A Genetic Algorithm Encoding for a Class of Cardinality Constraints

Helio J.C. Barbosa*
Laboratório Nacional de Computação Científica
Av. Getúlio Vargas 333
25651 075 Petrópolis, RJ, Brazil
hcbm@lncc.br

Afonso C.C. Lemonge
Universidade Federal de Juiz de Fora
Departamento de Estruturas
Cidade Universitária
36036 330 Juiz de Fora, MG, Brazil
lemonge@numec.ufjf.br

ABSTRACT

A genetic algorithm encoding is proposed which is able to automatically satisfy a class of important cardinality constraints where the set of distinct values of the design variables must be a subset –of cardinality not exceeding a given value– of a larger set of available items. The solution of the practically important structural optimization problem where the set of distinct values of the design variables must be a small subset of a larger set of commercially available values is used as a test-bed. Very good results have been found in the numerical experiments performed using standard binary encoding and genetic operators.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: [Heuristic methods]; J.2 [**Physical Sciences and Engineering**]: Engineering

General Terms

Algorithms

Keywords

cardinality constraint, genetic algorithm, structural optimization

1. INTRODUCTION

The application of genetic algorithms (GAs) to constrained optimization problems (COPs) gives rise in general to several difficulties: (i) the objective function may be undefined for some or all unfeasible elements, (ii) the check for feasibility can be more expensive than the computation of the objective function itself, and (iii) an informative measure of the degree of unfeasibility of a given candidate solution is not easily defined. It is easy to see that even if both the objective function $f(x)$ and a measure of constraint violation $v(x)$ are defined for all feasible x it is not possible to know in general which of two given infeasible solutions is closer to the optimum and thus should be operated upon or kept in the population. For instance, in a minimization problem, one can have $f(x_1) > f(x_2)$ and $v(x_1) = v(x_2)$ or a situation where $f(x_1) = f(x_2)$ and $v(x_1) > v(x_2)$ and still have x_1 closer to the optimum. As a result, several techniques have been proposed in the literature in order to enable a GA to tackle COPs which can be classified[7] either as *direct* (feasible or interior), when only feasible elements are considered, or as *indirect* (exterior), when both feasible and unfeasible elements are used during the search process.

Direct techniques comprise: a) the design of special *closed* genetic operators[11], b) the use of special decoders[6], c) repair techniques[8, 9], and d) the "death penalty", when unfeasible elements are just discarded. However, direct techniques require domain knowledge (with the exception of the "death penalty") and are actually of extremely reduced practical applicability, specially for handling implicit constraints. As a result, it seems always worthwhile to avoid constraints by carefully designing representations for the candidate solutions such that *all* elements in the search space satisfy one or more of the problem constraints.

One type of constraint which appears in real-world applications is the so-called *cardinality* constraint. An important example arises in portfolio optimization where a portfolio manager is faced with the problem of selecting from a usually large set of assets offered in the market (stocks, bonds, options, etc.) a subset of assets for investment following a given objective concerning performance and risk while respecting certain constraints from budget, legal regulations and other guidelines[3]. To reduce the complexity of portfolio control and also to control transaction costs, it is useful to introduce an upper bound to the *number* of assets that will be included in the portfolio. This simple constraint introduces zero-one variables which increase the size and complexity of the original problem. Meta-heuristics have been applied in this case by, for instance, Chang et al. [2].

In this paper, a more complex cardinality constraint is considered and a special encoding scheme is proposed to deal with it. In words, the class of problems considered consists in assigning one object (taken from a large set of available objects) to each item in a set of items in order

*Corresponding author

GECCO'05, June 25–29, 2005, Washington, DC, USA.

to minimize a given objective function with the constraint that the resulting assignment of objects contains up to a given number of different objects. A practical structural optimization problem –detailed in section 3– will then be used here not only to provide a concrete example of the cardinality constraint considered but also as an initial test bed for the proposed encoding.

Section 4 describes the encoding scheme designed to deal with the cardinality constraint. Numerical experiments are described in Section 5 and the paper ends with a Conclusions section.

2. THE CARDINALITY CONSTRAINT

In several optimization problems, the design variables can be continuous and/or discrete and the inclusion of the later usually makes the problem harder. The techniques from mathematical programming which are used in the continuous case must be augmented with procedures to deal with the discrete variables leading to a computational code with increased complexity.

In practice, it is often desirable (or even mandatory) to choose design variables from commercially available sizes or types. The use of a continuous optimization procedure – although usually more straightforward– will lead to non-available sizes and any attempt to "round" or substitute those values by the "closest" available commercial sizes can potentially make the design unfeasible (constraints are violated) or with unnecessarily degraded performance.

Even when the parts are to be fabricated and thus could be of the desired sizes, the solution obtained by a typical optimization code will usually prescribe as many sizes as the number of design variables defined for the problem. A common procedure, often referred to as variable *linking*, groups different design variables of the problem in a single design variable. That is useful when symmetry conditions are to be enforced, for example. This procedure allows for a reduction in the total number of design variables and, usually, in the complexity of the search problem.

It should be noted that each choice of design variable linking leads to a different optimization problem with a potentially substantially different optimal solution. However, the choice of which variables should be grouped together is not trivial, but must be made a priori by the analyst. As a result, the final set of independent design variables may be far from optimal and the corresponding optimal solution more expensive than necessary due to inadequate design variable linking.

A better solution for the problem above would be to provide the designer with the possibility of directly controlling only the *number* of different sizes or types to be used in a given problem and let the GA search also for the grouping of the design variables. The introduction of such cardinality constraint would lead to

1. economies and simplification in procurement, storage, and assembling which are more difficult to quantify and thus not usually included in the optimization procedure, and
2. it would also alleviate the designer's task of choosing which variables to link in a group.

To the best of the authors' knowledge, the mixed discrete-continuous optimization problem with the additional *cardinality* constraint on the set of the *final* distinct values of the design variables has not been solved in the literature.

Within the context of weight minimization of structures, Shea et al.[10] try to reduce the number m of groups formed by adding to the weight of the structure, W, a penalty term which grows with m. A procedure to form/re-arrange groups is introduced (with the additional control parameters to be set by the user) but due to the combination of W and m in the objective function, the user has not much control over m.

In this paper, a GA encoding is proposed where the user is able to prescribe the maximum number m of different sizes or types he or she is willing to use in a particular design in a way that both goals (1) and (2) above are attained.

For the discrete variable case, the GA simultaneously searches, among the commercially available sizes, for the optimal subset (of cardinality not exceeding m) of properties to be used, and the corresponding assignment to the design variables. For the continuous variable case, the GA simultaneously searches for a set (of cardinality not exceeding m) of properties in a given range and the corresponding assignment to the design variables.

In the following the optimization problem considered is detailed.

3. THE STRUCTURAL OPTIMIZATION PROBLEM

A standard constrained optimization problem in R^n can be thought of as the minimization of a given objective function $f(x)$, where $x \in R^n$ is the vector of design/decision variables, subject to inequality constraints $g_p(x) \geq 0$, $p = 1, 2, \ldots, \bar{p}$ as well as equality constraints $h_q(x) = 0$, $q = 1, 2, \ldots, \bar{q}$. Additionally, the variables may be subject to bounds $x_i^L \leq x_i \leq x_i^U$ but this type of constraint is trivially enforced in a GA and does not require further consideration.

The constrained optimization problems considered here include also discrete variables, that is, some components of $x \in R^n$ are further constrained to assume values belonging to a given discrete set.

A common structural design problem is the weight minimization of structures subject to stress, displacement, and other constraints. To fix ideas, truss structures can be considered and the problem reads: Find the set of areas

$$x = \{A_1, A_2, \ldots, A_N\}$$

which minimizes the volume of the structure

$$f(x) = \sum_{i=1}^{N} A_i l_i, \tag{1}$$

where l_i is the length of the i-th member of the truss and N is the number of bars. The most common constraints are stress constraints:

$$\frac{|\sigma_i|}{\sigma_{max}} - 1 \leq 0, \quad i = 1, 2, \ldots, p_\sigma \tag{2}$$

where σ_i is the stress at the i-th member and σ_{max} is the maximum allowable stress. Displacements constraints can also be considered:

$$\frac{|d_j|}{d_{max}} - 1 \leq 0, \quad j = 1, 2, \ldots, p_d \tag{3}$$

where d_j is the displacement at the j-th global degree of freedom, d_{max} is the maximum allowable displacement, and

$p_\sigma + p_d = \bar{p}$. Additional constraints such as buckling or a minimum natural vibration frequency can also be included.

The additional *cardinality* constraint of interest here requires that no more than m different areas should be used, that is,

$$A_i \in C_m = \{S_1, S_2, \ldots, S_m\}, \quad i = 1, 2, \ldots, N \qquad (4)$$

where the areas $S_j, j = 1, 2, \ldots, m$ are unknown but:

- belong to a larger ($M > m$) given set $S = \{A_1, A_2, \ldots, A_M\}$ for the discrete case, or
- are within a prescribed range $R = [A_{min}, A_{max}]$ for the continuous variable case.

It is clear that the standard optimization problem with no cardinality constraints is recovered when $m = N$.

In the following the genetic encoding proposed for handling the cardinality constraint is presented.

4. THE GENETIC ALGORITHM

The essential component of the GA used to handle the cardinality constraint is the encoding scheme detailed in the following.

4.1 The chromosome structure

For explicative purposes, assume that one has a set of 32 available commercial sizes from which a *maximum* of 4 types are to be used in a given design problem. Assume further that the problem has 10 design variables and, to simplify the presentation, that no variable links have been introduced, that is, each design variable is actually the cross sectional area of a given bar in the truss structure considered.

The chromosome proposed here is a string of variables in the form:

`type(1) type(2) type(3) type(4) pt(1) pt(2) ... pt(10)`

where `type(i)`$\in \{1, 2, \ldots, 32\}, i = 1, \ldots, 4$ are pointers to the (at most) 4 different sizes allowed to be chosen from the table of 32 available sizes

`table(1), table(2), ..., table(32)`

and

`pt(1), pt(2), ..., pt(10),`

with `pt(i)`$\in \{1, 2, 3, 4\}$, are pointers to one of the 4 types listed in the beginning of the chromosome for each of the 10 design variables/cross-sectional areas. As a result, the area of the `i`-th bar in the structure is given by

$$\texttt{area(i)} = \texttt{table(type(pt(i)))}.$$

The chromosome illustrated in the Figure 1 has the first four variables (`type(i), i=1,4`), pointed by the other variables stored in positions 5 to 14. The variables with values 23, 4, 12, and 31, are pointers to the `table` containing 32 different values of cross-sections as detailed in the left side of the Figure 1. For example, the sixth position, corresponding to the second design variable, stores the value 1 indicating `type(1)` which stores the value 23 which, in turn, points to `table(23)`=4.80. In this way one has

$$A_2 = \texttt{table(type(1))} = \texttt{table(23)} = 4.80.$$

Also, the sixth design variable A_6, coded in the tenth position, points to `type(3)`, which is equal to 12, resulting,

$$A_6 = \texttt{table(type(3)))} = \texttt{table(12)} = 3.80,$$

and so on.

For the continuous case one specifies the maximum number of different sizes allowed, for instance $m = 4$, and the chromosome structure proposed here is as follows:

`size(1) size(2) size(3) size(4) pt(1) pt(2) ... pt(10),`

with `size(i)`$\in [A^i_{min}, A^i_{max}]$, where again 10 design variables (cross-sectional areas) are assumed. Analogously to the discrete case, the area of the `i`-th bar in the structure is given by

$$\texttt{area(i)} = \texttt{size(pt(i))}.$$

Now the variable `size(i)` is continuous and has lower (A^i_{min}) and upper (A^i_{max}) bounds specified by the designer, and the variable `pt` is again a pointer to one of the $m = 4$ sizes listed in the beginning of the chromosome. The chromosome exemplified in Figure 2 corresponds to the following values of the design variables: $A_1 = A_7 = 1.08$, $A_2 = A_6 = A_8 = 2.16$, $A_3 = A_{10} = 0.6$, and $A_4 = A_5 = A_9 = 1.44$.

In the actual implementation in this paper all variables are binary encoded in the chromosome. For the example shown in Figure 1 the length of the chromosome is $c_l = 4 \times 5 + 10 \times 2 = 40$ bits.

Finally, it is worth mentioning that *less* than m distinct values can be used for the design variables in any candidate solution. This happens whenever:

- one of the variables, say `type(k)` (in the discrete case) or `size(k)` (in the continuous case), is not assigned to a design variable, that is, `pt(i)`$\neq k, \forall i$, or
- there are repeated values among the variables `type(k)` or `size(k)`.

The first case is exemplified in Figure 3 (for continuous variables).

4.2 Handling the other constraints

In order to take the remaining implicit constraints (2) and (3) into consideration in the GA, several penalty procedures can be found in the literature [1]. Among them, a parameter-less adaptive penalty scheme introduced by the authors in [1] has been chosen due to its simplicity and good performance [7].

Defining [1] the amount of violation of the j-th constraint by the candidate solution x as

$$v_j(x) = \begin{cases} |h_j(x)|, & \text{for an equality constraint,} \\ \max\{0, -g_j(x)\} & \text{otherwise} \end{cases}$$

the fitness function is given by

$$F(x) = \begin{cases} f(x), & \text{if } x \text{ is feasible,} \\ \mathtt{h}(x) + \sum_{j=1}^{\mathcal{M}} k_j v_j(x) & \text{otherwise} \end{cases} \qquad (5)$$

where $\mathcal{M}$ is the number of constraints to be penalized, and $\mathtt{h}(x)$ is defined as

$$\mathtt{h}(x) = \begin{cases} f(x), & \text{if } f(x) > \langle f(x) \rangle, \\ \langle f(x) \rangle & \text{otherwise} \end{cases} \qquad (6)$$

with $\langle f(x) \rangle$ denoting the average of the objective function values in the current population.

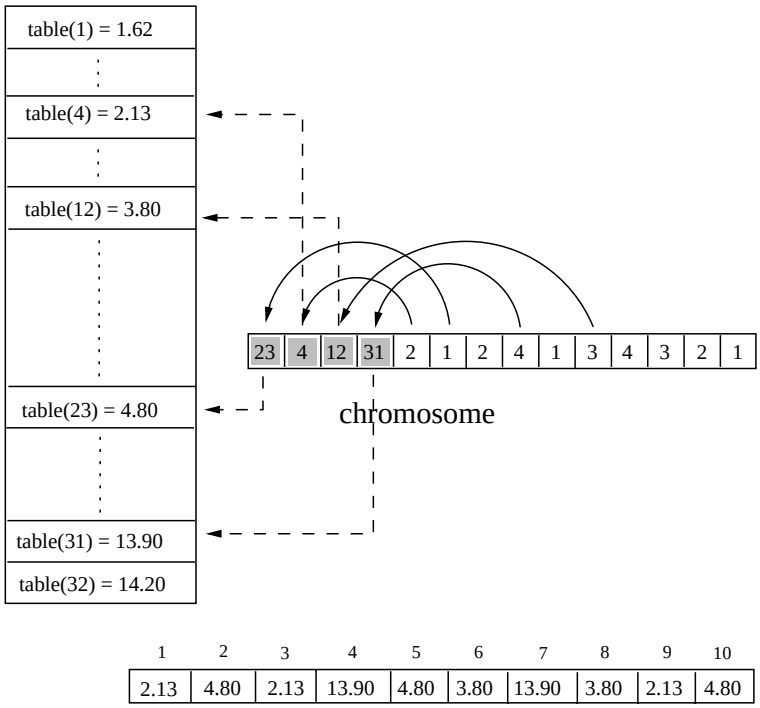


Figure 1: Chromosome for the discrete case (above) and the corresponding values of the cross-sectional areas in each bar (below).

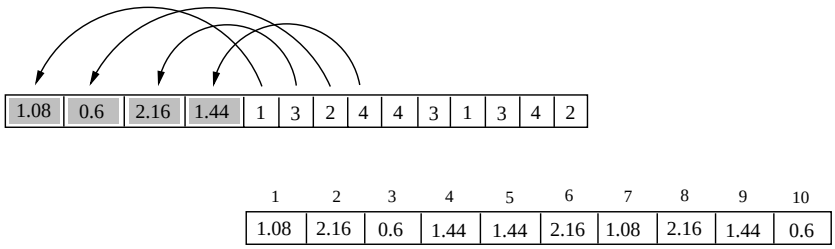


Figure 2: Example of chromosome for the continuous variable case. The third design variable has its area defined by the third pointer value (2) which indicates the second continuous area value (0.6). For convenience, the values are shown in decimal representation; as explained in the text, a binary representation is used in the GA.

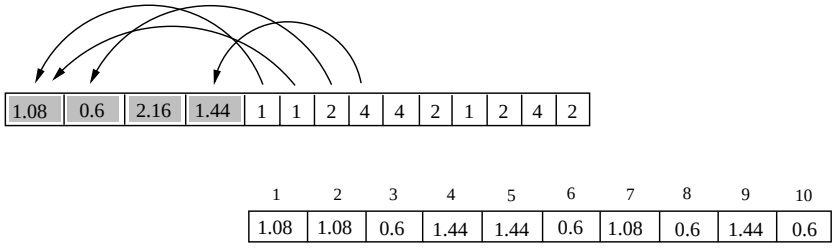


Figure 3: Example of chromosome where *less* than m variables are used in the continuous variable case. Only 3 areas are actually used. No design variable has been assigned to the *third* value (2.16) since there is no pointer equal to 3 stored in positions 5 to 14.

In the Figure 4, feasible as well as unfeasible solutions are shown for a minimization problem. Among the 6 unfeasible solutions, the individuals number 3, 4, 5, and 6 have their objective function values (represented by open circles), smaller than the average objective function and, accordingly, have $\overline{f}$ given by $\langle f(x)\rangle$. The solutions number 1 and 2 have objective function values which are worst than the population average and thus have $\overline{f}(x) = f(x)$.

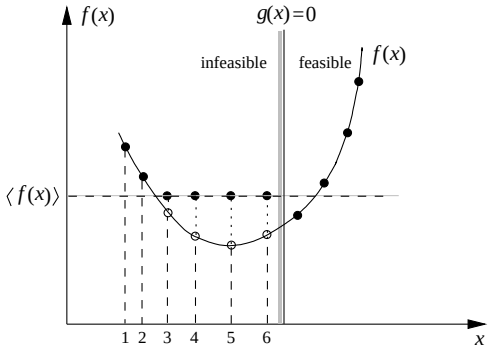


Figure 4: The definition of the function $\overline{f}$.

The penalty parameter is defined at each *generation* by:

$$k_j = |\langle f(x)\rangle| \frac{\langle v_j(x)\rangle}{\sum_{l=1}^{\mathcal{M}} [\langle v_l(x)\rangle]^2} \qquad (7)$$

and $\langle v_l(x)\rangle$ is the violation of the l-th constraint averaged over the current population. The idea is that the penalty coefficients should be distributed in such a way that those constraints which are more difficult to be satisfied should have a relatively higher penalty coefficient.

For the structural optimization problems considered in this paper the objective function is defined by (1). When the j-th constraint is a stress constraint in a given bar or a nodal displacement constraint one has, respectively,

$$v_j = \left[\frac{|\sigma_j|}{\sigma_{max}} - 1\right]^+ \qquad \text{or} \qquad v_j = \left[\frac{|d_j|}{d_{max}} - 1\right]^+ \qquad (8)$$

where $[\alpha]^+ = \alpha$, if $\alpha \in R$ is positive, and zero otherwise. The fitness function is finally defined from equations (1), (8), (5), and (6).

In the next section the proposed GA is applied to some structural optimization problems.

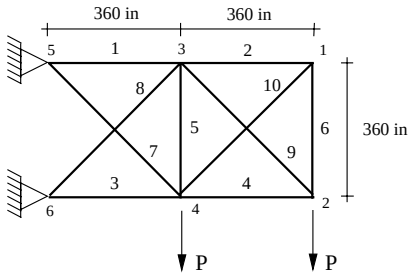
5. NUMERICAL EXPERIMENTS

In the GA used for the numerical experiments, a linear rank selection scheme is adopted with elitism: the best element is always copied into the next generation along with one copy where one bit has been mutated. In all cases the one-point, two-point, and uniform crossover operators were applied with probabilities $p_c^1 = 0.16$, $p_c^2 = 0.32$, and $p_c^u = 0.32$, respectively, followed by the standard mutation operator applied with rate $p_m = 0.03$. These are standard settings previously used by the authors in structural optimization problems and no tuning of the GA parameters was attempted here.

5.1 The first experiment

The well known test problem studied by Goldberg and Samtani[5] which corresponds to the weight minimization of the ten-bar truss shown in the Figure 5 will be analyzed here with a cardinality constraint in addition to the standard constraints involving the stress in each member and the displacements at the nodes. The design variables are the cross-sectional areas of the bars (A_i, $i = 1, 10$). The allowable stress is limited to $\pm$ 25ksi and the displacements are limited to 2 in, in the x and y directions. The density of the material is 0.1 lb/in^3, Young's modulus is E = 10^4 ksi and vertical downward loads of 100 kips are applied at nodes 2 and 4.

Figure 5: Test-problem 1 – The ten-bar truss.



Two cases are analyzed: discrete and continuous variables. For the discrete case the values of the cross-sectional areas (in^2) are chosen from the set $\mathcal{S}$ with 42 options: 1.62, 1.80, 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, 11.50, 13.50, 13.90, 14.20, 15.50, 16.00, 16.90, 18.80, 19.90, 22.00, 22.90, 26.50, 30.00, 33.50 . Using a six-bit string for each design variable, as in [4], a total of 64 strings are available. In this way, the first 22 values of the set $\mathcal{S}$ (from 1.62 to 4.59) are listed twice (1.62, 1.62, 1.80, 1.80, etc.). For the continuous case the minimum cross sectional area is equal to 0.1 in^2.

The Table 1 presents the results found using the Adaptive Penalty Method (APM)[1] for the discrete and continuous cases without any kind of cardinality constraints, and also setting $m = 2$ and $m = 4$. In this Table, the values of the vertical displacements at the nodes 1 and 2 of the truss are displayed since these are the hardest constraints observed for this problem. One can observe from the Table 1 that the values found for the discrete cases are greater than those of the corresponding continuous case as expected. Where no cardinality constraints are imposed the values presented in this experiment can be compared with those available in the extensive discussion found in [7]. For the case $m = 2$ the GA found the sets {2.62, 22.00}, and {0.1, 22.73555} for the discrete and continuous cases, respectively, and one can note from the Table 1 a clear coherence between these assigned values and their respective design variables. Using $m = 4$ the GA found the sets {1.62, 14.20, 22.00, 30.00}, and {0.1, 9.45096, 20.24207, 31.98362} for the discrete and continuous cases, respectively. Again, as expected, the GA found a better value for the continuous case.

The Table 2 summarizes the performance of the GA for the 10-bar truss in 20 independent runs.

Table 1: Comparison of results for the 10-bar truss. Subscripts d and c denote the discrete and continuous cases, respectively. Final weight in lb, and u_{y1} and u_{y2} are the displacements at the nodes 1 and 2, respectively.

Design Variables	APM_d	APM_c	$\text{APM}_d^{m=2}$	$\text{APM}_c^{m=2}$	$\text{APM}_d^{m=4}$	$\text{APM}_c^{m=4}$
A_1	33.50	31.27305	22.00	22.73555	30.00	31.98362
A_2	1.62	0.10715	2.62	0.10000	1.62	0.10000
A_3	22.90	23.23471	22.00	22.73555	22.00	20.24207
A_4	14.20	15.97336	22.00	22.73555	14.20	20.24207
A_5	1.62	0.11142	2.62	0.10000	1.62	0.10000
A_6	1.62	0.39052	2.62	0.10000	1.62	0.10000
A_7	7.97	7.66543	22.00	22.73555	14.20	9.45096
A_8	22.90	22.68053	22.00	22.73555	22.00	20.24207
A_9	22.00	19.20086	22.00	22.73555	22.00	20.24207
A_{10}	1.62	0.10000	2.62	0.10000	1.62	0.10000
Weight	5490.738	5086.851	6152.520	5947.479	5603.697	5167.010
u_{y1}	−1.9591	−1.9999	−1.8868	−1.8757	−1.9013	−1.9978
u_{y2}	−1.9889	−1.9908	−1.9956	−1.9999	−1.9933	−1.9999

Table 2: Performance of the GA for the 10-bar truss.

discrete	$m=2$	$m=4$	no c.c
best	6152.520	5603.697	5490.738
average	6152.520	5725.113	5551.846
worst	6152.520	5812.963	5631.638
pop. size	200	200	200
max. eval.	240000	360000	360000
continuous	$m=2$	$m=4$	no c.c
best	5943.847	5167.010	5086.851
average	5944.508	5262.947	5129.474
worst	5947.479	5402.350	5259.076
pop. size	200	200	200
max. eval.	280000	360000	400000

5.2 The second experiment

In this experiment, a truss with 25 bars shown in the Figure 6 is submitted to weight minimization. The constraints require that the maximum stresses in the members remain in the interval $[-40, 40]$ ksi and that the maximum displacements at the nodes 1 and 2 be limited to 0.35 in, in both the x and y directions. The design variables are cross-sectional areas of the bars to be chosen from the set with 30 different options (in square inches): 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.2, 3.4. In order to use a five-bit string for each design variable the first and the last values (0.1 and 3.4) are listed twice in this set. The member areas are grouped in eight design variables as detailed in the Table 4. The material has density equal to 0.1 lb/in^3 and the Young's modulus is equal to 10^4 ksi. The loading data is listed in the Table 3.

Table 3: Loading data for the 25-bar truss (kips).

node	F_x	F_y	F_z
1	1	−10.0	−10.0
2	0	−10.0	−10.0
3	0.5	0	0
6	0.6	0	0

The Table 5 displays the results found in the discrete and continuous cases using the APM technique with no cardinality constraints as well as setting $m = 2$, $m = 4$ and $m = 8$. The values of the vertical displacements u_{y1} and u_{y2}, at the nodes 1 and 2 respectively, are also shown since these are the hardest constraints observed for this problem. In the absence of cardinality constraints, several solutions found in the literature are discussed in [7].

For the case where $m = 2$ the values found were {3.4, 0.8} and they are assigned as shown in the fourth column

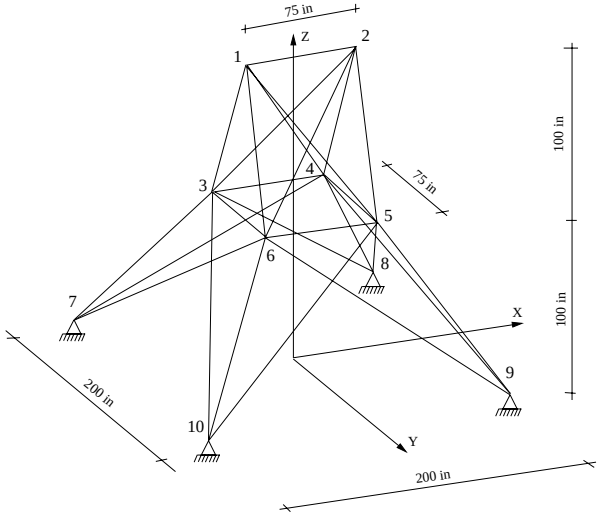


Figure 6: The 25-bar truss.

Table 4: Variable linking for the 25-bar truss.

group	bar connectivities
A_1	1-2
A_2	1-4, 2-3, 1-5, 2-6
A_3	2-5, 2-4, 1-3, 1-6
A_4	3-6, 4-5
A_5	3-4, 5-6
A_6	3-10, 6-7, 4-9, 5-8
A_7	3-8, 4-7, 6-9, 5-10
A_8	3-7, 4-8, 5-9, 6-10

Table 5: Comparison of results for the 25-bar truss. Subscripts d and c denote the discrete and continuous cases, respectively. Final weight in lb, and u_{y1} and u_{y2} are the displacements at the nodes 1 and 2, respectively.

Design Variables	APM_d	APM_c	$\text{APM}_d^{m=2}$	$\text{APM}_c^{m=2}$	$\text{APM}_d^{m=4}$	$\text{APM}_c^{m=4}$	$\text{APM}_d^{m=8}$	$\text{APM}_c^{m=8}$
A_1	0.1	0.10901	0.8	0.78568	0.1	0.10022	0.1	0.10015
A_2	0.3	0.47443	0.8	0.78568	0.1	0.10022	0.3	0.41636
A_3	3.4	3.39863	3.4	3.39846	3.4	3.38392	3.4	3.39566
A_4	0.1	0.10000	0.8	0.78568	0.1	0.10022	0.1	0.10015
A_5	2.1	1.73260	0.8	0.78568	1.6	1.50448	2.1	2.20206
A_6	1.0	1.01379	0.8	0.78568	0.9	0.92025	1.0	0.98439
A_7	0.5	0.43589	0.8	0.78568	0.9	0.92025	0.5	0.41636
A_8	3.4	3.39859	3.4	3.39846	3.4	3.38392	3.4	3.39566
Weight	484.854	484.736	514.451	510.943	488.651	488.625	484.854	484.855
u_{y1}	−0.3498	−0.3499	−0.3479	−0.3499	−0.3494	−0.3499	−0.3481	−0.3499
u_{y2}	−0.3478	−0.3479	−0.3468	−0.3478	−0.3483	−0.3489	−0.3477	−0.3477

of the Table 5. For the continuous case (fifth column) the values found were {0.78568, 3.39846} and one can note the coherence between the continuous and the discrete cases.

For $m = 4$ the values found were {0.1, 0.9, 1.6, 3.4} and they are assigned as shown in the sixth column of the Table 5. For the continuous case the GA found the set {0.10022, 0.92025, 1.50448, 3.38392} and, again, a complete coherence between the discrete and the continuous case with a better value for the last one (488.625 against 488.651).

For $m = 8$, the values found for the discrete case were {0.1, 0.3, 0.5, 1.0, 2.1, 3.4} where only 6 different areas are chosen. They correspond exactly to the solution found for the case with no cardinality constraint shown in the second column of the Table 1. Since the bars of the truss are linked in eight groups, the optimum should be equal to the case where $m = 8$. Finally, for the continuous case the values found by the GA were {0.10015, 0.41636, 0.98439, 2.20206, 3.39566} and, again, only 6 values were assigned and the best value achieved for the weight of the truss is practically equal to the weight of the corresponding discrete case.

Table 6: Performance of the GA for the structural optimization of the 25-bar truss.

discrete	$m = 2$	$m = 4$	$m = 8$	no c.c
best	514.451	488.651	484.854	484.854
average	514.451	493.269	488.715	486.177
worst	514.451	499.447	496.358	488.320
pop. size	70	200	200	200
max. eval.	35000	160000	160000	160000
continuous	$m = 2$	$m = 4$	$m = 8$	no c.c
best	510.943	488.625	484.855	484.736
average	511.187	492.869	491.397	487.665
worst	512.090	498.065	495.944	493.039
pop. size	70	200	200	200
max. eval.	35000	240000	400000	240000

It is important to note in this example that a great reduction (from 8 to 4) in the number of different types of bar can be achieved –with the corresponding economies not usually included in the optimization process– with a very small impact in the final weight of the structure: less than 1 %.

5.3 An additional experiment

In order to provide some comparative results, the first experiment is revisited using a standard penalty scheme without any kind of special encoding as the one proposed in this work. The idea is to define for the constraint corresponding to the maximum number of distinct cross-sectional areas (m) of the truss a violation term analogous to those given in (8) for the standard structural optimization constraints (2-3). In this way an additional constraint v_j to be considered is:

$$v_j = \left[\frac{k}{m} - 1\right]^+$$

where k is the number of distinct cross-sectional areas for a given solution. As in the previous experiments, 20 independent runs were performed. The population size was set to 200 and 200000 function evaluations were allowed.

The Table 7 presents the solution found in the best run with the corresponding weight of the structure (objective function) as well as the value of the fitness function $F(x)$. In the previous experiments these values were omitted since they are exactly the same of the objective function, indicating that all solutions found are feasible with respect to (2-3) and the cardinality constraint. The number of distinct cross-sectional areas used in the best solution found is indicated by m^*.

In Table 7 one can notice, for instance, that when $m = 2$ in the continuous case, the best solution found after 200000 evaluations has a weight of 7156.388 with a set of design variables of cardinality 8 {1.44996, 2.62560, 10.47752, 12.36701, 22.95446, 24.34380, 26.54509, 34.05173} which, although feasible with respect to (2-3), has much more than $m = 2$ distinct cross-sectional areas. Feasible solutions were found only for the discrete cases, although for $m = 4$ the solution is worse than that found using the proposed encoding (sixth column of the Table 1).

Comparing Table 1 against Table 7 one observes that even using a smaller number of function evaluations the special encoding proposed in this work reached much better solutions which were also feasible with respect to the whole set of constraints. The performance of the GA for both cases is displayed in Tables 2 and 8.

Summarizing, the experiment illustrates the efficacy (by design) and efficiency of the proposed encoding when compared to penalization of the cardinality constraint by means

of an adaptive penalty scheme (APM), which has been shown to be robust for several constrained (not necessarily structural) optimization problems as discussed in [7].

Table 7: Results found using a standard penalty scheme not considering any special encoding for the 10-bar truss.

Var.	$\text{APM}_d^{m=2}$	$\text{APM}_c^{m=2}$	$\text{APM}_d^{m=4}$	$\text{APM}_c^{m=4}$
A_1	22.00	22.95446	26.50	21.12460
A_2	2.62	22.95446	1.62	15.20247
A_3	22.00	34.05173	26.50	25.17763
A_4	22.00	10.47752	22.90	8.46919
A_5	2.62	1.44996	1.62	19.93876
A_6	2.62	2.62560	1.62	13.57450
A_7	22.00	26.54509	7.22	25.63037
A_8	22.00	10.47752	22.90	13.57450
A_9	22.00	24.34380	22.90	16.90908
A_{10}	2.62	12.36701	1.62	22.65287
Weight	6152.520	7156.388	5689.175	7735.689
$F(x)$	6152.520	1185993	5689.175	815320
m^*	2	8	4	9
u_{y1}	−1.8868	−1.5529	−1.9632	−1.8096
u_{y2}	−1.9656	−1.9482	−1.9826	−1.9425

Table 8: Performance of the GA for the structural optimization of the 10-bar truss using a standard penalty scheme.

	$\text{APM}_d^{m=2}$	$\text{APM}_c^{m=2}$	$\text{APM}_d^{m=4}$	$\text{APM}_c^{m=4}$
best	6152.5	1185993.2	5689.1	815320.1
average	47773.9	477983.6	6225.5	150924.3
worst	152431.9	1421780.1	7334.3	1312701.9
pop. size	200	200	200	200
max. eval	400000	400000	400000	400000

6. CONCLUSIONS

A genetic algorithm encoding has been proposed which is able to automatically satisfy a class of important cardinality constraints where the set of distinct values of the design variables must be a subset –of low cardinality– of a larger set of available items. The practically important structural optimization sizing problem with discrete and/or continuous variables which inspired the research can be dealt with and very good results have been found in the numerical experiments performed using the proposed solution representation which has been encoded in a binary chromosome and operated with standard genetic operators.

It is worth mentioning that the technique can be applied to other similar (not necessarily structural) optimization problems where it would be desirable to enforce a cardinality constraint on the set of distinct values of the (continuous or discrete) design variables.

7. ACKNOWLEDGMENTS

The authors thank CNPq (grant no. 302299/2003-3), MCT/LNCC/PRONEX, and the reviewers for the corrections and suggestions which helped improve the quality of the paper.

8. REFERENCES

[1] H. Barbosa and A. Lemonge. An adaptive penalty scheme in genetic algorithms for constrained optimization problems. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conf.*, pages 287–294, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[2] T.-J. Chang, N. Meade, J. Beasley, and Y. Sharaiha. Heuristics for cardinality constrained portfolio optimisation. *Computers & Operations Research*, 27:1271–1302, 2000.

[3] U. Derigs and N.-H. Nickel. On a local-search heuristic for a class of tracking error minimization problems in portfolio management. *Annals of Operations Research*, 131(1-4):45–77, October 2004.

[4] M. Ghasemi, E. Hinton, and R. Wood. Optimization of trusses using genetic algorithms for discrete and continuous variables. *Engineering Computations*, 16:272–301, 1997.

[5] D. Goldberg and M. Samtani. Engineering optimization via genetic algorithms. In *Proc., 9th Conf. Electronic Computation, ASCE.*, pages 471–482, New York, NY, 1986.

[6] S. Koziel and Z. Michalewicz. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation*, 7(1):19–44, 1999.

[7] A. C. Lemonge and H. J. Barbosa. An adaptive penalty scheme for genetic algorithms in structural optimization. *Int. Journal for Numerical Methods in Engineering*, 59(5):703–736, 2004.

[8] G. Liepins and W. Potter. A Genetic Algorithm Approach to Multiple-Fault Diagnosis. In L. Davis, editor, *Handbook of Genetic Algorithms, chapter 17*, pages 237–250, Van Nostrand Reinhold, New York, NY., 1991.

[9] D. Orvosh and L. Davis. Using a Genetic Algorithm to Optimize Problems with Feasibility Contraints. In *Proc. of the First IEEE Conference on Evolutionary Computation*, pages 548–553, IEEE Press, 1994.

[10] K. Shea, J. Cagan, and S. Fenves. A shape annealing approach to optimal truss design with dynamic grouping of members. *ASME Journal of Mechanical Design*, 119:388–394, 1997.

[11] M. Shoenauer and Z. Michalewicz. Evolutionary computation at the edge of feasibility. In H.-M. V. W. E. I. Rechenberg and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN IV*, volume 1141, pages 245–254, Berlin, 1996. Springer-Verlag. LNCS.