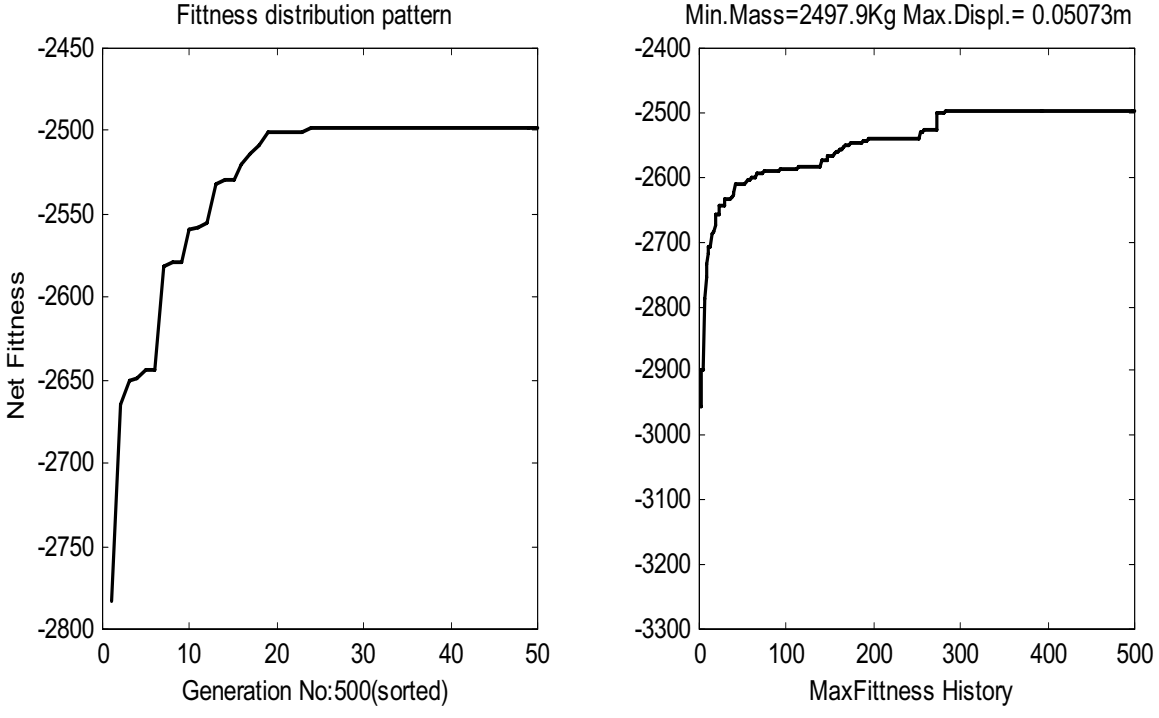
(a) (b)

Figure 6 . a) Sample fitness distribution , b) Maximum Fitness history and the corresponding total mass and displacement response applying BEM in DIC after 500 generations.



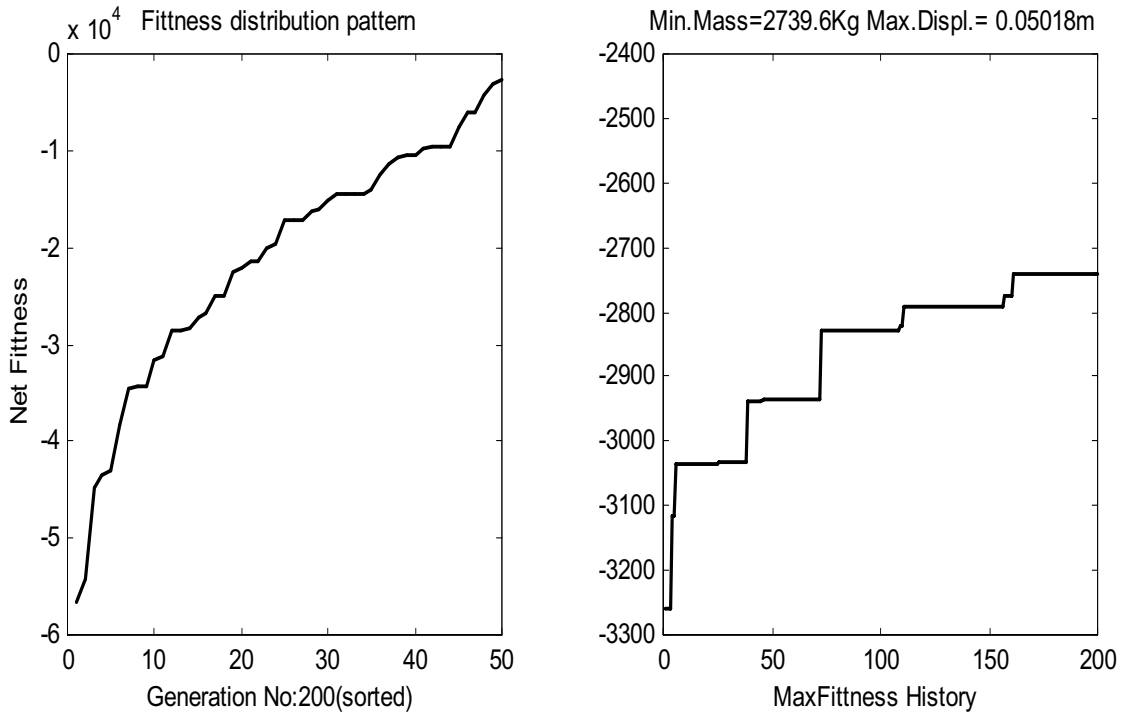
(a) (b)

Figure 7 . a) Sample fitness distributaion , b) Maximum Fitness history and corresponding total mass and displacement response applying DIM in DIC after 500 generations.



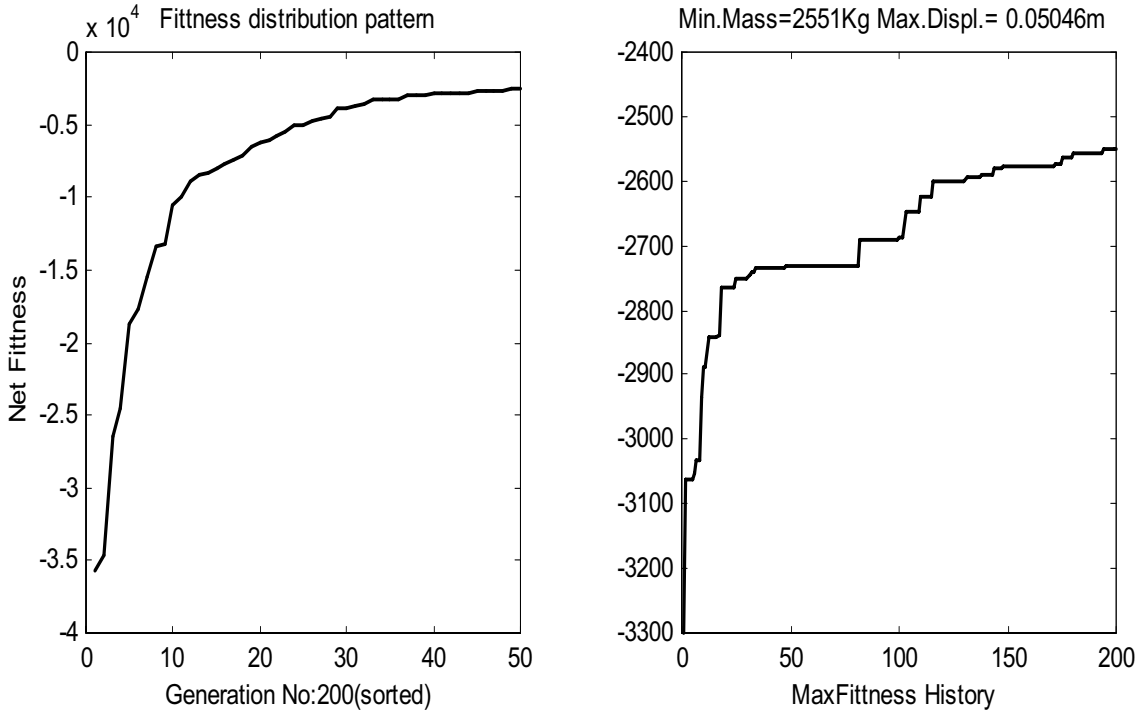
(a) (b)

Figure 8 . a) Sample fitness distributaion, b) Maximum Fitness history and corresponding total mass and displacement response applying AIM in DIC after 500 generations.



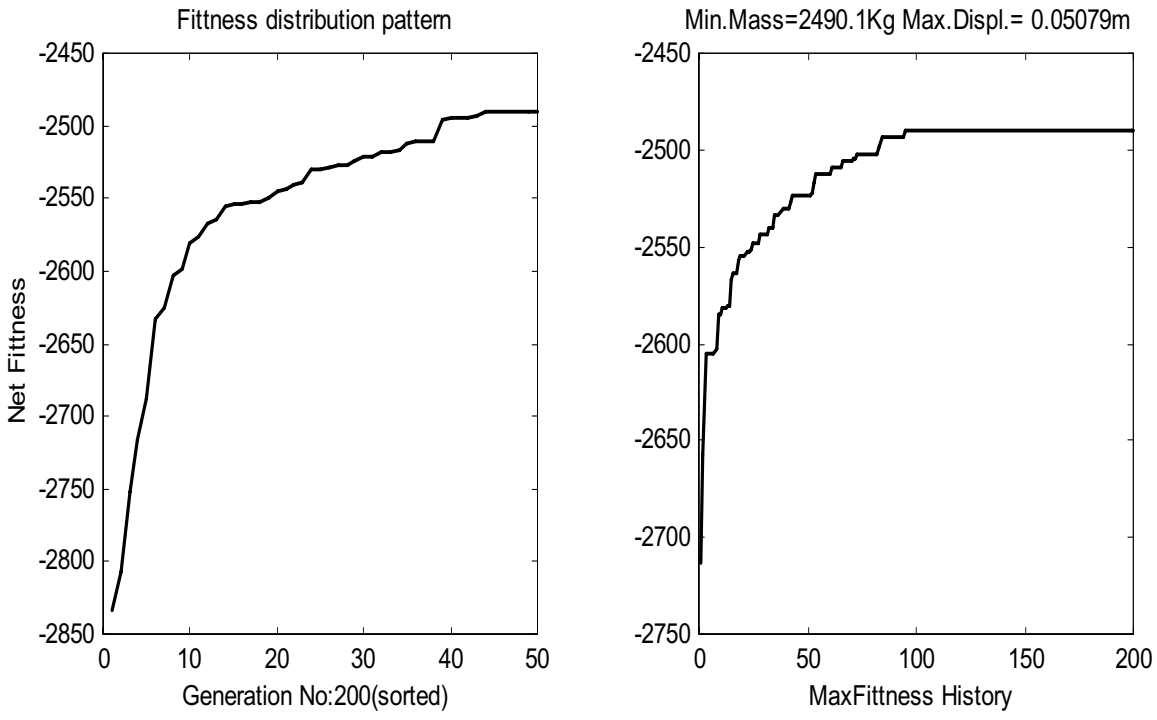
(a) (b)

Figure 9 . a) Sample fitness distributaion, b) Maximum Fitness history and corresponding total mass and displacement response applying BEM in DIC after 200 generations.



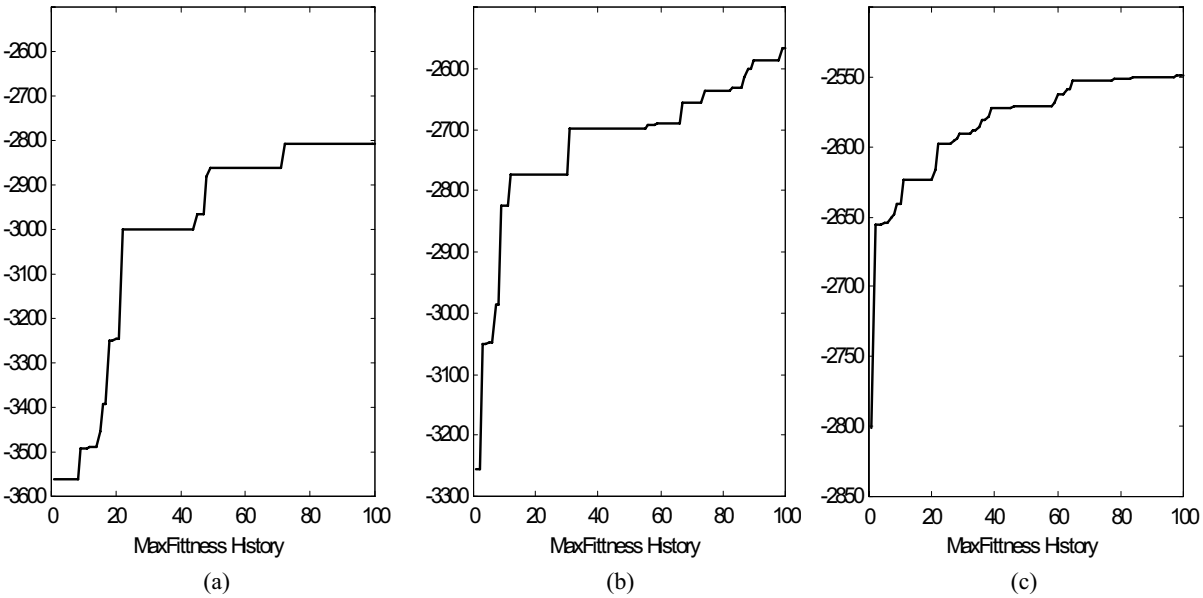
(a) (b)

Figure 10. a) Sample fitness distributaion, b) Maximum Fitness history and corresponding total mass and displacement response applying DIM in DIC after 200 generations.



(a) (b)

Figure 11. a) Sample fitness distributaion, b) Maximum Fitness history and corresponding total mass and displacement response applying AIM in DIC after 200 generations.



(a) (b) (c)

Figure 12. Comparison between sample results of a) DIC+BEM, b)DIC+DIM and c) DIC+AIM for the fitness improvement lags in maximum fitness history and corresponding responses obtained with maximum 100 generations.

is denoted by Δ_j and its upper bound by Δ_a. The problem is solved for the three mentioned types of mutation, i.e., BEM, DIM , AM. In all the cases, random allele initiation and tournament selection are applied. Figures 6 to 8 demonstrate the obtained results for maximum generation numbers of 500, while it is taken as 200 in Figures 9 to 11. The selected GA parameters in each case together with the typical results, are compared to the results of the previous works from literature (Tables 2 and 3).

5. DISCUSSION AND CONCLUSIONS

As reported in Tables 2 and 3, the genetic search combined with direct index coding (DIC) has led to comparable results to those of the binary and real coded methods, even for a considerably smaller number of generations. Though a fewer number of generations (200) have led to inferior results as expected specially for binary expanded (BEM) and Direct Index Mutation (DIM), however, the results of Adjacent Index Mutation (AIM) still stands better than all the others.

It should be noticed from Fig. 3 that a nominal uniformity percent in binary expanded chromosome may really correspond to higher uniformity percentages in DIM or AIM applied to the corresponding contracted index. The results of BEM in Table 2 are not as good as the other mutation methods combined with DIC.

Considering the fact that increasing the amount of mutation probability may exploit and slowthe GA convergence rate, more runs have been conducted for all 3 mutation schemes, in most of which DIC+AIM showed better computational stability (Fig. 12, Table 4). The fitness pattern and histories in Figs 6-8 and also in Figs 9-11, show that generally DIC+AIM has resulted in shorter fitness improvement lags during generations of the GA search. This confirms the efficiency and suitability of the proposed method for the treated structural sizing problems (Table 3).

It can be concluded that the DIC coding only deals with indices associated to structural section properties, and no undesired priority (such as sectional area sequence) will necessarily enter in the GA, preserving its stochastic search characteristic. As a second advantage, a structural design section with various types of properties can be addressed only with one index. This not only minimizes the chromosome length and resulted search space cardinality but also enforces direct application of the proposed encoding method to frame sizing problems. Minimizing the discrete search space by direct index coding, also leads to a higher convergence rate for the utilized genetic algorithm.

REFERENCES

1. Hajela P. Stochastic Search in Structural Optimization: Genetic Algorithms and Simulated Annealing, Chapter 22, 1992, pp. 611-635.

2. Goldberg DE. and Samtani MP. Engineering Optimization via Genetic Algorithm, Proceedings of the 9tht Conference on Electronic Computation, ASCE, New York, NY, 1986, pp. 471-484.

3. Caruana RA. and Schaffer JD. Representation and hidden bias: Gray vs. binary coding for genetic algorithms, in Proceedings of the 5th International Conference on Machine

Learning (editor: John Laird), 1988, pp. 153-161.

4. Eshelman LJ. Bit-climbers and naïve evolution, GA-Digest, 5(39), December 1991.

5. Furuya H. and Haftka RT. Genetic Algorithms for Placing Actuators on Space Structures, Proceedings of the 5th International Conference on Genetic Algorithms, Urbana, IL., July 17-22, 1993, pp. 536-542.

6. Wright AH. Genetic Algorithms for Real Parameter Optimization, Foundations of Genetic Algorithms, Morgan Kaufman, 1991.

7. Turkkan N. Discrete Optimization of Structures Using a Floating-Point Genetic Algorithm., Annual Conference of the Canadian Society for Civil Engineering, Moncton, N.B., Canada, June 4-7, 2003.

8. Feng G., Qiao H. and Dejun W. The Effect on Structural Genetic Design of Binary Coding Floating Coding and Non-Uniform Mutation, Chinese Journal of Applied Mechanics, 3, 1998.

9. Salomon R. The influence of different coding schemes on the computational complexity of genetic algorithms in function optimization, in the 4th International Conference on Parallel Problem Solving from Nature, HM. Voigt, W. Ebeling, I. Rechenberg, and HP. Schwefel, Eds. Berlin: Springer-Verlag, 1996, pp. 227–235.

10. Tang W. Tong L. and Gu Y. Improved genetic algorithm for design optimization of truss structures with sizing, shape and topology variables", International Journal of Numerical Methods in Engineering, No. 13, 62; 2005: 1737-1762.

11. Holland JH. Adaptation in Natural and Artificial Systems. University of Michigan (1975) and MIT Press: Cambridge, MA, 1992.

12. Gen M. and Cheng R. Genetic Algorithms and Engineering Optimization, John Wiley, New York, 2000.

13. Rothlauf F. Binary Representation of Integers and the Performance of Selectorecombinative Genetic Algorithms, Department of Information Systems, University of Bayreuth, Germany, 2002.

14. Chakraborty UK. and Janikow CZ. An analysis of Gray versus binary encoding in genetic search, Information Sciences: An International Journal, Nos 3-4, Special issue: Evolutionary Computation, 156; 2003: 253–269.

15 Rajeev S. and Krishnamoorthy CS. Genetic algorithms-based methodologies for design optimization of trusses. Journal of Structural Engineering, ASCE, 123(3); 1997: 350 –358.

16. Kaveh A. and Kalatjari V. Genetic algorithm for discrete-sizing optimal design of trusses using the force method, International Journal of Numerical Methods in Engineering, 55; 2002: 55–72.

17. Adeli H. and Cheng NT. Concurrent

Genetic Algorithms for Structural Optimization, J. Aero. Engg., 7(3); 1994: 276-296.

18. Garcia S. Experimental Design Optimization and Thermophysical Parameter Estimation of Composite Materials Using Genetic Algorithms, Ph.D. Thesis, Virginia Polytechnic Institute and State University, Nantes, 1999.

19. Rajeev S. and Krishnamoorthy CS. Discrete optimization of trusses using genetic structures, Journal of Structural Engineering (ASCE), 118; 1992: 1233-1350.

20. Cai J. and Thierauf G. Discrete Structural Optimization Using Evolution Strategies, Technical Report, Department of Civil Engineering, University of Essen, Essen, Germany, 1993.

21. Coello CA. Discrete Optimization of Trusses Using Genetic Algorithms, EXPERSYS-94, I.I.T.T., 1994, pp.331-336.

22. Galante M. Genetic Algorithms as an Approach to Optimize Real-World Trusses, International Journal of Numerical Methods in Engineering, 1996; 39: 361-382.

23. Ghasemi MR. Hinton E. and Wood RD. Optimization of Trusses Using Genetic Algorithms for Discrete and Continuous variables", Engineering Computations, 1999;3: 272-301.