INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng. **45**, 1069–1084 (1999)

SOLVING STRUCTURAL OPTIMIZATION PROBLEMS WITH GENETIC ALGORITHMS AND SIMULATED ANNEALING

SALVADOR BOTELLO[1,2], JOSE L. MARROQUIN[2],
EUGENIO OÑATE[3,*] AND JOHAN VAN HOREBEEK[2]

[1] *Universidad de Guanajuato Facultad de Ingenieria Civil, Av. Juarez 77 Guanajuato, Gto. 36000, Mexico*

[2] *Centro de Investigación en Matemáticas, Apdo. Postal 402 Guanajuato, Gto. 36000, Mexico*

[3] *E.T.S. de Ingenieros de Caminos, Canales y Puertos, Universidad Politécnica de Cataluña, Gran Capitán s/n, Módulo C1, 08034 Barcelona, Spain*

SUMMARY

In this paper we study the performance of two stochastic search methods: Genetic Algorithms and Simulated Annealing, applied to the optimization of pin-jointed steel bar structures. We show that it is possible to embed these two schemes into a single parametric family of algorithms, and that optimal performance (in a parallel machine) is obtained by a hybrid scheme. Examples of applications to the optimization of several real steel bar structures are presented. Copyright © 1999 John Wiley & Sons, Ltd.

KEY WORDS: structural optimization; genetic algorithms; simulated annealing; bar structures

1. INTRODUCTION

Structural optimization problems arise frequently in civil engineering applications. They consist in selecting, from a given catalogue, the type (shape and cross-section) of the elements of a given structure in such a way that the total weight (or cost) is minimized, while satisfying the design constraints. Mathematically, these problems may be formulated in terms of combinatorial optimization: if we call x the vector of catalogue entries corresponding to the structure (i.e. x_i is the type of element i), the problem is to find an instance of x that minimizes the cost function

$$U(x) = W(x) + \lambda(\delta_\sigma(x) + \delta_{\mathrm{d}}(x)) \tag{1}$$

where $W(x)$ is the total weight of the structure, $\delta_\sigma(x)$ and $\delta_{\mathrm{d}}(x)$ are the amounts of stress and displacement, respectively, exceeding the maximum permissible value and λ is a large positive number. To compute $\delta_\sigma(x)$ and $\delta_{\mathrm{d}}(x)$ one must solve a linear system whose dimension equals the number of elements in the structure. This solution is, in general, computationally expensive; for

* Correspondence to: Eugenio Oñate, International Centre for Numerical Methods in Engineering, Universidad Politécnica de Cataluña Módulo C1, Campus Norte UPC, Gran Capitán s/n, 08034 Barcelona, Spain. E-mail: onate@etseccpb.upc.es

Contract/grant sponsor: Consejo Nacional de Ciencia y Tecnologia, Mexico

CCC 0029–5981/99/201069–16$17.50

Received 18 November 1996
Revised 15 October 1998

this reason, it is important to have search algorithms which use as few function evaluations as possible, and that may be easily implemented in multi-processor machines.

There are a number of stochastic search techniques that have been successfully applied to solve a variety of complex combinatorial optimization problems similar to this one. The oldest one is probably Simulated Annealing (SA) [1], which generates a sequence of solutions by combining a mutation operation with an increasingly tight acceptance criterion. Other techniques, such as Evolutionary Strategies (ES) [2, 3] and Genetic Algorithms (GA) [4] also involve a mutation operation, but this is applied to a whole population of searching agents which are allowed to interact competing with each other (in the selection step) and interchanging information (in the cross-over step in GAs).

There have been several attempts for combining the strong points of these techniques [5–7]. One of the most successful is possibly Parallel Recombinative Simulated Annealing (PRSA) [8] which uses a population of Metropolis algorithms and cross-over to produce the mutations at each step. In this paper we generalize this idea and show that by including also a selection step, one may get an improved performance in a variety of situations. This is not surprising, since, as we will show, SA, ES, GA and PRSA may be seen as particular instances of a parametric family of algorithms, and therefore, it should be possible to find a specific setting for the parameters that performs at least as well as the best one of the known schemes.

The plan of our presentation is as follows: in Section 2 we introduce the basic operators and present the general family of parallel stochastic search algorithms. In Section 3, we study the experimental behaviour of these algorithms for several structural optimization problems, and finally, in Section 4, we present some recommendations regarding the optimal use of computational resources in multi-processor machines.

2. STOCHASTIC SEARCH ALGORITHMS

The general problem that we are trying to solve is the following: we are given a *state space* $\Omega = Q_1 \times \cdots \times Q_n$ where $Q_i,\ i = 1, \ldots, n$ are finite sets and a function $U : \Omega \mapsto \mathbf{R}$ which will be called the *cost function*, and one wishes to find the vector $x^* = (x^{*1}, \ldots, x^{*n}) \in \Omega$ that minimizes U globally.

2.1. Genetic algorithms

In a Genetic Algorithm, one has a *population* X, that is, an ordered set of N-vectors $(x_1, \ldots, x_N)$, $x_i \in \Omega$ on which operate three basic operators, namely, selection, cross-over, and mutation. Schematically, these operators are concatenated as in figure 1(a). To make this paper self-contained, we give a short description of the basic form of these operators.

2.1.1. Selection. For a given parametrized fitness function $f_\gamma : \Omega \mapsto \mathbf{R}$, selection consists in the choice of N elements from a given population with probability proportional to their fitness. In order to reduce the variance on the weighting between similarity and diversity, some special sampling procedures have been proposed as, e.g. the simple roulette selection and *Stochastic Universal Selection* [9], eventually combined with some explicit adaptations of the algorithm like, e.g. *Tournament Selection* [4, 10], introduction of species or clustering in the population [11] or the adaptation of the cost function.

A simple and efficient scheme is the *Stochastic Remainder Technique*: the underlying philosophy is to put deterministically in the new population the expected number of copies of each element (truncated to a natural number) and allocate the remaining positions in the classical (stochastic) way.

The result is a one-parameter family of selection operators $S_\gamma : \Omega^N \mapsto \Omega^N$ as defined in Algorithm 1.

Operator $S_\gamma(X)$

Compute for each element x_k of X, the relative fitness f_k and the truncated expected number of occurences n_k

$$f_k = \frac{f_\gamma(x_k)}{\sum_{j=1}^{N} f_\gamma(x_j)}$$
$$n_k = Int(N f_k)$$

where the function $Int(\ )$ computes the integer part of its argument

Call $t = \sum_{k=1}^{N} n_k$ and construct the set $(y_1, \ldots, y_t)$ by making n_k copies of each element x_k;

Choose $y_{t+1}, \ldots, y_N$ from X with probability of selecting element x_k equal to

$$\frac{N_{f_k} - n_k}{\sum_{j=1}^{N} (N f_j - n_j)}$$

Set $S_\gamma(X) = (y_1, \ldots, y_N)$

Algorithm 1

The family of fitness functions f_γ is chosen in such a way that for $\gamma = 0$, S_γ becomes the identity (i.e., $S_0(X) = X$). For example, if for all $x \in \Omega$, the cost function $U(x)$ is always positive, one may use

$$f_\gamma(x) = 1 - \gamma(1 + U(x)) \tag{2}$$

with $\gamma \in [0, 1]$.

One may also use the exponential fitness [12]

$$f_\gamma(x) = \exp[-\gamma U(x)] \tag{3}$$

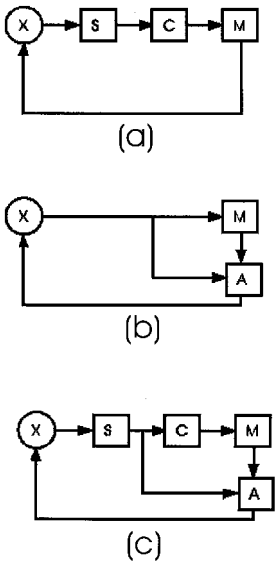

Figure 1. Schematic representation of the different optimization techniques: (a) PGA: population X is fed in parallel to selection (S), cross-over (C) and mutation (M) operators to obtain a new population which repeats the cycle. (b) PSA: the acceptance (A) operator compares in parallel the mutated and original populations to produce a new population. (c) GSSA: combines the features of the algorithm of panels (a) and (b)

which for large values of γ generates an operator S_γ which chooses the best element of a population and reproduces it N times.

2.1.2. Mutation. We may define a one-parameter family of mutation operators $M_\mu : \Omega^N \mapsto \Omega^N$ by Algorithm 2.

Operator $M_\mu(X)$

For each element x of the population do:
  Construct an element y as follows:
    For each component x^i of x do:
      $y^i = r^i$ with prob. $p(\mu,x,X)$
      $= x^i$ with prob. $1 - p(\mu,x,X)$
    where r^i is an element chosen at random from Q_i
      with uniform probability

Set $M_\mu(X) = (y_1, \ldots, y_N)$

Algorithm 2

The mutation probability $p(\mu,x,X)$ may be uniform (i.e. $p(\mu,x,X)=\mu$) or adapt its value to the fitness of x. Thus, in [13] this adaptive probability is defined as

$$p(\mu,x,X)=\begin{cases}\mu\dfrac{f_{\max}-f(x)}{f_{\max}-\bar{f}} & \text{if } f(x)>\bar{f}\\ \mu & \text{if } f(x)\leqslant\bar{f}\end{cases} \tag{4}$$

where $f_{\max}$ and $\bar{f}$ are the maximum and average fitnesses of the population, respectively.

In the *elitist* approach the best individual in the population is transmitted by the operator without mutation (i.e. one defines $p(\mu,x,X)=0$ if x is the best element in X and equal to μ otherwise).

2.1.3. Cross-over. This operator allows to interchange information by splitting two elements of the population in two parts and concatenate them the other way round; in order to improve the diversity, one often requires that every element is used exactly once in a cross-over operation and guarantee that the best element is not lost. The result is a one-parameter family of cross-over operators $C_\xi:\Omega^N\mapsto\Omega^N$ as defined in Algorithm 3 where the cross-over probability $p(\xi,X,x,y)$ may be uniform, or depend adaptively on the fitness values of the involved elements.

Operator $C_\xi(X)$

While some elements of X have not been used in a cross-over:
  Select 2 unused elements x_j,x_k from X;
    with probability $p(\xi,X,x_j,x_k)$, make a cross-over:
      Pick a position r between 1 and n from a uniform distribution;
      Construct $y_j=(x_j^1,\ldots,x_j^r,x_k^{r+1},\ldots,x_k^n)$
        $y_k=(x_k^1,\ldots,x_k^r,x_j^{r+1},\ldots,x_j^n)$
  else with probability $1-p(\xi,X,x_j,x_k)$
    put $y_j=x_j, y_k=x_k$

Set $C_\xi(X)=(y_1,\ldots,y_N)$

Algorithm 3

Other variants of this scheme have been proposed, such as *uniform* or two-point cross-over, in which two positions in each element are selected and then either the substring between them or outside them is interchanged. These schemes usually give better results at the expense of a slight increase in the computational complexity.

2.2. Simulated annealing

In this approach the minimization of a cost function $U(x)$ is performed by constructing a regular Markov [14] chain whose asymptotic distribution is the Gibbs measure

$$P(x) = \frac{1}{Z} \exp[-\beta U(x)]$$

where Z is a normalizing constant and β is a parameter called the *inverse temperature*. A simple way of constructing this chain is the *Metropolis algorithm* [15], which consists in generating, at each step, a candidate state via a mutation operation, and then accepting or rejecting this candidate using a probabilistic criterion (acceptance operator) that depends on β. Algorithm 4 describes this procedure applied to each element of a population X and a candidate mutated population Y. This algorithm, therefore, describes a set of N Metropolis Markov chains operating in parallel.

Operator $A_\beta(X, Y)$

For all elements $x_k \in X, y_k \in Y$ do:
  Set $\Delta U = U(y_k) - U(x_k)$
  iF $\Delta U \leqslant 0$, set $u_k = y_k$;
  if $\Delta U > 0$, set $u_k = y_k$ with prob. $\exp[-\beta \Delta U]$
  and $u_k = x_k$ with prob. $(1 - \exp[-\beta \Delta U])$

Set $A_\beta(X, Y) = (u_1, \ldots, u_N)$

Algorithm 4

The simulated annealing process [1] consists in slowly increasing the value of β, so that the chain remains in equilibrium. When β is high enough, the state of the chain will approximate the global minimum of U. The SA procedure applied to a population of searching agents (i.e. parallel SA with multiple starting points) is represented schematically in Figure 1(b).

2.3. General stochastic search algorithm

The general scheme that we are proposing, combines SA and GA by inserting the acceptance operator after the mutation step in the GA, so that the population obtained after the selection step is compared to the one modified by cross-over and mutation before restarting the cycle. This is represented schematically in Figure 1(c).

This scheme may be used with any implementation of the selection, mutation and cross-over operators, provided only that the last one allows for the identification of corresponding individuals before and after cross-over (i.e. provided that every element is either left unchanged or used exactly once in a cross-over operation). If the selection box is eliminated, one gets the PRSA algorithm, and if both selection and cross-over are left out, one gets parallel SA with multiple starting points (if the population size is greater than 1).

The most interesting behaviour, however, is obtained when all operators are present. It is possible to show that in this case the asymptotic convergence properties of the SA algorithm—i.e. the convergence with probability 1 to the global minimizer of U when β increases at a logarithmic rate—are preserved (see [16] for this and other related theoretical results). Also, as we show in the next section, by inserting the acceptance operator one may significantly improve the experimental behavior of a GA, at least for the class of structural optimization problems we are interested in.

3. EXAMPLES

As mentioned in the introduction, we are interested in minimizing the cost function

$$U(x) = W(x) + \lambda(\delta_\sigma(x) + \delta_d(x)) \tag{5}$$

where $W(x)$ is the total weight of the structure, $\delta_\sigma(x)$ and $\delta_{\mathrm{d}}(x)$ are the amounts of stress and displacement, respectively, exceeding the maximum permissible value and $\lambda = 10\,000$.

We performed three sets of experiments: in the first set we study the effect of free parameters on the performance of GSSA, in the second set was to compare the performance of GSSA with other published optimization methods and in the third section we illustrate the power of the method we are proposing in the optimization of real two and three-dimensional structures.

3.1. Effect of free parameters

The first set of experiments was designed to study the effect of specific parameters on the performance of the GSSA. In particular, we compared the extreme cases where the population size is 1 (i.e. standard SA) and where the acceptance rate is 1 (i.e. standard GA) with the GSSA for two population sizes: 5 and 50. The implementation of the GA operators is similar to that reported in [17], since it was used for a similar class of problems. Specifically, we used stochastic remainder and exponential fitting without elitism for the selection step, one-point cross-over and uniform mutation and cross-over probabilities.

We considered the optimization of a planar articulated bar structure with 49 elements (due to symmetry considerations the number of independent variables is 25), like those used to support the ceiling of industrial enclosures. The shape of the structure and its loads and boundary conditions appear in Figure 2. We used the following characteristics: Young modulus: $2{\cdot}1 \times 10^6\,\mathrm{kg/cm^2}$, Maximum allowable stress: $3500\,\mathrm{kg/cm^2}$, Maximum allowable displacement: 10 cm (for any point in the structure), Design regulation: American Institute of Steel Construction [18], Search space for the bar elements: 233 entries taken from the Altos Hornos de México S.A. catalog [19]. We considered in this case two degrees of freedom at each joint.

We studied the behaviour of different optimization methods for three increasingly severe design constraints: in the first case, only the maximum stress constraint is enforced; in the second, we also include the maximum displacement constraint and in the third, we also consider that the maximum design stress is reduced for elements subject to compression loads [18].

Since all these methods are stochastic, to obtain meaningful comparisons it is better to perform a number of independent runs (where different sequences of pseudo-random numbers are generated) and compare the average behaviour. In this case we performed 50 such runs.

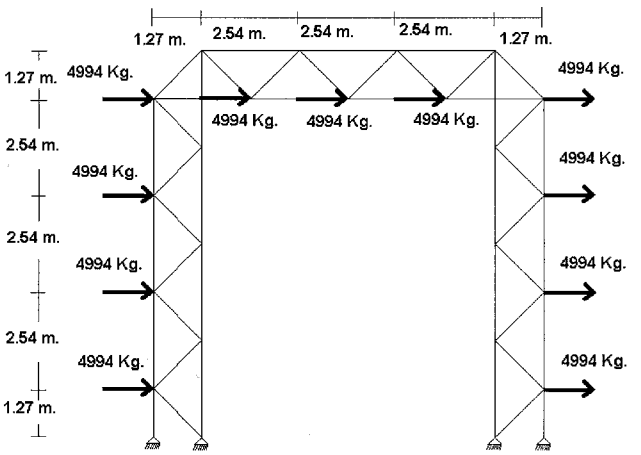

Figure 2. Shape, loads and boundary conditions of a planar bar structure used to support the ceiling of industrial enclosures

We compared the following methods:

SA: Standard Simulated Annealing with an exponential annealing schedule: $\beta(t) = \alpha^t$, with $\alpha = 1{\cdot}001$

GA50: A Genetic Algorithm with a population size of 50 individuals; cross-over probability of 80 per cent and mutation rate of 0·006/element. This is one of the many possible implementations of the GA operators that have been reported in the literature [11].

GSSA50: The General Stochastic Search Algorithm described in the previous section with a population size of 50; cross-over probability of 80 per cent; mutation rate of 0·04/element and exponential annealing schedule with $\alpha = 1{\cdot}01$.

GSSA5: Same as the previous case with a population of size 5 and $\alpha = 1{\cdot}001$.

The values for the parameters for each optimization method were adjusted by hand to obtain the best results in each case. To do this we performed a large number of trial runs. Some critical observations are the following:

1. The optimal value for the parameter α that controls the annealing schedule, depends on the population size, with smaller values for small populations. If one is uncertain about its value, it is better to use a small one (e.g. $\alpha = 1{\cdot}001$), since in this way one will obtain a good solution, even though the convergence may be slower.
2. The value of the cross-over probability is critical for the performance of the GA; for the GSSA, however, it has a much smaller influence, and it may even be set to zero without a significant impairment of its performance.
3. In the case of the GSSA, the mutation rate may be significantly higher than in the GA, because very bad mutations will be rejected anyway in the acceptance step.
4. Once the 'optimal' values for the parameters are found, they seem to work well for a variety of different problems [16]. We used the same values for all the examples reported here.

The results are summarized in Tables I–III: the first column in each table indicates the number of generations needed to reach a predefined target weight (650, 775 and 3000 kg, for cases 1, 2

Table I. Average results (over 50 Monte-Carlo runs) of different optimization methods for the structure of Figure 2—case 1

Method	Generations	Funct. ev.	Minimum weight
SA	13 500	13 500	627 kg – 250 000 gen.
GA50	4400	220 000	649 kg – 5000 gen.
GSSA50	1900	95 000	619 kg – 5000 gen.
GSSA5	3200	16 000	625 kg – 5000 gen.

Table II. Average results (over 50 Monte-Carlo runs) of different optimization methods for the structure of Figure 2—case 2

Method	Generations	Funct. ev.	Minimum weight
SA	13 700	13 700	737 kg – 250 000 gen.
GA50	N.R.	N.R.	817 kg – 5000 gen.
GSSA50	300	15 000	748 kg – 5000 gen.
GSSA5	800	4000	769 kg – 5000 gen.

Table III. Average results (over 50 Monte-Carlo runs) of different optimization methods for the structure of Figure 2—case 3

Method	Generations	Funct. ev.	Minimum weight
SA	5420	5420	2724 kg – 250 000 gen.
GA50	2000	100 000	2784 kg – 5000 gen.
GSSA50	600	30 000	2570 kg – 5000 gen.
GSSA5	2200	11 000	2716 kg – 5000 gen.

and 3, respectively), which was considered to be an acceptable result in each case; the second column contains the total number of function evaluations and the third column, the minimum weight obtained when each algorithm reached a stable condition (where no further improvements were achieved), and the number of generations needed. The convergence behaviour for case 1 is also illustrated in detail in Figure 3; this behaviour is qualitatively similar in all other cases.

As one can see, the inclusion of the acceptance operator allows for a significantly faster convergence rate, and also better final results in all cases.

We have also tried other variations, such as: adaptive mutation and cross-over probabilities; uniform cross-over; linear fitting and simple roulette selection; rebirth strategies as suggested in [17], special operators for controlling the diversity [16], and also a modern public domain GA software based on the GENESIS package [20]. The results in these cases are qualitatively similar to the ones reported here, in the sense that a significant improvement is always achieved by including the Metropolis acceptance operator.

3.2. Comparison with published results

The second set of experiments was designed to compare the performance of GSSA with other optimization methods that have been reported in the literature for the same type of problems. To

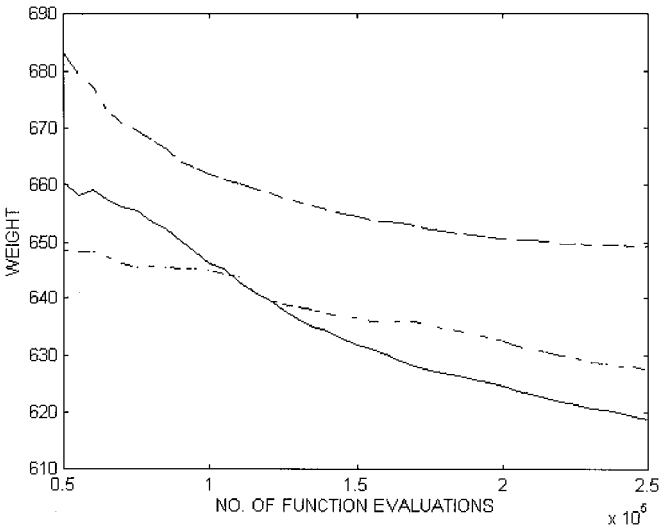

Figure 3. Convergence behavior of: a GA with population size 50 (dashed curve); SA (dot–dashed curve) and GSSA with population size 50 (solid line) for the optimization of the bar structure of Figure 2 (case 1)

do this comparison, we used the 10-bar structure depicted in Figure 4, which has been used as a standard problem by several authors, and for which there are a number of published results [21–25]. We considered in this case two degrees of freedom: the $x - y$ -direction displacements, at each joint. The vertical load is applied at joints 8 and 6 and equals 45 4540 kg, each point. The allowable stress is given as 1755 kg/cm^2, and the displacement constraint is 5·08 cm. at all joints in both the x and y directions. The Young's modulus of the material is 730 000 kg/cm^2.

We compared the following optimization schemes:

GSSA: The General Stochastic Search Algorithm (with a population of size 5, cross-over probability of 0 per cent; mutation rate of 0·10/element and exponential annealing schedule with $\alpha = 1{\cdot}001$), the total number of function evaluations are 5000.

VGA: The Variable string length Genetic Algorithm of Rajeev and Krishamoorthy [21] with population size 50 and the total number of function evaluations are 6050.

MC: The Monte-Carlo annealing algorithm of Elperin [22] with 60 000 function evaluations.

SAARSR: Simulated Annealing with Automatic Reduction of Search Range of Tzan and Pantelides [23] with 392 function calls.

ISA: The Iterate Simulated Annealing of Ackley [24], with 2504 function evaluation calls.

SSO: The State Space Optimal [25] with only 15 function evaluation calls.

The search space for the bar elements consists of 79 entries and is taken from the published literature [22]. The entries are:

$x_i = [0{\cdot}6425\,\text{cm}^2, 3{\cdot}226K\,\text{cm}^2, [K = 1, 2, \ldots, 76], 256{\cdot}85\,\text{cm}^2, 258{\cdot}08\,\text{cm}^2]$

The results for this example are shown in Tables IV–VI. The minimum structure weight that satisfies the stress and displacement constraints is obtained with the GSSA scheme. The MC scheme obtains a smaller weight, but these constrains are violated (see Tables V and VI).

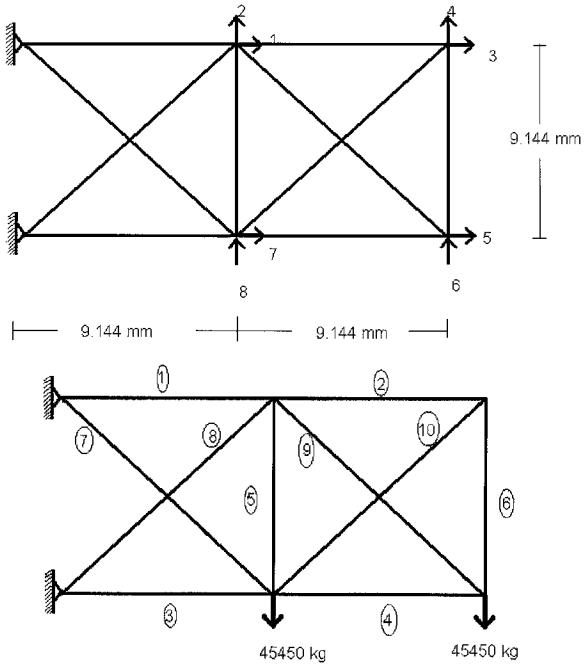

Figure 4. Shape, loads and boundary conditions of ten bar structure

Table IV. Cross-section in cm^2 for 10-bar structure of Figure 4

Element	GSSA	VGA	MC	SSO	ISA	SAARSR
1	205·17	206·46	200·01	193·75	269·48	201·35
2	0·6452	0·6452	0·6452	0·6452	79·810	0·6452
3	134·20	151·62	129·04	150·15	178·45	161·55
4	90·973	103·23	90·328	98·62	152·90	95·68
5	0·6452	0·6452	0·6452	0·6452	70·390	0·6452
6	0·6452	0·6452	0·6452	3·23	10·260	4·19
7	55·487	54·84	51·616	48·18	147·87	49·16
8	127·75	129·04	145·17	136·64	14·710	131·55
9	133·56	132·27	96·78	139·47	156·06	134·32
10	0·6452	0·6452	0·6452	0·6452	87·740	0·6452
Volume	805777	833258	765710	828956	1313131	833258
Weight	5982 kg	6186 kg	5685 kg	6155 kg	9750 kg	6187 kg

3.3. Optimization of real structures

For the next examples we considered the following material properties, design regulation and catalogue of steel sections: Young modulus: $2{\cdot}1\times10^6$ kg/cm^2, maximum allowable stress:3500 kg/cm^2, maximum allowable displacement: 10 cm (for any point in the structure), design regulation: American Institute of Steel Construction [18], Search space for the bar elements: 233 entries taken from

Table V. Stress in kg/cm^2 for ten bar structure of Figure 4

Element	GSSA	VGA	MC	SSO	ISA	SAARSR
1	−447·65	−444·75	−460·10	−475·31	−209·75	−476·58
2	0·41	3·41	−15·30	91·98	−111·35	43·99
3	670·31	593·43	695·72	597·46	449·90	569·04
4	499·60	440·30	503·06	461·46	239·13	485·80
5	−1464·09	−1428·68	**−1757·16**	−1754·88	362·13	−1641·04
6	0·41	3·41	−15·30	18·37	−866·13	14·83
7	−1134·31	−1148·24	−1214·48	−1299·10	−763·45	−1311·60
8	513·60	508·24	453·71	482·74	1064·60	528·83
9	−481·25	−485·97	−664·00	−461·46	−331·34	−492·79
10	−0·58	−4·82	21·64	−130·07	143·23	−65·61

Table VI. Displacements in cm of ten bar structure of Figure 4

Element	GSSA	VGA	MC	SSO	ISA	SAARSR
1	0·5602	0·5528	0·5954	0·4802	0·4022	0·5419
2	−5·0798	−4·9040	**−5·4352**	−4·9056	−3·8008	**−5·0889**
3	−1·4654	−1·2948	−1·5016	−1·3264	−0·8631	−1·3213
4	−5·0792	−4·8997	**−5·4543**	−4·8826	−4·8857	−5·0746
5	0·5607	0·5571	0·5763	0·5954	0·2627	0·5970
6	−1·8474	−1·8303	−1·7130	−1·8047	−2·9298	−1·9303
7	−0·8396	−0·7433	−0·8715	−0·7484	−0·5636	−0·7129
8	−3·6813	−3·6199	−3·9140	−4·0030	−2·4762	−3·9901

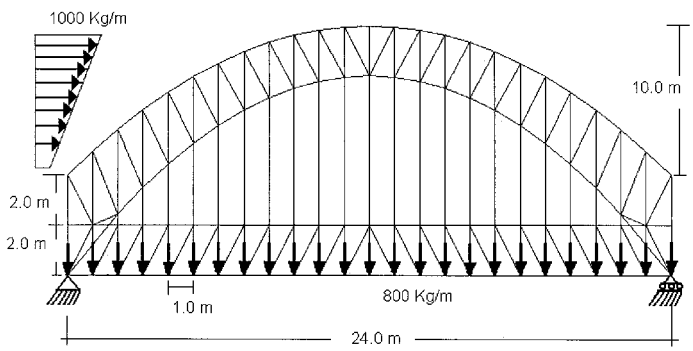

Figure 5. Shape, loads and boundary conditions of a pedestrian bridge structure

the Altos Hornos de México S.A. catalogue [19]. We considered in this case two degrees of freedom at each joint and the maximum design stress is reduced for elements subject to compression loads [18].

The first example of this section is a pedestrian bridge with 213 elements (96 independent variables due to symmetry constraints; Figure 5). The next example is a tall electric tower with

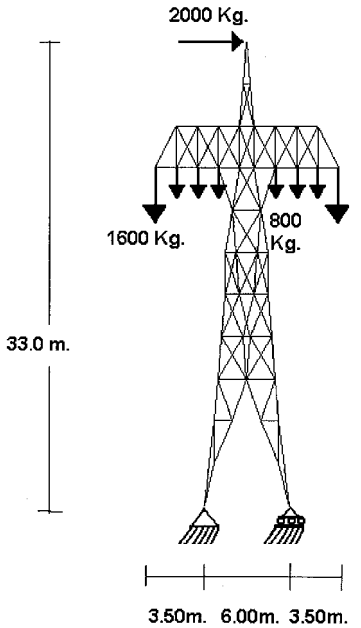

Figure 6. Shape, loads and boundary conditions of a tall electric tower

Table VII. Average results (over 50 Monte-Carlo runs) of different optimization methods for the bridge structure of Figure 5

Method	Generations	Funct. ev.	Minimum weight
SA	8700	8700	9147 kg – 20 000 gen.
GA50	2000	100 000	9370 kg – 5000 gen.
GSSA50	1500	75 000	8436 kg – 5000 gen.
GSSA5	3400	17 000	8620 kg – 5000 gen.

Table VIII. Average results (over 50 Monte-Carlo runs) of different optimization methods for the tower structure of Figure 6

Method	Generations	Funct. ev.	Minimum weight
SA	11 400	11 400	12 789 kg – 20 000 gen.
GA50	N.R	N.R	17 313 kg – 5000 gen.
GSSA50	1900	95 000	12 264 kg – 5000 gen.
GSSA5	4700	23 500	14 525 kg – 5000 gen.

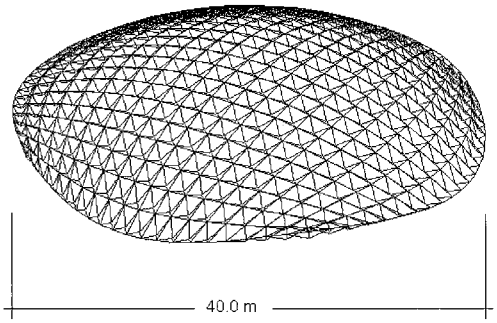

Figure 7. Shape of a tridimensional structure with 2440 elements

113 elements (59 independent variables; Figure 6). The parameters used in which scheme are the same reported in Section 3.1 The optimization results (with the same parameter values described above in Section 3.1) are summarized in Tables VII and VIII (the number of generations needed to reach a predefined target weight of 10 000 and 15 000 kg respectively).

The last example is the optimization of a three-dimensional structure (symmetric and simply supported in its perimeter) shown in Figure 7, using the GSSA. In this case we have 2440 bars, and the structure supports a load of 441 000 kg (1000 kg in each joint of the upper part). We considered in this case three degrees of freedom at each joint. The final weight of the structure is 451 380 kg obtained with 5000 evaluations of the cost function and a population of size 5.

4. CONCLUSIONS

We have presented a family of parallel stochastic search algorithms that includes as particular cases several popular schemes, such as GA and ESSA (with multiple starting points). It also includes a hybrid algorithm that combines parallel SA with selection.

We have applied this scheme to a number of structural optimization problems. From these experiments one may draw the following conclusions:

The best results are obtained for the hybrid case of Figure 1(c), where an acceptance operator is inserted before selection in the GA cycle. It should be emphasized that this idea is still applicable if the mutation operation includes other genetic operators, such as inversion. In fact, we believe that any GA implementation that preserves the identification of corresponding individuals before and after cross-over should improve its performance if an acceptance operator is included in this way.

The convergence rate of the GSSA algorithm improves as the population size increases; however, the total computational load increases as well. This means that the best results should be obtained in parallel machines where the number of processors equals the population size, so that one processor is assigned to each individual. We have performed experiments with an implementation of the algorithm in a Transputer board with four processors; in this implementation, we used a population of size 4, and each individual of the population was assigned to a specific processor, which was dedicated to the evaluation of the corresponding cost function at each iteration. At the end of each iteration these values were transmitted to the master processor (processor 0)

which performed the selection, mutation, cross-over and acceptance operations. The total computational time with this four processor machine was 0·27 of the computation time required by a single proccesor. This high efficiency is possible because for structural optimization problems of the kind we are considering, most of the time is spent evaluating the cost function, so that the number of function evaluations per processor is a good indicator of the total convergence time.

For serial machines, the best trade-off between solution quality and computational cost seems to be achieved by the GSSA with a small population size (e.g. $N \leqslant 5$).

ACKNOWLEDGEMENTS

The first, second and fourth authors were supported by grants from the Consejo Nacional de Ciencia y Tecnologia, Mexico.

REFERENCES

1. Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. *Science* 1983; **220**(4598):671–680.
2. Rechenberg I. *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution.* Fromman-Holzboog: Stuttgart, 1973.
3. Fogel LJ, Owens AJ, Walsh MJ. *Artificial Intelligence through Simulated Evolution.* Wiley: New York, 1966.
4. Goldberg DE. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley: Reading, MA, 1989.
5. Ackley DH. *A Connections Machine for Genetic Hillclimbing.* Kluwer Academic Publishers: Boston, 1987.
6. Boseniuk T, Ebling W. Boltzmann-, Darwin- and Haeckel-strategies in optimization problems. *Lecture Notes in Computer Science*: *Parallel Problem Solving from Nature* 1991; **496**:430–444.
7. Lin FT, Kao CY, Hsu CC. Incorporating genetic algorithms into simulated annealing. *Proceedings of the 4th International Symposium on Artificial Intelligence*, 1991:290–297.
8. Mahfoud SW, Goldberg DE. Parallel recombinative simulated annealing: a genetic algorithm. *Technical Report*, Department of Computer Science, University of Illinois, 1994.
9. Mahfoud SW. Niching methods for genetic algorithms. *Doctoral Dissertation*, University of Illinois, 1995.
10. Goldberg DE. A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Systems* 1990; **4**:445–460.
11. Keane AJ. Experiences with optimizers in structural design. *Adaptive Computing in Engineering Design and Control*, Plymouth, UK, September 1994.
12. de la Maza M, Tidor B. Boltzmann weighted selection improves performance of genetic algorithms. A.I. Memo 1345, Artificial Intelligence Lab., Massachusetts Institute of Technology, Cambridge, MA, 1991.
13. Srinivas M, Patnaik LL. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics* 1994; **24**(5):656–667.
14. Osaacson DL, Madsen RW. *Markov Chain Theory and Applications*. Wiley: New York, 1976.
15. Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E. Equations of state calculations by fast computing machines, *Journal of Chemical Physics* 1953; **21**(6):1087–1092.
16. Marroquin JL, Botello S, Horebeek JV. A Family of Parallel Stochastic Search Algorithms. *Comunicaciones del Cimat* 1996; **183**.
17. Galante M. Genetic Algorithms as an approach to optimize real–world trusses. *International Journal for Numerical Methods in Engineering* 1996; **39**:361–382.
18. AISC, *Load and Resistance Factor Design. Manual of Steel Construction* (1st edn), American Institute of Steel Construction Inc., U.S.A., 1986.
19. Altos Hornos de México SA. Base de Datos para el Manual de la Industria Siderurgica para la Construcción en Acero. AHMSA, 1991.
20. Schraudolph NN, Gefaustette II. A user guide to GAucsd. *Technical Report*, CSE Department, University of California, San Diego, 1996.
21. Rajeev S, Krishamoorthy CS. Genetic algorithms-based methodologies for design optimization of trusses. *Journal of Structural Engineering* 1997; **123**(3):350–358.
22. Elperin T. Monte-carlo structural optimization in discrete variables with annealing algorithm (1998). *International Journal for Numerical Methods in Engineering* 1988; **26**:815–821.

23. Tzan S, Pantelides CP. Annealing strategy for optimal structural design. *Journal of Structural Engineering* 1996; **122**(7):815–827.
24. Acckley DH. An empirical study of bit vector function optimization. In *Genetic Algorithms and Simulated Annealing*, Lawrence D (ed.) Norgan Kaufmann Publishers: Los Altos CA, 170–271.
25. Haug EI, Arora JS. *Applied Optimal Design Mechanical and Structural Systems*. Wiley: New York, 1979.
26. Goldberg DE. Simple genetic algorithms and the minimal deceptive problem. In *Genetic Algorithms and Simulated Annealing*, Davis L (ed.) Pitman: London, 1987:74–88.

