METAMODEL ASSISTED GENETIC ALGORITHM FOR TRUSS WEIGHT MINIMIZATION

L. G. Fonseca, H. J. C. Barbosa
Laboratório Nacional de Computação Científica - LNCC/MCT
Av. Getúlio Vargas, 333, Quitandinha, Petrópolis, RJ, Brazil, CEP 25651-075
{goliatt, hcbm}@lncc.br

A. C. C. Lemonge
Universidade Federal de Juiz de Fora, Faculdade de Engenharia,
Campus Universitário, Bairro Martelos, CEP 36036-330, Juiz de Fora, MG, Brazil
afonso.lemonge@ufjf.edu.br

ABSTRACT

Genetic Algorithms (GAs) are population based stochastic search procedures which have been widely applied in several difficult optimization problems. A common structural optimization problem is the weight minimization of framed structures subjected to stress, displacements, and other constraints. The constraints verification usually involves a simulation (such as the solution of the discretized equilibrium equations from a finite element model), and for real-world problems this simulation can be time-consuming. As a result, the total computing time may be too large, rendering the application of a GA impractical or even prohibitive.

To reduce the use of the expensive simulation model during the search performed by the GA, a metamodel is introduced. Similarity based metamodels for the objective function as well as for the constraint violation are used, and a stochastic ranking is adopted which does not require penalty parameters. The proposed procedure is implemented in a binary-coded generational GA and applied to structural optimization problems involving discrete and continuous design variables. Numerical experiments are performed in order to assess the applicability and the performance of the proposed technique.

The results obtained show that the metamodel allows computational gains by reducing the number of expensive simulations, significantly reducing the total computational time.

1. INTRODUCTION

Genetic Algorithms (GAs) are population based stochastic search procedures which have been widely applied in several difficult optimization problems. They do not require differentiability or continuity of the objective function, they are less sensitive to the initialization procedures and less prone to entrapment in local optima. However, GAs usually require a large number of fitness evaluations in order to reach a satisfactory solution, and when expensive simulations are involved, that can become a serious drawback to their application to larger problems.

The idea of reducing the computation time by performing (approximate) less computationally expensive fitness function evaluations appeared early in the evolutionary computation literature but its actual implementation is problem dependent and not straightforward. The development of methodologies that aim to reduce the computational cost without substantial loss of quality in the final solution is an active area of research in Genetic and Evolutionary Computation [1–7].

A common structural optimization problem is the weight minimization of framed structures subjected to stress, displacements and other constraints. In general, the constraints verification involves a simulation, performed by a simulation model. A simulation model is a computer model (computer program), that attempts to simulate an abstract model of a particular system, such as the solution of a finite element model. For some real-world problems this simulation can be excessively time-consuming. Due to the characteristics of the constrained optimization problem, and the number of evaluations needed by GAs, their application in several cases can be prohibitive.

To reduce computing time, the introduction of metamodels during the search procedure is proposed. Metamodels are inexpensive models (i) derived from numerical or physical simplifications of the simulation model, (ii) constructed under less rigorous physical assumptions, or (iii) based on a set of samples evaluated using the simulation model. A metamodel for the objective function and another for constraint violation are built. The constraints are handled here by a stochastic tournament selection procedure which does not require penalty parameters. The procedure is implemented in a binary-coded generational GA and the metamodels are updated at each generation by using the simulation model.

The procedure is applied to structural optimization problems involving discrete and continuous design variables and numerical experiments are performed in order to verify the applicability and assess the performance gains of the proposed technique.

The results obtained show that the metamodel leads to computational gains by reducing the number of simulations, significantly reducing the total computational time. However, there is a relationship between the frequency of the use of metamodel and the quality of the final solution. It is clear that, although faster, metamodels are less accurate than the simulation model, and depending on the rate of application of the metamodel, it may affect in a negative manner the evolutionary process.

2. THE GENETIC ALGORITHM

Pioneered by John Holland [8], GAs have been widely applied in many fields of engineering and science. GAs differ from the traditional methods of search and optimization, mainly by [9]: (i) operating upon a codification of the set of parameters instead of the parameters themselves, (ii) maintaining a population of candidate solutions, (iii) requiring only objective function values, and (iv) using probabilistic transition rules.

The popularity of GAs is partially due to the facts that (i) they do not require continuity and/or differentiability of the functions involved, (ii) they do not require extensive problem (re-)formulation, (iii) are not very sensitive to the initialization procedure, (iv) they are less prone to entrapment in local optima, and (v) are naturally parallel.

However, there are some drawbacks for the application of GAs in real world applications. GAs in general require a higher number of evaluations in order to achieve a satisfactory solution, thus in some computationally expensive problems the application of GAs can become prohibitive.

The first step in a GA is to encode all the variables corresponding to a candidate solution in a chromosome. Here we adopted the standard binary coding: each variable is encoded in a string of binary digits of a convenient length and these strings are then concatenated to form a single string which is an individual in the population of candidate solutions. The following step is to randomly generate an initial population. Each individual has then an objective function value and, in cases of constrained optimization, a measure of constraint violation value associated with it. The population is sorted according to these values in order to establish a “ranking”. Individuals are then selected for reproduction in a way that better performing solutions have

a higher probability of being selected. The genetic material contained in the chromosome of such "parent" individuals is then recombined and mutated, by means of crossover and mutation operators, giving rise to offspring which will form a (hopefully improved) new generation of individuals. The process is repeated for a given number of generations or until certain stopping criteria are met.

The baseline GA presented here implements Gray code [10], rank-based selection and elitism, where the best individual is copied into the next generation along with one mutated copy. The standard 2-point and uniform crossover operators are applied with 0.80 and 0.20 probability respectively. The mutation rate p_m is set to the inverse of the chromosome length.

3. THE STRUCTURAL OPTIMIZATION PROBLEM

The discrete structural optimization problem considered here consists in finding the set of areas $\mathbf{a} = \{A_1, A_2, ..., A_n\}$ which minimizes the weight of the truss structure:

$$w(\mathbf{a}) = \sum_{k=1}^{n} \gamma A_k \left(\sum_{j=1}^{N_G} L_j \right) \tag{1}$$

subject to the normalized displacements constraints

$$\frac{|u_{i,l}|}{u_{adm}} - 1 \leq 0 \qquad 1 \leq i \leq M, \quad 1 \leq l \leq N_L \tag{2}$$

and the normalized stress constraints

$$\frac{|s_{j,l}|}{s_{adm}} - 1 \leq 0 \qquad 1 \leq j \leq N, \quad 1 \leq l \leq N_L \tag{3}$$

where γ is the specific weight of the material, L_j is the length of jth bar of the structure, u_i and s_j are respectively the nodal displacement of the ith translational degree of freedom and the stress of the jth bar, s_{adm} is the allowable stress for the material, and u_{adm} is the maximum displacement for each nodal point. M is the number of translational degrees of freedom, N is the total number of bars in the truss structure, N_G is the number of member groups which share the same cross-sectional area, and N_L is the number of load cases applied to the structure.

Although the function w from Eq. (1) is linear, the constraints (2) and (3) are nonlinear implicit functions of the design variables **a** and require the solution of the equilibrium equations of the discrete model given by

$$\mathbf{K(a)u}_l = \mathbf{f}_l \qquad 1 \leq l \leq N_L. \tag{4}$$

$\mathbf{K}$ is the symmetric and positive definite stiffness matrix of the structure, derived from the finite element formulation, given by

$$\mathbf{K} = \mathop{\mathsf{A}}_{j=1}^{N} K_j \tag{5}$$

where A denotes the operator used for assembling the matrix contribution K_j of the jth bar, which is a linear function of **a**. The vector of nodal displacements is denoted by $\mathbf{u}_l$, and $\mathbf{f}_l$ is the vector of applied nodal forces for the lth load condition.

For each one of the load conditions, the system is solved for the displacement field

$$\mathbf{u}_l = [\mathbf{K}(\mathbf{a})]^{-1} \mathbf{f}_l. \tag{6}$$

The stress in the jth bar is calculated according to Hooke's Law

$$s_{j,l} = E\varepsilon(\mathbf{u}_l) \tag{7}$$

where E is the Young's modulus and ε is the unit change in length of the bar.

From the displacements at the nodal points, and the stresses in the elements, the constraints can finally be checked. A feasible design satisfies the constraints (2) and (3), and the degree of constraint violation $\phi(\mathbf{a})$ is measured by

$$\phi(\mathbf{a}) = \sum_{l=1}^{N_L} \left(\sum_{j=1}^{N} \left[\frac{|s_{j,l}|}{s_{adm}} - 1 \right]_+ + \sum_{i=1}^{M} \left[\frac{|u_{i,l}|}{u_{adm}} - 1 \right]_+ \right) \tag{8}$$

where $[x]_+ = x$ if x is positive, and 0 otherwise. Clearly, feasible designs have the constraint violation $\phi(\mathbf{a})$=0.

To avoid introducing penalty parameters into evolutionary procedure, constraints are handling using the stochastic ranking procedure [11]. In the stochastic ranking technique the balance between the objective and penalty functions is achieved through a ranking procedure based on the stochastic bubble-sort algorithm. In this approach a probability p_f of using only the objective function for comparing individuals in the infeasible region of the search space is introduced. Given any pair of two adjacent individuals, the probability of comparing them (in order to determine which one is fitter) according to the objective function is 1 if both individuals are feasible, and p_f otherwise . The procedure provides a convenient way of balancing the dominance in a ranked set. The procedure is halted when no change in the rank ordering occurs within a complete sweep.

Algorithm 1 shows the stochastic bubble sort procedure used to rank a group of individuals.

Because one is interested at the end in feasible solutions, p_f should be less than 0.5, so that there is a pressure against infeasible solutions. In this paper the parameter p_f is set to 0.40 for all test problems.

4. METAMODELING APPROACH

The application of GAs in real world engineering optimization problems often requires the use of a simulation model to evaluate the individuals in the population. In many cases, each simulation takes a considerable amount of time and thus for optimization problems that require a high number of evaluations, the overall computational time becomes a critical issue.

In the following the metamodel concept and its relationship with simulation models are explained.

4.1 Simulation models

Simulation models are used to compute the system response corresponding to a given set of design variables. A simulation model is a representation of the real system, constructed from information about how the system operates [12]. Simulation models are not generic, each of them strongly related to a specific physical problem.

In structural engineering, a simulation model for stress and displacements is often constructed by the finite element method [13].

4.2 Metamodels

Metamodels are inexpensive models, numerical or physical simplifications of the simulation model, constructed under less rigorous physical assumptions or based on a

Algorithm 1 Stochastic Ranking Algorithm

```
1: procedure STOCHASTIC RANKING(I, f, φ, λ, p_f)
2:     for j = 1 : λ do
3:         I_j = j
4:     end for
5:     for j = 1 : λ do
6:         swap ← false
7:         for j = 1 : λ − 1 do
8:             u = RANDOM(0, 1)
9:             if φ_{I_j} = φ_{I_{j+1}} = 0 or u < p_f then
10:                if f_{I_j} > f_{I_{j+1}} then
11:                    tmp = I_{j+1}
12:                    I_{j+1} = I_j
13:                    I_j = tmp
14:                    swap ← true
15:                end if
16:            else
17:                if φ_{I_j} > φ_{I_{j+1}} then
18:                    tmp = I_{j+1}
19:                    I_{j+1} = I_j
20:                    I_j = tmp
21:                    swap ← true
22:                end if
23:            end if
24:        end for
25:        if not swap then
26:            BREAK
27:        end if
28:    end for
29: end procedure
```

set of samples evaluated using the simulation model. Metamodels are approximations of the input/output relations of the simulation model [14–16]. They aim to reduce model complexity [17], and may be understood as surrogate evaluation models that are built using existing information [18].

Others techniques, specific for genetic and evolutionary algorithms, such as evaluation relaxation [19], fitness estimation [20, 21], and genetic inheritance [22, 23], can also be referred to as metamodeling techniques.

The metamodel treats the simulation model as a black box [24]: the simulation model's input/output relation is observed, and the parameters of the metamodel are estimated. The underlying premise of this approach is that, one can construct an approximation of the analysis codes that is much more efficient to run, and which yields an insight into the functional relationship between the input variables and the responses [25].

For metamodels to be useful, they must represent the input-output relations accurately.

An advantage of the metamodeling approach is that the metamodel can be applied to all types of simulation models [24], to provide an insight to the input-output behavior of the original model, and to identify key variables. However, it cannot take advantage of the specific

structure of a given simulation model, and depending on the estimation phase may take a long computation time.

In the following, the metamodel assisted genetic algorithm proposed here is described.

4.3 Metamodel assisted genetic algorithm

In the Evolutionary Computation context, the metamodel, built from previously evaluated solutions in the search space, is used to predict the fitness of new candidate solutions [26] avoiding expensive simulations, reducing the amount of computational effort required.

The most simple and transparent approximation model is the nearest neighbor (NN) model. The approximations are built based on a set $\mathcal{V}$, which stores η individuals evaluated by the simulation model. The approximation for this method is constructed as follows: given an offspring x^h, the corresponding fitness value $f(x^h)$, based on its k nearest neighbors, is given by

$$\hat{f}(x^h) = \frac{\sum_{i=1}^{k} w_i f(x^i)}{\sum_{i=1}^{k} w_i}, \quad x^i \in \mathcal{V} \tag{9}$$

where

$$w_i = \frac{1}{d(x^h, x^i)^u} \qquad \text{if } d(x^h, x^i) \neq 0, \quad x^i \in \mathcal{V} \tag{10}$$

and $d(x^h, x^i)$ is the distance between the individuals x^h and x_i, and $\hat{f}(x^h)$ is the fitness value to assigned x^h. If $d(x^h, x^i) = 0$, then the fitness value of x_i is assigned immediately to x^h, and the weights are not calculated. The metric $d(\cdot, \cdot)$ used here is the Hamming distance, that is the number of positions for which the corresponding bits are different for two chromosomes. In this paper the exponent u is set to 2.

The convenience in using the nearest neighbor method is that the approximation requires only storing points evaluated using the simulation model. If the size η of the archived set $\mathcal{V}$ is large, then the calculation of $\hat{f}(x^h)$ from Eq. (9) takes longer, since the entire set must be sorted to define the nearest neighbors.

To avoid the sorting step, the proposed approach consists in choosing the neighbors by means of a tournament selection, where p individuals are draw randomly out of the archive $\mathcal{V}$, the Hamming distances are calculated and the nearest one is selected to be a neighbor. The procedure is repeated until all the k neighbors are chosen. The procedure described above is named (k,p)-tournament approximation, and the weights w_i are computed according to Eq. (10). If k=1, the algorithm assigns to $f(x^h)$ the fitness value of the individual nearest to x^h according to the p-tournament.

In the experiments conducted here the size η of the archive $\mathcal{V}$ is equal to the population size and the tournament size p is set to 20.

To improve the quality of the approximations in Eq. (9) the metamodel is updated every generation. According to sequential policy, an individual is chosen to be replaced in $\mathcal{V}$. The policy adopted is to choose an individual according to the number of evaluations performed by the simulation model. In this way, the individuals in $\mathcal{V}$ are uniformely replaced along to the generations.

The nearest neighbor metamodel is introduced in the GA according to an user defined parameter p_{sm}, which gives the probability of an individual to be evaluated according to the simulation model. As p_{sm} decreases, greater is the interference of the metamodel in

the evolutionary process. The number of individuals evaluated by the metamodel is thus proportional to $p_{mm} = 1 - p_{sm}$.

The metamodel assisted genetic algorithm implemented here is displayed in Algorithm 2.

Algorithm 2 Metamodel Assisted Genetic Algorithm

```
1:  procedure MA-GA
2:      t ← 0
3:      Initialize the population P_t
4:      Evaluate each chromosome in P_t
5:      Rank the population P_t
6:      Initialize the archive V ← P_t
7:      while t ≤ maxgen do
8:          repeat
9:              Select 2 individuals in P_t
10:             Apply recombination on selected individuals
11:             Apply mutation with rate p_m
12:             Insert new individuals in G_t
13:         until population G_t complete
14:         repeat
15:             r=RANDOM(0, 1)
16:             if r < p_sm then
17:                 Evaluate individual by the simulation model
18:             else
19:                 Evaluate individual by the metamodel
20:             end if
21:         until all individuals in population G_t evaluated
22:         Rank the population G_t
23:         P_{t+1} ← G_t
24:         Update V with the individuals evaluated by the simulation model
25:         t ← t + 1
26:     end while
27: end procedure
```

5. NUMERICAL EXPERIMENTS

In order to investigate the performance of the implemented algorithms, they are applied to some truss weight minimization problems.

In the following, the test problems are described and the results are discussed. The size of the archive $\mathcal{V}$ is set equal to the population size, and the tournament size is set to 20 individuals. The results are compared for different number of neighbors from Eq. (9) and several values of p_{sm}. A total of 30 independent runs were performed for each test-problem.

5.1 The 22-bar truss

The first structure considered is the plane truss shown schematically in Fig. 1, where a vertical load P is applied at the rightmost node. The weight of the structure is to be minimized, and the design variables are the cross-sectional areas of the bars $\mathbf{a} = \{A_1, A_2, ..., A_8\}$ which are to be chosen from the 32 values in the set $\mathcal{K} = \{0.1, 0.2, \ldots, 3.1, 3.2\}$.

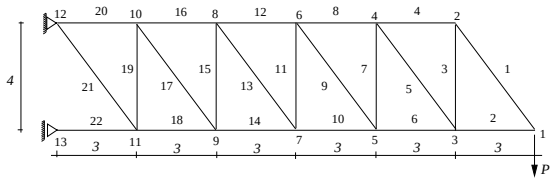

Figure 1: The 22-bar truss.

As an isostatic structure, the forces acting in each bar (compression or tension) do not depend on the value of the cross-sectional areas adopted. It follows that each bar should be working at the maximum allowable stress,

$$\frac{f_i}{A_i} = s_c \quad \text{or} \quad \frac{f_i}{A_i} = s_t,$$

if the ith bar is in compression or tension, respectively, and s_c and s_t are the corresponding allowable limits. The optimal design is thus a "fully stressed" structure. It can also be shown that in the structure considered (Fig. 1), all vertical bars are working under the same conditions, and that the same can be said of the diagonal bars, resulting in 8 design variables. Additionally, for the case of equal material behavior in tension and compression, $s_t = s_c = s_{adm}$. A consistent set of units is assumed and the following values are adopted: $P = 12$, and $s_c = s_t = s_{adm} = 25$. The exact solution for the optimization problem can be found analytically (the minimum volume is equal to 68.20) and controlled numerical experiments and comparisons can more easily be performed.

For this experiment, the population size was set equal to 50, 80 generations were performed in 30 independent runs, and 5 bits were used for each one of the 8 variables, resulting in a 40-bit chromosome, and a mutation rate $p_m = 1/40$.

The results for 22-bar truss are presented in Table 1. The first line shows the results obtained using only the simulation model to evaluate the individuals. The number of neighbors k from Eq. (9), the size of the tournament p, the parameters p_{sm} and p_{mm} which control how many individuals are evaluated using the simulation model or the metamodel are also displayed. In this Table, FR denotes the number of runs (among the 30) which produced a feasible final solution. Also are displayed the minimum ($best$) and maximum ($worst$) weights found as well as the average (avg) and standard deviation observed (std) in the 30 runs. It is important to notice that the average is calculated considering only the runs which produced feasible solutions.

From the Table 1 it can be noticed that the values of the best values do not suffer significant modifications as the parameter p_{sm} increases, although the average in 30 runs increases, that is, as the number of individuals evaluated by the metamodel increases, the evolutionary process is affected, leading in average to worse solutions, as expected. The same behavior is observed when the worst values are compared, and when only 30% of the individuals are evaluated by simulation model (in average), the final solutions have a considerable decrease of quality, as can be seen in the last five lines of the Table 1.

It can also observed that, for larger values of p_{sm}, the best results in terms of best, average and standard deviation values are found for the smaller values of k. However, for smaller values of p_{sm}, where more individuals are evaluated using the metamodel, better results are found using larger values of k.

It is important to notice that, although the evolutionary process is affected and the quality of the solutions decreases, the introduction of the metamodel represents substantial computational

Table 1: Results for 22-bar truss.

k	p	p_{sm}	p_{mm}	FR	$best$	avg	$worst$	std
–	–	1.00	0.00	30	68.20	68.33	68.80	0.25
1	20	0.90	0.10	29	68.20	68.38	69.10	0.31
2	20	0.90	0.10	30	68.20	68.34	69.40	0.31
3	20	0.90	0.10	30	68.20	68.45	69.40	0.39
5	20	0.90	0.10	30	68.20	68.43	69.40	0.34
10	20	0.90	0.10	30	68.20	68.56	72.40	0.80
1	20	0.80	0.20	30	68.20	68.57	70.00	0.54
2	20	0.80	0.20	29	68.20	68.54	70.60	0.59
3	20	0.80	0.20	30	68.20	68.52	70.30	0.54
5	20	0.80	0.20	30	68.20	68.64	70.60	0.58
10	20	0.80	0.20	30	68.20	68.68	70.90	0.70
1	20	0.70	0.30	30	68.20	68.95	70.30	0.71
2	20	0.70	0.30	30	68.20	68.72	70.60	0.66
3	20	0.70	0.30	30	68.20	68.55	70.00	0.49
5	20	0.70	0.30	30	68.20	68.70	71.20	0.66
10	20	0.70	0.30	30	68.20	68.99	72.10	0.94
1	20	0.60	0.40	30	68.20	69.66	72.70	1.22
2	20	0.60	0.40	30	68.20	69.19	71.20	1.01
3	20	0.60	0.40	30	68.20	69.21	71.50	0.93
5	20	0.60	0.40	29	68.20	69.09	71.20	0.77
10	20	0.60	0.40	29	68.20	68.97	70.90	0.74
1	20	0.50	0.50	30	68.20	71.04	75.00	1.87
2	20	0.50	0.50	30	68.20	70.23	79.90	2.57
3	20	0.50	0.50	30	68.20	69.25	71.50	0.88
5	20	0.50	0.50	29	68.20	69.25	73.00	1.07
10	20	0.50	0.50	30	68.20	69.53	71.80	0.96
1	20	0.40	0.60	28	68.20	73.24	86.20	3.81
2	20	0.40	0.60	30	68.20	70.09	73.30	1.39
3	20	0.40	0.60	30	68.20	69.90	72.40	1.16
5	20	0.40	0.60	29	68.20	69.82	71.80	1.13
10	20	0.40	0.60	30	68.20	70.07	73.60	1.49
1	20	0.30	0.70	30	70.00	78.15	103.00	8.60
2	20	0.30	0.70	29	68.20	71.46	76.00	2.00
3	20	0.30	0.70	30	68.20	71.84	78.70	2.42
5	20	0.30	0.70	30	68.50	71.31	75.80	1.91
10	20	0.30	0.70	30	68.80	71.43	78.70	2.06

savings, since an expensive simulation is replaced by a simple computation using an ensemble of stored values.

5.2 The 10-bar truss

This test-problem corresponds to the weight minimization of the classic 10-bar truss [27] shown in the Fig. 2.

The constraints involve the stress in each member and the displacements at the nodal points. The design variables are the cross-sectional areas of the bars $\mathbf{a} = \{A_1, A_2, \ldots, A_{10}\}$. The allowable stress is limited to ± 25 ksi and displacements are limited to 2 in, in the x and y directions. The density of the material is 0.10 lb/in^3, Young modulus is $E = 10^4$ ksi and vertical nodal loads of 100 kips are applied at nodes 2 and 4.

The values of cross-sectional areas, in square inches, are to be chosen from the set $\mathcal{L}$ = { 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 } resulting in 42 options for each bar in the structure.

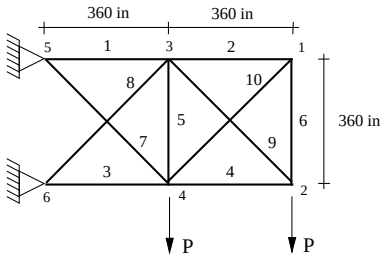

Figure 2: The 10-bar truss.

The population size was set to 100 individuals and 900 generations were performed in 30 independent runs. Each one of the 10 variables was coded using 6 bits, leading to a 60-bit chromosome. The mutation rate was equal to $p_m = 1/60$. For each run, 90000 evaluations were performed.

The Table 2 presents the results for the 10-bar truss problem. The first line of the Table shows the results using only the simulation model to evaluate the individuals. Again, the best, average, worst, and standard deviation values found in the 30 runs are displayed, together with the number of neighbors k and the values of p_{sm}. FR indicates the number of runs (among the 30) which produced a feasible final solution.

The results shown in Table 2 present a similar behavior when compared to those obtained for the 22-bar truss problem. Here, it can also be noticed that the values of the average in 30 runs increase as p_{sm} decreases, independent of the number of neighbors k. Also, for larger values of p_{sm}, the best results in terms of best, average and standard deviation values are found for the smaller values of k, better results are found for larger values of k when smaller values of p_{sm} are set.

5.3 The 25-bar truss

This classical problem [27] is the weight minimization of a truss with 25 bars shown in the Fig. 3. The allowable stress for each member is $s_{adm} = 40$ ksi and the displacements must not exceed $u_{adm} = 0.35$ in, in the x and y directions. The material has a Young modulus $E = 10^7$

Table 2: Results for 10-bar truss.

k	p	p_{sm}	p_{mm}	FR	$best$	avg	$worst$	std
–	–	1.00	0.00	30	5490.738	5496.452	5513.418	8.673
1	20	0.90	0.10	30	5490.738	5508.265	5569.510	19.894
2	20	0.90	0.10	30	5490.738	5511.478	5599.918	28.988
3	20	0.90	0.10	30	5490.738	5499.333	5537.516	13.062
5	20	0.90	0.10	29	5490.738	5510.825	5593.193	25.002
10	20	0.90	0.10	30	5490.738	5501.639	5536.965	14.266
1	20	0.80	0.20	30	5490.738	5507.823	5549.423	16.265
2	20	0.80	0.20	29	5490.738	5504.085	5545.250	15.876
3	20	0.80	0.20	30	5490.738	5508.440	5567.509	22.407
5	20	0.80	0.20	30	5490.738	5506.981	5585.147	20.813
10	20	0.80	0.20	30	5490.738	5509.895	5550.386	15.375
1	20	0.70	0.30	30	5490.738	5517.552	5650.360	32.311
2	20	0.70	0.30	30	5490.738	5518.282	5576.705	26.825
3	20	0.70	0.30	30	5490.738	5512.095	5582.307	23.564
5	20	0.70	0.30	30	5490.738	5515.692	5593.529	25.705
10	20	0.70	0.30	30	5490.738	5510.055	5549.446	17.102
1	20	0.60	0.40	30	5490.738	5529.881	5610.702	29.964
2	20	0.60	0.40	30	5490.738	5519.181	5602.136	26.181
3	20	0.60	0.40	30	5490.738	5511.048	5574.393	22.721
5	20	0.60	0.40	30	5490.738	5521.369	5582.463	26.069
10	20	0.60	0.40	29	5490.738	5521.167	5593.224	30.022
1	20	0.50	0.50	30	5503.934	5600.230	5730.343	69.106
2	20	0.50	0.50	30	5490.738	5536.888	5753.562	51.106
3	20	0.50	0.50	30	5490.738	5518.768	5588.854	24.380
5	20	0.50	0.50	30	5490.738	5528.218	5658.424	36.497
10	20	0.50	0.50	30	5490.738	5528.106	5579.044	28.042
1	20	0.40	0.60	30	5511.358	5644.434	5783.558	69.006
2	20	0.40	0.60	30	5497.218	5556.216	5677.276	49.299
3	20	0.40	0.60	30	5490.738	5541.256	5660.241	41.053
5	20	0.40	0.60	30	5490.738	5533.438	5615.039	35.296
10	20	0.40	0.60	30	5490.738	5536.082	5631.113	39.218
1	20	0.30	0.70	30	5522.221	5702.771	6056.010	118.014
2	20	0.30	0.70	29	5507.758	5603.588	5759.292	63.341
3	20	0.30	0.70	29	5507.539	5602.149	5782.929	67.988
5	20	0.30	0.70	30	5490.738	5557.172	5682.782	44.967
10	20	0.30	0.70	30	5490.738	5546.229	5632.076	40.723

psi and density of 0.10 lb/in^3. The loads applied in the structure are displayed in the Table 3.

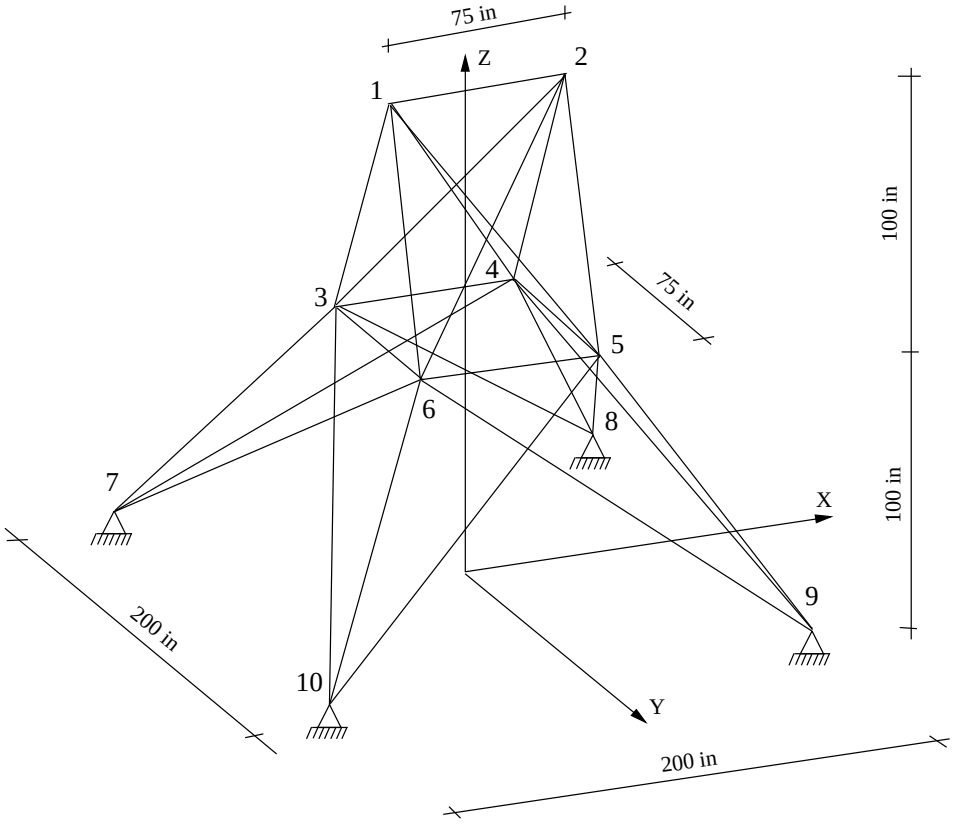

Figure 3: The 25-bar truss.

The design variables are the cross-sectional areas which are organized into eight groups, as shown in Table 4. This arrangement results in a structural optimization problem with eight discrete variables, to be chosen from the set of 34 values (in square inches) $\mathcal{M}$ = { 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, 2.7, 2.8,2.9, 3.0, 3.1, 3.2, 3.3, 3.4 }.

Table 3: Load case for 25-bar truss.

Node	F_x (kips)	F_y (kips)	F_z (kips)
1	1.00	-10.00	-10.00
2	–	-10.00	-10.00
3	0.50	–	–
6	0.60	–	–

The population size was set to 100, and 200 generations were performed. Five bits were used for each one of the 8 variables, resulting on a 40-bit chromosome, and a mutation rate $p_m = 1/40$. A total of 20000 evaluations are performed for each run.

The results for the 25-bar truss problem are shown in Table 5. The first line of the Table shows the results obtained using only the simulation model to evaluate the individuals. Again, 30 independent runs were performed for each number of neighbors k and value of the parameter p_{sm}.

Table 4: Member groups for 25-bar truss.

Group	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

The results displayed in Table 5 show a similar behavior as that observed for both the 22-bar and the 10-bar truss problems, although for this test problem the results for smaller values of p_{sm} are significantly worse than the results obtained with larger values of p_{sm}, when compared the values of average in 30 runs, the worst solution and the standard deviation. In Table 5 that can be seen when $p_{sm} = 0.40$ and $p_{sm} = 0.30$.

It can be noticed that the algorithm produces satisfactory solutions up to a certain level of metamodel insertion: as the metamodel insertion grows, the quality of the results decrease. However, the computational gains increase in the same way. There is a trade off between the quality of the solutions found and the computation time of the optimization procedure, and for time-consuming problems, the use of metamodels is an attractive alternative to obtain computational gains which allow for the solution of more complex problems with a given computational budget.

6. SUMMARY AND CONCLUSIONS

In this paper a metamodel-assisted genetic algorithm, which implements the nearest-neighbor approximation is presented. Essentially, the approach consists in the correct evaluation of only a fraction of the population. The proposed procedure is applied to structural optimization problems involving discrete design variables.

The introduction of the nearest neighbor metamodel reduces the use of the simulation model during the search procedure performed by the GA, allowing for computational gains. However there is a relationship between the frequency of use of the approximation model and the quality in the final solutions.

From the results presented here, satisfactory solutions are found for values of p_{sm} as low as 0.5, which means that the results can be trusted at a limit when in average 50% of the evaluations are done using the simulation model. Besides, according to the obtained results, for smaller values of p_{sm}, which is desirable when dealing with computationally expensive problems, larger values of k lead to better results.

It is clear that, although faster, metamodel evaluations are less accurate and, depending on the rate of application of the metamodel during the search procedure, it may affect in a negative manner, leading to unsatisfactory solutions.

Although implemented here in a binary-coded GA, the procedures can be easily implemented in a real-coded GA, where a wide range of approximation techniques exist, and may potentially lead to better results. The procedure presented here make use only of stored instances, and due to the features of the approximation techniques, it is possible to approximate

Table 5: Results for 25-bar truss.

k	p	p_{sm}	p_{mm}	FR	$best$	avg	$worst$	std
–	–	1.00	0.00	30	484.854	485.780	488.443	0.822
1	20	0.90	0.10	30	484.854	485.920	489.885	1.262
2	20	0.90	0.10	30	484.854	486.004	491.385	1.253
3	20	0.90	0.10	30	484.854	485.835	488.769	0.969
5	20	0.90	0.10	30	484.854	486.087	490.140	1.401
10	20	0.90	0.10	30	484.854	485.886	488.438	0.925
1	20	0.80	0.20	30	484.854	486.412	489.165	1.084
2	20	0.80	0.20	30	484.854	486.022	489.691	1.083
3	20	0.80	0.20	30	484.854	486.329	491.114	1.416
5	20	0.80	0.20	30	485.049	486.131	488.574	0.959
10	20	0.80	0.20	30	485.049	486.220	489.413	1.168
1	20	0.70	0.30	30	485.574	487.150	497.880	2.573
2	20	0.70	0.30	30	484.854	486.534	492.496	1.735
3	20	0.70	0.30	30	484.854	486.180	489.938	1.289
5	20	0.70	0.30	30	484.854	487.658	501.313	3.246
10	20	0.70	0.30	30	484.854	486.857	493.529	1.756
1	20	0.60	0.40	30	484.854	487.259	491.260	1.857
2	20	0.60	0.40	30	485.049	487.123	490.871	1.587
3	20	0.60	0.40	30	484.854	487.084	497.078	2.603
5	20	0.60	0.40	30	484.854	488.113	503.067	4.145
10	20	0.60	0.40	30	484.854	488.157	507.378	4.652
1	20	0.50	0.50	30	485.049	489.772	502.086	3.674
2	20	0.50	0.50	30	484.854	487.840	494.326	2.869
3	20	0.50	0.50	30	484.854	488.122	496.553	2.922
5	20	0.50	0.50	30	485.380	488.899	503.540	3.858
10	20	0.50	0.50	30	484.854	489.032	501.891	3.536
1	20	0.40	0.60	30	485.049	492.527	523.659	7.994
2	20	0.40	0.60	29	484.854	491.293	505.463	5.318
3	20	0.40	0.60	30	485.380	491.477	506.617	5.359
5	20	0.40	0.60	30	485.049	491.582	508.436	6.546
10	20	0.40	0.60	30	485.049	490.049	499.618	4.309
1	20	0.30	0.70	30	485.769	495.835	520.019	9.080
2	20	0.30	0.70	30	485.380	493.638	514.072	7.972
3	20	0.30	0.70	30	485.049	493.489	519.726	7.651
5	20	0.30	0.70	30	485.049	492.811	505.618	6.355
10	20	0.30	0.70	30	484.854	492.309	509.740	6.492

the responses for problems with discrete, continuous, as well as mixed discrete-continuous variables.

Moreover, the procedure is simple and can be implemented in a GA,or any other population-based algorithm.

Acknowledgment

The authors acknowledge the support received from CNPq (grants 302299/2003-3 and 154674/2006-0) and PRONEX/FAPERJ E-26.171.199/2003.

REFERENCES

[1] L. Bull, "On model-based evolutionary computation," *Soft Computing*, vol. 3, no. 2, pp. 76 – 82, 1999.

[2] K.-H. Liang, X. Yao, and C. Newton, "Evolutionary search of approximated N-dimensional landscapes," *International Journal of Knowledge-Based Intelligent Engineering Systems*, vol. 4, pp. 172–183, July 2000.

[3] T. W. Simpson, J. Poplinski, P. N. Koch, and J. Allen, "Metamodels for computer-based engineering design: Survey and recommendations," *Engineering with Computers*, vol. 17, pp. 129–150, July 2001.

[4] Z. Zhou, Y. S. Ong, and P. B. Nair, "Hierarchical surrogate-assisted evolutionary optimization framework," in *Congress on Evolutionary Computation*, pp. 1586–1593, IEEE, 2004.

[5] T. W. Simpson, A. J. Booker, D. Ghosh, A. A. Guinta, P. N. Koch, and R.-J. Yang, "Approximation methods in multidisciplinary analysis ans optimization: a panel discussion," *Structural and Multidisciplinary Optimization*, vol. 27, pp. 302–313, 2004.

[6] J. Branke and C. Schmidt, "Faster convergence by means of fitness estimation," *Soft Computing*, vol. 9, no. 1, pp. 13–20, 2005.

[7] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing Journal*, vol. 9, no. 1, pp. 3–12, 2005.

[8] J. H. Holland, *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[9] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Co., 1989. Reading, Mass., USA.

[10] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Verlag, third ed., 1996.

[11] T. P. Runarsson and X. Yao, "Stochastic Ranking for Constrained Evolutionary Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, pp. 284–294, September 2000.

[12] V. C. Chen, K.-L. Tsui, R. R. Barton, and M. Meckesheimer, "A review on design, modeling and applications of computer experiments," *IIE Transactions*, vol. 38, no. 4, pp. 273 – 291, 2006. DOI:10.1080/07408170500232495.

[13] T. J. R. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. New Jersey: Prentice Hall International Inc., 1987.

[14] R. R. Barton, "Metamodels for simulation input-output relations," in *WSC '92: Proceedings of the 24th conference on Winter simulation*, (New York, NY, USA), pp. 289–299, ACM Press, 1992.

[15] J. P. C. Kleijnen and R. G. Sargent, "A methodology of fitting and validating mftamodels in simulation," *European Journal of Operational Research*, vol. 120, pp. 14–29, 2000.

[16] J. Fonseca, D. O. Navaresse, and G. P. Moynihan, "Simulation metamodeling through artificial neural networks," *Engineering Applications of Artificial Intelligence*, vol. 16, no. 3, pp. 177–183, 2003.

[17] T. Yang, H.-C. Lin, and M.-L. Chen, "Metamodeling approach in solving the machine parameters optimization problem using neural network and genetic algorithms: A case study," *Robotics and Computer-Integrated Manufacturing*, vol. 22, no. 4, pp. 322–331, 2006.

[18] M. Emmerich, K. Giannakoglou, and B. Naujoks, "Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels," *Evolutionary Computation*, vol. 10, no. 4, pp. 421–439, 2006.

[19] K. Sastry, C. F. Lima, and D. E. Goldberg, "Evaluation relaxation using substructural information and linear estimation," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, (New York, NY, USA), pp. 419–426, ACM Press, 2006.

[20] Y. Hanaki, T. Hashiyama, and S. Okuma, "Accelerated evolutionary computation using fitness estimation," in *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on*, vol. 1, pp. 643 – 648, 1999.

[21] F. Mota. and F. Gomide, "Fuzzy clustering in fitness estimation models for genetic algorithms and applications," in *Fuzzy Systems, 2006 IEEE International Conference on*, pp. 1388– 1395, July 16-21 2006. ISBN: 0-7803-9488-7.

[22] R. E. Smith, B. A. Dike, and S. A. Stegmann, "Fitness inheritance in genetic algorithms," in *SAC '95: Proceedings of the 1995 ACM symposium on Applied computing*, (New York, NY, USA), pp. 345–350, ACM Press, 1995.

[23] K. Sastry, D. Goldberg, and M. Pelikan, "Don't evaluate, inherit," in *Proceedings of genetic and Evolutionary Computation Conference*, pp. 551–558, Morgan Kaufmann, 2001.

[24] J. P. C. Kleijnen and W. C. M. van Beers, "Kriging for interpolation in random simulation," Tech. Rep. 2001-74, Tilburg University, Tilburg, Netherlands, Deprtament of Infomation Systems, October 2001.

[25] M. El-Beltagy, P. Nair, and A. Keane, "Metamodeling techniques for evolutionary optimization of computationally expensive problems: promises and limitations," in *Proceedings of Genetic and Evolutionary Conference*, (Orlando), pp. 196–203, Morgan Kaufmann, 1999.

[26] M. Emmerich, A. Giotis, M. Özdenir, T. Bäck, and K. Giannakoglou, “Metamodel-assisted evolution strategies,” in *Parallel Problem Solving from Nature*, no. 2439 in Lecture Notes in Computer Science, pp. 371–380, Springer, 2002.

[27] A. Lemonge and H. Barbosa, “An adaptive penalty scheme for genetic algorithms in structural optimization,” *Int. J. Num. Meth. in Engineering*, vol. 59, pp. 703–736, 2004.