ELSEVIER

Comput. Methods Appl. Mech. Engrg. 156 (1998) 309–333

Computer methods in applied mechanics and engineering

Structural optimization using evolution strategies and neural networks

Manolis Papadrakakis*, Nikos D. Lagaros, Yiannis Tsompanakis

Institute of Structural Analysis & Seismic Research, National Technical University Athens, Zografou Campus, Athens 15773, Greece

Received 17 March 1997

Abstract

The objective of this paper is to investigate the efficiency of combinatorial optimization methods, in particular algorithms based on evolution strategies (ES) when incorporated into the solution of large-scale, continuous or discrete, structural optimization problems. Two types of applications have been investigated, namely shape and sizing structural optimization problems. Furthermore, a neural network (NN) model is used in order to replace the structural analysis phase and to compute the necessary data for the ES optimization procedure. The use of NN was motivated by the time-consuming repeated analyses required by ES during the optimization process. A back propagation algorithm is implemented for training the NN using data derived from selected analyses. The trained NN is then used to predict, within an acceptable accuracy, the values of the objective and constraint functions. The numerical tests presented demonstrate the computational advantages of the proposed approach which become more pronounced in large-scale optimization problems. © 1998 Elsevier Science S.A.

1. Introduction

Optimization of large-scale structures, such as sizing optimization of multi-storey 3-D frames or shape optimization of 2-D mechanical parts, is a computationally intensive task. In sizing optimization the aim is to minimize the weight of the structure under certain restrictions imposed by design codes, whereas in shape optimization the goal is to improve a given topology by minimizing an objective function subjected to certain constraints. When a gradient-based optimizer is used the most time-consuming part of the optimization process, in both cases, is devoted to the sensitivity analysis phase which is an important ingredient of all mathematical programming optimization methods [1]. On the other hand, the application of combinatorial optimization methods based on probabilistic searching, such as evolution strategies (ES), do not need gradient information and therefore avoid performing the computationally expensive sensitivity analysis step [2]. Furthermore, it is widely recognized that combinatorial optimization techniques, such as ES, are in general more robust and present a better global behaviour than mathematical programming methods. They may suffer, however, from a slow rate of convergence towards the global optimum.

During the last fifteen years there has been a growing interest in problem solving systems based on algorithms which rely on analogies to natural processes. The best known algorithms in this class include evolutionary programming (EP) [3], genetic algorithms (GA) [4,5], evolution strategies (ES) [6,7]. Evolution-based systems maintain a population of potential solutions. These systems have some selection process based on fitness of individuals and some recombination operators. ES like GA imitate biological evolution and combine the concept of artificial survival of the fittest with evolutionary operators to form a robust search mechanism.

* Corresponding Author.

0045-7825/98/$19.00 © 1998 Elsevier Science S.A. All rights reserved
PII S0045-7825(97)00215-6

Another important technique that follows natural processes, and in particular human brain functions, is artificial neural networks which simulate the structure of the biological neural network of the human brain. The use of artificial intelligence techniques, such as neural networks, to predict analysis outputs has been studied previously in the context of optimal design of structural systems [8,9], and also in some other areas of structural engineering applications. In the review papers of Berrais [10] and Waszczyszyn [11] a number of references can be found on the application of neural networks (NN) in computational mechanics. The principal advantage of a properly trained NN is that it requires a trivial computational effort to produce an acceptable approximate solution. Such approximations appear to be valuable in situations where the actual response computations are intensive in terms of computing time and a quick estimation is required.

In this work the efficiency of ES combined with neural networks (NN) both in sizing and shape structural optimization problems is investigated in an effort to increase the robustness as well as the computational efficiency of the optimization procedure. The use of NN was motivated by the time-consuming repeated analyses required for ES during the optimization process. The suitability of NN predictions is investigated in a number of structural problems optimized using ES and the computational advantages of the proposed approach are demonstrated. In addition, a thorough investigation is performed on the selection of the training set used for the NN learning procedure in order to ensure the generality and robustness of the proposed methodology.

For each problem an NN is trained utilizing information generated from a number of properly selected analyses. The data from these analyses are processed in order to obtain the necessary input and output pairs which are subsequently used to produce a trained NN. The trained NN is then used to predict the response of the structure in terms of objective and constraints function values due to different sets of design variables. The predicted values of the optimization functions should resemble closely to the corresponding values of the conventional analyses, which are considered exact. The NN type considered here is based on the feed-forward error back-propagation training algorithm [12]. It appears that the use of a properly selected and trained NN can eliminate any limitation on the dimensionality of the problem, due to a drastic reduction of the computing time required for the repeated analyses. Additionally, the cost of each optimization cycle of ES using NN is trivial, therefore the number of analyses during the optimization procedure is not of great importance. In other words, the proposed combination of NN with ES increases the applicability of ES, or any other probabilistic searching optimization algorithm, and makes it an efficient general purpose optimizer.

2. Shape optimization

In structural shape optimization problems the aim is to improve a given topology by minimizing an objective function subjected to certain constraints [2,13,14]. All functions are related to the design variables which are some of the coordinates of the key points in the boundary of the structure. The shape optimization method used in the present study is based on a previous work by Hinton and Sienz [13] for treating two-dimensional problems. It consists of the following essential ingredients: (i) shape generation and control, (ii) mesh generation, (iii) adaptive finite element analysis, (iv) sensitivity analysis, when gradient-based optimization methods are applied, and (v) shape optimization. The mathematical formulation of a typical structural shape optimization problem with respect to the design variables, the objective and constraint functions can be expressed in standard mathematical terms as a nonlinear programming problem (NLP) as follows:

$$\begin{aligned} &\min && F(s) \\ &\text{subject to} && g_j(s) \leqslant 0 \qquad j = 1, \ldots, m \\ &&& s_i^{\mathrm{l}} \leqslant s_i \leqslant s_i^{\mathrm{u}} \quad i = 1, \ldots, n \end{aligned} \tag{1}$$

where s is the vector of design variables, $F(s)$ is the objective function to be minimized (or maximized), $g_j(s)$ are the behavioural constraints, s_i^{l} and s_i^{u} are the lower and the upper bounds on a typical design variable s_i. Equality constraints are usually rarely imposed. Whenever they are used they are treated for simplicity as a set of two inequality constraints.

The set of design variables gives a unique definition of a particular design. The selection of design variables is very important in the optimization process. The designer has to decide a priori where to allow design changes

and to evaluate how these changes should take place by defining the location of the design variables and the moving directions. The use of the coordinates at key points of the curves that define the shape of the structural model as design variables leads to fewer design variables and more freedom in controlling the shape of the structure.

It is an issue of extreme importance to formulate the optimization problem correctly, otherwise unrealistic solutions may be found. Normally, it is necessary to constrain some function of the stresses (e.g. the principal stress) so that it will not exceed a specified value throughout the entire structure. From a practical point of view a finite number of so-called 'stress constraint points' is selected, where the condition is enforceable. These points are either some predefined points within the domain, or some boundary nodes. Other types of constraints, like displacement or frequency constraints, can also be imposed depending on the type of problem. Usually, the constraint functions and their derivatives are normalized in order to improve the performance of the optimizer.

The shape optimization methodology proceeds with the following steps: (1) At the outset of the optimization, the geometry of the structure under investigation has to be defined. The boundaries of the structure are modeled using cubic B-splines which, in turn, are defined by a set of key points. Some of the coordinates of these key points will be the design variables which may or may not be independent of each other. (2) An automatic mesh generator is used to create a valid and complete finite element model. A finite element analysis is then carried out and the displacements and stresses are evaluated. In order to increase the accuracy of the analysis an h-type adaptivity analysis may be incorporated at this stage. (3) If a gradient-based optimizer, like the sequential quadratic programming SQP algorithms, is used then the sensitivities of the constraints and the objective function to small changes of the design variables are computed either with the finite difference, or with the semi-analytical method. (4) The design variables are being optimized. If the convergence criteria for the optimization algorithm are satisfied, then the optimum solution has been found and the process is terminated, else a new geometry is defined and the whole process is repeated from step 2. The whole procedure is depicted in Fig. 1.

3. Sizing optimization

In sizing optimization problems the aim is usually to minimize the weight of the structure under certain behavioural constraints on stress and displacements. The design variables are most frequently chosen to be dimensions of the cross-sectional areas of the members of the structure. Due to engineering practice demands the members are divided into groups having the same design variables. This linking of elements results in a trade-off between the use of more material and the need for symmetry and uniformity of structures due to practical considerations. Furthermore, it has to be taken into account that due to fabrication limitations the design variables are not continuous but discrete since cross-sections belong to a certain set.

A discrete structural optimization problem can be formulated in the following form:

$$\begin{aligned} &\min && F(s) \\ &\text{subject to} && g_j(s) \leq 0 \quad j = 1, \ldots, m \\ &&& s_i \in R^d, \quad i = 1, \ldots, n \end{aligned} \tag{2}$$

where R^d is a given set of discrete values and design variables s_i $(i = 1, \ldots, n)$ can take values only from this set. In the present study the sizing optimization of multi-storey 3-D frames is investigated. Optimal designs of frames have been studied initially using conventional plastic design methods. Then more sophisticated optimization algorithms were introduced in order to solve this type of problem more efficiently [15,16]. Most frequently the objective function is the weight of the structure and the constraints are the member stresses and nodal displacements or inter-storey drifts. For rigid frames in rolled W-shapes, under allowable stress design requirements specified by AISC [17], the stress constraints are the non-dimensional ratio q of the following formulas:

$$q = \frac{f_a}{F_a} + \frac{f_b}{F_b} \leq 1.0 \quad \text{if } \frac{f_a}{F_a} \leq 0.15 \tag{3}$$

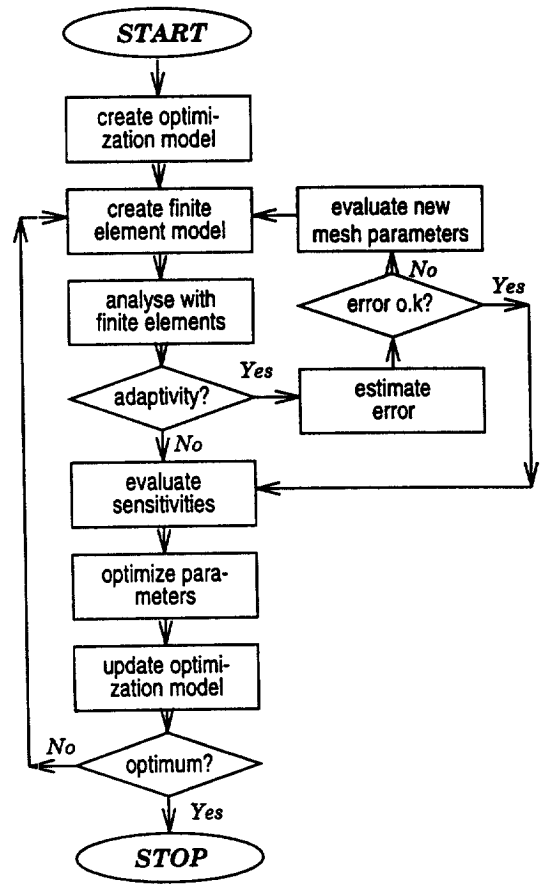


Fig. 1. Structural shape optimization algorithm.

and

$$q = \frac{f_a}{F_a} + \frac{C_m f_b}{(1 - f_a/F'_e)F_b} \leq 1.0 \quad \text{if } \frac{f_a}{F_a} > 0.15 \tag{4}$$

$f_a = (P/A)$ is the computed axial stress, where A is the cross-sectional area of the member. The computed bending stress is taken from $f_b = (M_{\max}/W)$, where $M_{\max}$ is the maximum bending moment in the element and W is the section modulus, which is equal to $W = 0.256 * A^{3/2}$ for rolled W-shaped cross-sections. Finally, F'_e is the Euler stress divided by a safety factor, C_m is a coefficient depending upon column curvature, $F_a = 0.60 * F_y$ is the allowable axial stress, $F_b = 0.66 * F_y$ is the allowable bending stress and F_y is the yield stress.

4. Evolution strategies (ES)

There are three types of algorithms belonging to the class of evolutionary computation that imitate nature by using biological methodologies in order to find the optimum solution of a problem: (i) evolutionary programming (EP), (ii) genetic algorithms (GA) and (iii) evolution strategies (ES). Their main difference is that GA deal with bit-strings of fixed sizes, ES with real vectors and EP with finite state automata. GA basic assumption is that the optimal solution can be found by assembling building blocks, i.e. partial pieces of solutions, while ES and EP simply ensure the emergence of the best solutions. The most important consequence of this different approach is related to the recombination operator, viewed as essential for GA, as potentially useful for ES and as possibly harmful for EP. The modern tendencies seem to follow combinations of the two

approaches, since GA users have turned to real number representations when dealing with real numbers following experimental results or heuristic demonstrations, whereas ES users have included recombination as a standard operator, and have designed special operators for non real-valued problems [18].

Evolution strategies were proposed for parameter optimization problems in the 1970s by Rechenberg [6] and Schwefel [7]. Similar to genetic algorithms, ES imitate biological evolution in nature and have three characteristics that make them differ from other conventional optimization algorithms: (i) in place of the usual deterministic operators, they use randomized operators: mutation, selection as well as recombination; (ii) instead of a single design point, they work simultaneously with a population of design points in the space of variables; (iii) they can handle continuous, discrete and mixed optimization problems. The second characteristic allows for a natural implementation of ES on parallel computing environments. The ES, however, achieve a high rate of convergence than GA due to their self-adaptation search mechanism and are considered more efficient for solving real world problems [19]. The ES were initially applied for continuous optimization problems, but recently they have also been implemented in discrete and mixed optimization problems. The ES algorithms used in the present study are based on the work of Thierauf and Cai who applied the ES methodologies in sizing structural optimization problems having discrete and/or continuous design variables [20,21].

4.1. ES algorithms for continuous problems

The ES for continuous optimization problems are divided into two-membered (2-ES) and multi-membered algorithms (M-ES).

4.1.1. The two-membered ES

The earliest evolution strategies were based on a population consisting of one individual only. The two-membered scheme is the minimal concept for an imitation of organic evolution. The two principles of mutation and selection, which Darwin in 1859 recognized to be most important, are taken as rules for variation of the parameters and for recursion of the iteration sequence, respectively.

The two-membered ES works in two steps for the solution of the optimization problem:

Step 1 (mutation). The parent $s_p^{(g)}$ of the generation g produces an offspring $s_0^{(g)}$, whose genotype is slightly different from that of the parent:

$$s_0^{(g)} = s_p^{(g)} + z^{(g)} \tag{5}$$

where $z^{(g)} = [z_1^{(g)}, z_2^{(g)}, \ldots, z_n^{(g)}]^{\mathrm{T}}$ is a random vector.

Step 2 (selection). The selection chooses the best individual between the parent and the offspring to survive:

$$s_p^{(g+1)} = \begin{cases} s_0^{(g)} & \text{if } g_i(s_0^{(g)}) \leqslant 0 \ (i = 1, 2, \ldots, 1) \text{ and } f(s_0^{(g)}) \leqslant f(s_p^{(g)}) \\ s_p^{(g)} & \text{otherwise} \end{cases} \tag{6}$$

The question that arises is how to choose the random vector $z^{(g)}$ in Step 1. This choice has the role of mutation.

Mutation is understood to be random, purposeless events, which occur very rarely. If one interprets them, as a set of many individual events the 'natural' choice is to use a probability distribution according to which small changes occur frequently, but large ones only rarely. As a result of this assumption two requirements arise together by analogy with natural evolution: (i) the expected mean value ξ_i for a component $z_i^{(g)}$ to be zero; (ii) the variance σ_i^2, the average squared standard deviation from mean value, to be small. The role of mutation operation, both in ES and GA, is to make the population less homogeneous and permit a diversity in the process of random search which results in a better exploration of the design space and a faster allocation of the optimum design.

The probability density function for normally distributed random events is given by

$$p(z_i^{(g)}) = \frac{1}{\sqrt{(2\pi)\sigma_i}} \exp\left(-\frac{(z_i^{(g)} - \xi_i)^2}{2\sigma_i^2} \right) \tag{7}$$

When $\xi_i = 0$ the so-called $(0, \sigma_i)$ normal distribution is obtained. By analogy with other deterministic search strategies, σ_i can be called step length, in the sense that it represents average values of the length of the random steps.

If the step length is too small the search takes an unnecessarily large number of iterations. On the other hand, if the step length is too large the optimum can only be crudely approached and the search can even get stuck far away from the global optimum. Thus, as in all optimization strategies, the step length control is the most important part of the algorithm after the recursion formula, and it is further more closely linked to the convergence behaviour.

The standard deviation σ_i can be adjusted during the optimum design search as follows (Rechenberg's 1/5 success rule [6]): *'The ratio ψ of successful mutations to all mutations should be 1/5. If it is greater, increase the standard deviations σ_i; while if it is less, decrease the standard deviations σ_i'*. Successful mutations are those who produced offsprings having a better objective function value than their parents. According to Schwefel [7] the check should take place every n mutations over the preceding $10*n$ mutations, while the increase and decrease factors of the step length should be $(1/0.85)$ and 0.85, respectively. The intuitive reason behind this rule is to increase the efficiency of the optimum design search by monitoring the step lengths. If ψ is greater than 1/5 the search continues with larger steps else the steps are shorter.

4.1.2. Multi-membered ES

The disadvantage of the 1/5 success rule for controlling the 'step size' of the two-membered strategies, as well as the need for acceleration of the convergence rate of the methodology, led to the development of multi-membered evolution strategies (M-ES). The M-ES differ from the (2-ES) in the size of the population and the recombination and mutation operators. In this case a population of μ parents will produce λ offsprings. Thus, the two steps of the ES are defined as follows:

Step 1 (recombination and mutation). The population of μ parents at gth generation produces λ offsprings. The genotype of any descendant differs only slightly from that of its parents. For every offspring vector a temporary parent vector $\tilde{s} = [\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n]^T$ is first built by means of recombination. For continuous problem the following recombination cases can be used:

$$\tilde{s}_i = \begin{cases} s_{\alpha,i} \text{ or } s_{b,i} \text{ randomly} & \text{(A)} \\ 1/2(s_{\alpha,i} + s_{b,i}) & \text{(B)} \\ s_{bj,i} & \text{(C)} \\ s_{\alpha,i} \text{ or } s_{bj,i} \text{ randomly} & \text{(D)} \\ 1/2(s_{\alpha,i} + s_{bj,i}) & \text{(E)} \end{cases} \tag{8}$$

$\tilde{s}_i$ is the ith component of the temporary parent vector $\tilde{s}$, $s_{\alpha,i}$ and $s_{b,i}$ are the ith components of the vectors s_a and s_b which are two parent vectors randomly chosen from the population. In case C of Eq. (8), $\tilde{s}_i = s_{bj,i}$ means that the ith component of $\tilde{s}$ is chosen randomly from the ith components of all μ parent vectors. From the temporary parent $\tilde{s}$ an offspring can be created following the mutation operator of 2-ES (Eq. (5)).

Step 2 (selection). There are two different types of the multi-membered ES:

$(\mu + \lambda)$-ES: The best μ individuals are selected from a temporary population of $(\mu + \lambda)$ individuals to form the parents of the next generation.

(μ, λ)-ES: The μ individuals produces λ offsprings $(\mu \leq \lambda)$ and the selection process defines a new population of μ individuals from the set of λ offsprings only.

In the second type, the life of each individual is limited to one generation. This allows the (μ, λ)-ES selection to perform better on dynamic problems where the optimum is not fixed, or on problems where the objective function is noisy.

Multi-membered ES termination criteria are the following: (i) when the absolute or relative difference between the best and the worst objective function values of the current generation is less than a given value ε_1, or (ii) when the mean value of the objective values from all parent vectors in the last $2*n$ generations has not been improved by less than a given value ε_2.

4.2. ES for discrete optimization problems

In engineering practice the design variables are not continuous because usually the structural parts are constructed with certain variation of their dimensions. Thus, design variables can only take values from a predefined discrete set. For the solution of discrete optimization problems a modified ES algorithm has been proposed by Thierauf and Cai [20]. The basic differences between discrete and continuous ES are focused on the mutation and the recombination operators.

The mutation operator in the continuous version of ES produces a normally distributed random change vector $z^{(g)}$. Each component of this vector has small standard deviation value σ_i and zero mean value. As a result of this there is a possibility that all components of a parent vector may be changed, but usually the changes are small. In the discrete version of ES the random vector $z^{(g)}$ is properly generated in order to force the offspring vector to move to another set of discrete values. The fact that the difference between any two adjacent values can be relatively large is against the requirement that the variance σ_i^2 should be small. For this reason it is suggested that not all the components of a parent vector, but only a few of them (e.g. l) should be randomly changed in every generation. This means that $n - l$ components of the randomly changed vector $z^{(g)}$ will have zero value. In other words, the terms of vector $z^{(g)}$ are derived from

$$z_i^{(g)} = \begin{cases} (\kappa + 1)\,\delta s_i & \text{for } l \text{ randomly chosen components} \\ 0 & \text{for } n - l \text{ other components} \end{cases} \tag{9}$$

where δs_i is the difference between two adjacent values in the discrete set and κ is a random integer number which follows the Poisson distribution

$$p(\kappa) = \frac{(\gamma)^\kappa}{\kappa!}\,\mathrm{e}^{-\gamma} \tag{10}$$

γ is the standard deviation as well as the mean value of the random number κ. For a very small γ (e.g. 0.001) the probability that κ will be zero is greater than 99%. For greater values of γ (e.g. 0.05) the probability that κ will be zero is 95% and the probability that it will be one is 5%. This shows how the random change $z_i^{(g)}$ is controlled by the parameter γ. The choice of l depends on the size of the problem and it is usually taken as the 1/5 of the total number of design variables. The l components are selected using uniform random distribution in every generation according to Eq. (9).

As far as the recombination operator is concerned, in the case of discrete optimization problems the recombination cases (B) and (E) in Eq. (8) are not suitable since the mean value of two discrete values is usually not a discrete value from the given set of the design variables. Therefore, the following recombination cases are employed

$$\tilde{s}_i = \begin{cases} s_{\alpha,i} \text{ or } s_{b,i} \text{ randomly} & \text{(A)} \\ s_{m,i} \text{ or } s_{b,i} \text{ randomly} & \text{(B)} \\ s_{bj,i} & \text{(C)} \\ s_{\alpha,i} \text{ or } s_{bj,i} \text{ randomly} & \text{(D)} \\ s_{m,i} \text{ or } s_{bj,i} \text{ randomly} & \text{(E)} \end{cases} \tag{11}$$

The recombination cases (A), (C) and (D) are the same as those in Eq. (8). In cases (B) and (E) of Eq. (11) the current vector of design variables s_m is not randomly chosen but is the best of the μ parent vectors in the gth generation. It has been verified that when the best parent is used during the recombination a better convergence rate is achieved. Using the temporary parent vector of Eq. (11) an offspring can be created as in the two-membered type ES (Eq. (6)).

For discrete optimization the procedure terminates when one of the following termination criteria is satisfied: (i) when the best value of the objective function in the last $4 * n * \mu/\lambda$ generations remains unchanged, (ii) when the mean value of the objective values from all parent vectors in the last $2 * n * \mu/\lambda$ generations has not been improved by less than a given value ε_b (=0.0001), (iii) when the relative difference between the best objective function value and the mean value of the objective function values from all parent vectors in the current generation is less than a given value ε_c (=0.0001), (iv) when the ratio μ_b/μ has reached a given value

ε_d (=0.5–0.8) where μ_b is the number of the parent vectors in the current generation with the best objective function value.

4.3. ES in structural optimization problems

So far comparatively little effort has been spent in applying probabilistic search methods to structural optimization problems [1,22]. Usually these types of problems are solved with a mathematical programming algorithm such as the sequential quadratic programming method SQP [23,24], the generalized reduced gradient method (GrG) [25], the method of moving asymptotes (MMA) [26], which need gradient information. In structural optimization problems, where the objective function and the constraints are particularly highly nonlinear functions of the design variables, the computational effort spent in gradient calculations is usually large.

In a recent work by Papadrakakis et al. [1] it was found that combinatorial type algorithms are computationally efficient even if a greater number of analyses are needed to reach the optimum. These analyses are computationally less expensive than in the case of mathematical programming algorithms since they do not need gradient information. Furthermore, probabilistic methodologies, due to their random search, are considered as global optimization methods because they are capable of finding the global optimum, whereas mathematical programming algorithms may be trapped in local optima. Finally, the natural parallelism inherent in combinatorial algorithms makes them very attractive for application in parallel computer architectures.

The implementation of ES in structural optimization follows the same steps as described in Fig. 1 without performing the sensitivity analysis step. The ES optimization procedure starts with a set of parent vectors. If any of these parent vectors gives an infeasible design then this parent vector is modified until it becomes feasible. Subsequently, the offsprings are generated and checked if they are in the feasible region. According to $(\mu + \lambda)$ selection scheme in every generation the values of the objective function of the parent and the offspring vectors are compared and the worst vectors are rejected, while the remaining ones are considered to be the parent vectors of the new generation. On the other hand, according to (μ, λ) selection scheme only the offspring vectors of each generation are used to produce the new generation. This procedure is repeated until the chosen termination criterion is satisfied.

The computational efficiency of the multi-membered ES discussed in this work is affected by the number of parents and offsprings involved. It has been observed that values of μ and λ equal to the number of the design variables produce best results [1]. The ES algorithm for structural optimization applications can be stated as follows:

1. *Selection step:* selection of s_i $(i = 1, 2, \ldots, \mu)$ parent vectors of the design variables
2. *Analysis step:* solve $K(s_i)u_i = f$ $(i = 1, 2, \ldots, \mu)$
3. *Constraints check:* all parent vectors become feasible
4. *Offspring generation:* generate s_j $(j = 1, 2, \ldots, \lambda)$ offspring vectors of the design variables
5. *Analysis step:* solve $K(s_j)u_j = f$ $(j = 1, 2, \ldots, \lambda)$
6. *Constraints check:* if satisfied continue, else change s_j and go to step 4
7. *Selection step:* selection of the next generation parents according to $(\mu + \lambda)$ or (μ, λ) selection schemes
8. *Convergence check:* if satisfied stop, else go to step 3

4. Elements of artificial neural networks theory

Only the basic principals of neural networks (NN) will be discussed in this section. A more detailed introduction to NN may be found in [12,27]. A number of computational structures technology applications have been investigated in order to demonstrate neural network capabilities [28–30]. NN have been also applied to a variety of problems like image processing, pattern recognition, classification, medical and financial applications etc. NN are most suitable for applications that have any of the following characteristics: (i) they are data-intensive and depend on multiple criteria, (ii) data set is incomplete, noisy, and contain errors, (iii) there are many results about the problem area, and (iv) there is no way to describe the problem with a function.

The aim of the present study is to train a neural network to provide computationally inexpensive estimates of

analysis outputs required during the optimization process. A trained network presents some distinct advantages over the numerical computing paradigm. It provides a rapid mapping of a given input into the desired output quantities, thereby enhancing the efficiency of the redesign process. This major advantage of a trained NN over the conventional procedure, under the provision that the predicted results fall within acceptable tolerances, leads to results that can be produced in a few clock cycles, representing orders of magnitude less computational effort than the conventional computational process.

4.1. Back propagation learning algorithm

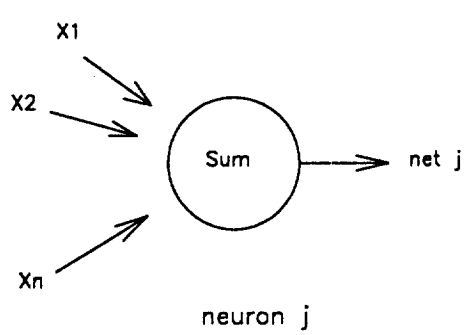
In general there are many different types of NN topologies and usually there is not one architecture that is suitable for all problems. The main types of NN are the following: competitive learning, the Boltzmann machine, the Hopfield network, and the back propagation network. The latter type is the most popular due to its simplicity and ease of use. Its name comes from the way it 'learns': by back propagating the errors that occur during the training process. The basic model for a neural network processing element is shown in Fig. 2.

A back propagation (BP) neural network consists of multiple interconnected processing elements belonging to different layers. In a BP algorithm learning is carried out using a set of input training patterns propagated through a network consisting of an input layer, one or more hidden layers and an output layer as shown in Fig. 3. The hidden layers represent complicated associations between patterns and propagate data in a feed-forward manner from the input towards the output layer. The use of hidden layers and nonlinear activation functions enhance the ability of NN to 'learn' nonlinear relationships among I/O quantities. The selection of the number of nodes in the hidden layer(s) is an important factor in the design of the network. The number of nodes in the hidden layer(s) is usually selected as the mean value of the number of the input and output nodes plus the input nodes. More sophisticated networks use dynamic node pruning or node growing in the intermediate layer(s).

Each layer of the network has its corresponding units (processing elements, neurons or nodes) and weight connections that perform the learning process. Most of the NN training methods use gradient descent algorithms, such as least squares, in order to correct the values of the weight connections. This is the outcome of a minimization of an error measure which is the difference between the computed and the desired output values summed over the output units and all pairs of I/O vectors. This correction step of the weights is known as *delta rule*. Once a trained network is established then every time it is activated with a new input data set it produces an estimation of the desired output. In an optimization problem using NN, for example, one could create a mapping between design variables and values of the objective and constraint(s) functions.

The inputs $x_i, i = 1, \ldots, n$, received by the input layer are analogous to the electrochemical signals received by neurons in human brain. In the simplest model these input signals are multiplied by connection weights $w_{p,ij}$ and the effective output $\text{net}_{p,j}$ of the elements in the next layer p is the weighted sum of the inputs

$$\text{net}_{p,j} = \sum_{i=1}^{n} w_{p,ij}\, \text{net}_{q,i} \tag{12}$$



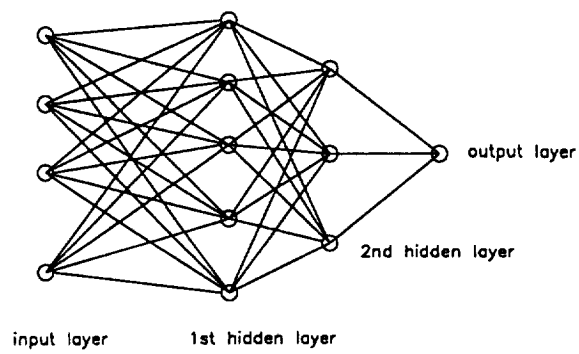


Fig. 2. Basic model for a processing element.

Fig. 3. A fully connected NN configuration.

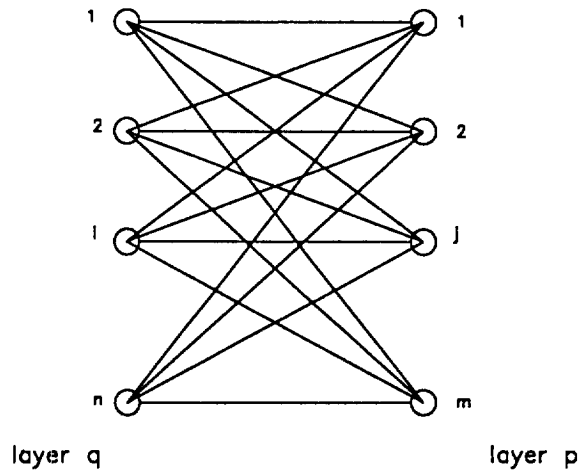


Fig. 4. Connection pattern between two layers.

As shown in Fig. 4, $w_{p,ij}$ is the connecting weight of the i neuron in the q (source) layer and the j neuron in the p (target) layer, $\text{net}_{q,i}$ is the input signal transmitted from the i neuron of the layer q to the nodes of the target layer p, $\text{net}_{p,j}$ is the output produced at the j neuron of the layer p. The exterior inputs x_i correspond to $\text{net}_{q,i}$ for the input layer.

In the biological system, a typical neuron produces an output signal only if the incoming signal is strong enough to stimulate the neuron. This output is simulated with NN by

$$\text{out}_{p,j} = F(\text{net}_{p,j}) \tag{13}$$

where F is an activation function which produces the output at the j neuron of the p layer. The activation function used in the present study is the commonly used sigmoid function

$$F(\text{net}_{p,j}) = \frac{1}{1 + \text{e}^{-(\text{net}_{p,j} + b_{p,j})}} \tag{14}$$

where $b_{p,j}$ is a bias parameter which acts as a function shifting term that improves the overall network accuracy. Bias parameters can be learned during the training in the same manner as the other weights. The principal advantage of the sigmoid function is its ability to handle both large and small input signals. Initially the weights and biases are selected randomly and during the network learning process their values are improved.

At the output layer the computed output(s), also called observed output(s), are subtracted from the desired or target output(s) to give the error signal

$$\text{err}_{k,i} = \text{tar}_{k,i} - \text{out}_{k,i} \tag{15}$$

where $\text{tar}_{k,i}$ and $\text{out}_{k,i}$ are the target and the observed output(s) for the node i of the output layer k, respectively. This type of training is called supervised learning.

For the evaluation of the weight changes in the output layer related to the error signal given by Eq. (15) the following gradient descent relationship is used

$$\Delta w_{k,ji} = \eta \cdot \delta_{k,i} \cdot \text{out}_{p,j} \tag{16}$$

where η denotes a learning rate coefficient usually selected between 0.01 and 0.9 and $\text{out}_{p,j}$ denotes the output of the last hidden layer p. This learning rate coefficient is analogous to the step size parameter in the numerical optimization algorithms.

The term $\delta_{k,i}$ is the result of the multiplication of the derivative of the activation function, for the neuron in question, with the error signal

$$\delta_{k,i} = \text{d}F(\text{net}_{k,i}) \cdot \text{err}_{k,i} \tag{17}$$

while the derivative dF of the sigmoid function is given by

$$dF(\text{net}_{k,i}) = \text{out}_{k,i} \cdot (1 - \text{out}_{k,i}) \tag{18}$$

Alternatively, the changes of the weights may be expressed with a generalized delta rule according to [15] by

$$\Delta w_{k,ji}^{t+1} = \eta \cdot \delta_{k,i} \cdot \text{out}_{p,j} + \alpha \cdot \Delta w_{k,ji}^{t} \tag{19}$$

where the superscript t denotes the cycle of the weight modification and α is the momentum term which controls the influence of the previous weight changes. The momentum term is used in order to accelerate and at the same time to minimize the oscillations of the learning process of the network. For the hidden layers the corresponding weight changes are given by

$$\delta_{p,j} = dF(\text{net}_{p,j}) \cdot \left(\sum_{i=1}^{n} \delta_{k,i} \cdot w_{k,ji} \right) \tag{20}$$

$$\Delta w_{p,lj}^{t+1} = \eta \cdot \delta_{p,j} \cdot \text{out}_{r,l} + \alpha \cdot \Delta w_{p,lj}^{t} \tag{21}$$

where $\text{out}_{r,l}$ denotes the output of the neuron l of the hidden layer r and $\Delta w_{p,lj}^{t}$ is the weight changes between neuron l of the hidden layer r to neuron j of the hidden layer p. After the evaluation of the weight changes the updated values of the weights, $w_p^{t+1} = w_p^t + \Delta w_p^{t+1}$ are used for the next training cycle. This process has to be repeated for all input training patterns until the desired level of error is obtained.

There are two categories of NN training algorithms depending on the way they treat the I/O data set. Algorithms that compute the error measure and gradient from the entire data set are called *batch* or *off-line* training methods. Algorithms that compute the error and gradient and update the weights for each set or groups of sets, are called *incremental* or *on-line* training methods. On-line methods may be unstable, since the error measure for the entire data set is unknown and do not provide a criterion of training termination [31]. In this study the off-line method is employed with a single pattern training where all the weights are updated before the next training pattern is processed.

4.2. The NN training

In the present implementation the objective is to investigate the ability of the NN to predict accurate structural analysis outputs that are necessary for the ES optimizer. This is achieved with a proper training of the NN. The NN training comprises the following tasks: (i) select the proper training set, (ii) find a suitable network architecture and (iii) determine the appropriate values of characteristic parameters such as the learning rate and momentum term.

The main limitation of a NN training algorithm is the fact that its efficiency depends on the learning rate, momentum term and network architecture. Unfortunately, there are not any general rules on the selection of these parameters. They usually depend on the type of the problem and the experience of the designer and sometimes they can be chosen with a trial and error approach. The appropriate selection of I/O training data is one of the most important features of NN training. The number and the distribution of training patterns is of great importance especially for the BP algorithm which provides good results only if the training set includes data over the entire range of the design space.

The output of the sigmoid function used in the BP algorithm lies between 0 and 1. Thus, in order to produce meaningful results using Eq. (15), the output values of the training patterns should be normalized within this range. During the training phase the weights can be adjusted to very large values, which can force all or most of the neurons to operate with large output values in a region where the derivative of the activation function is very small. Since the correction of the weights depends on the derivative of the sigmoid function the network in this case may become virtually standstill. Initializing the weights to small random values could help to avoid this situation, although a more appropriate one is to normalize the input patterns to lie between 0 and 1.

Typically there are two types of network architectures: fully and patterned connected networks. In a fully connected network, as shown in Fig. 3, each unit in a layer is connected to all the units of the previous and the next layer. This type of network architecture is widely used. Alternatively, partial associativity between the units may be created reducing the number of connections and producing a patterned connected network. The number

of neurons to be used in the hidden layers is not known in advance and usually is estimated by a trial and error procedure. At the first phase of learning it is convenient to increase the number of hidden units gradually and when the desired convergence is achieved some of them are removed in order to find the minimal size of the network [32].

The learning rate coefficient and the momentum term are two user defined BP parameters that affect the training of NN. The learning rate coefficient, employed during the adjustment of weights, can speed-up or slow-down the learning process. A bigger learning coefficient increases the weight changes, hence large steps are taken towards the minimum error level but may lead to oscillation, while smaller learning coefficients increase the number of steps taken to reach the desired error level and may trap the process at a local optimum. If an error curve shows a downward trend but with poor convergence at the latest stages a decrease of the learning rate coefficient is likely to accelerate the convergence.

The momentum term α when used with batch training affects the learning procedure in such a way that if the gradient points lie at the same direction for several consecutive iterations, it increases the step size by a factor of approximately $1/(1-\alpha)$. Therefore, a momentum term close to one is useful when a small learning rate is used. However, a large momentum term can exacerbate divergence problems when a large learning rate is used. In this study both parameters are initialized from the value 0.5. Then, when the error curve starts to oscillate the value of the learning rate coefficient is divided by two. If a flat error curve is obtained the value of the momentum term is increased to the value 0.99 while the value of the learning rate coefficient remains unchanged.

The basic NN configuration employed is fully connected with one hidden layer. Tests performed for more than one hidden layer showed no significant improvement in the obtained results. Based on this configuration various NN architectures are tested in order to find the most suitable one in terms of the smallest prediction error. This is done either with a direct comparison of predicted 'exact' results, or by means of the root mean square (RMS) error which is given by

$$e_{\mathrm{FMS}} = \sqrt{\frac{1}{N_P N_{\mathrm{out}}} \sum_{N_P} \sum_{i=1}^{N_{\mathrm{out}}} (\mathrm{tar}_i - \mathrm{out}_i)^2} \tag{22}$$

where NP is the total number of I/O pairs in the training set and N_{out} is the number of output units. 'Exact' results are those computed by a conventional structural analysis.

4.3. Selection of the training set

An important factor governing the success of the learning procedure of a NN architecture is the selection of the training set. A sufficient number of input data properly distributed in the design space together with the output data resulting from complete structural analyses are needed for the BP algorithm in order to provide satisfactory results. Overloading the network with unnecessary similar information results in overtraining without increasing the accuracy of the predictions. A few tens of structural analyses have been found sufficient for the examples considered to produce a satisfactory training of the NN. Ninety percent of those runs is used for training and the rest is used to test the results of the NN.

Most researchers split the design space into subregions and try to combine randomly the values within each subregion in order to obtain a training set which is representative of the whole design space. This procedure frequently leads to a huge number of training patterns in order to ensure that the whole design space is properly represented. In an effort to increase the robustness as well as the computational efficiency of the NN procedure two types of selection are used in this study: (i) the training set is chosen automatically based on a uniform distribution of the design variables in the design space, (ii) the training set is chosen automatically based on a Gaussian distribution of the design variables around the midpoints of the design space as shown in Fig. 5. These two selection approaches are compared with a third one where the training set is chosen manually on the basis of a trial and error procedure.

The last type of training set selection is considered to be the 'optimum' since it is based on the results obtained by the conventional ES optimization procedure and the data used is taken in the vicinity of the known optimum solution. The first type follows the general rule of randomly choosing combinations of input data from the whole range of the design space. The second type of training set selection was motivated from the fact that

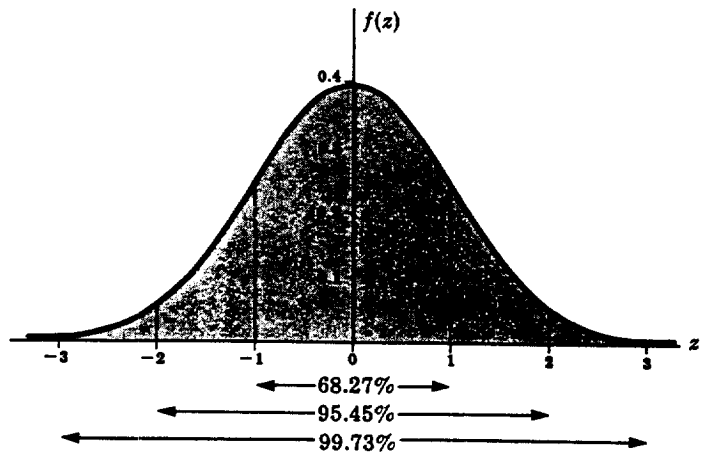


Fig. 5. Gaussian distribution.

usually the searching for the optimum and its location lies in the region near the midpoints of the design space. A Gaussian distribution was therefore used for the random selection of input data in order to cover the whole design space and enforce the selection of most input patterns around the midpoints of the design space.

5. Structural optimization based on ES and NN

After the selection of the suitable NN architecture and the training procedure is performed using a number (M) of data sets, selected as described previously, in order to obtain the I/O pairs needed for the NN training. Since the NN based structural analysis can only provide approximate results it is suggested that a correction on the output values should be performed in order to alleviate any inaccuracies entailed, especially when the constraint value is near the limit which divides the feasible with the infeasible region. Thus, a relaxation of this limit was introduced in this study before entering the optimization procedure during the NN testing phase. Therefore, a 'correction' of the allowable constraint values was performed analogous to the maximum testing error of the NN configuration. The maximum testing error is the bigger average error of the output values among testing patterns. When the predicted values were smaller than the accurate ones derived from the normal structural analysis then the allowable values of the constraints were decreased according to the maximum testing error of the NN configuration and vice versa.

The combined ES–NN optimization procedure is performed in two phases. The first phase includes the training set selection, the structural analyses required to obtain the necessary I/O data for the NN training, and finally the selection, training and testing of a suitable NN configuration. The second phase is the ES optimization stage where instead of the standard structural analyses the trained NN is used to predict the response of the structure in terms of objective and constraints function values due to different sets of design variables.

The proposed methodology ES–NN can be described in the following algorithm:

- NN training phase:
  1. *Training set selection step:* select s_i $(i = 1, 2, \ldots, M)$ input patterns.
  2. *Structural analysis step:* solve $K(s_i)u_i = f$ $(i = 1, \ldots, M)$.
  3. *Training step:* selection and training of a suitable NN architecture.
  4. *Testing step:* test NN and 'correct' allowable constraint values.

- ES optimization phase:
  1. *Selection step:* selection of s_i $(i = 1, 2, \ldots, \mu)$ parent vectors of the design variables.
  2. *Prediction step:* using NN to compute optimization function values for the μ parent vectors.
  3. *Constraints check:* all parent vectors become feasible.

4. *Offspring generation:* generate s_j ($j = 1, 2, \ldots, \lambda$) offspring vectors of the design variables.
5. *Prediction step:* using NN to compute optimization function values for the λ offspring vectors.
6. *Constraints check:* if satisfied continue, else change s_j and go to step 4.
7. *Selection step:* selection of the next generation parents according to $(\mu + \lambda)$ or (μ, λ) selection schemes.
8. *Convergence check:* if satisfied stop, else go to step 3.

6. Examples

The performance of the optimization methodology discussed in previous sections is investigated in a number of characteristic test examples in sizing as well as in shape structural optimization. The NN method used in this study is the back propagation algorithm META-NETS [33]. In the tables containing the results of the test examples the following abbreviations are used: ES refers to the standard evolution strategies optimization procedure, in which structural analyses are performed in the conventional way. ES–NN refers to the combined NN and ES optimization procedure, where the structural analysis response is predicted by a trained NN. For the three different types of training set selection that have been compared in this study the following abbreviations are used: (i) UT stands for the automatic selection of training set based on a uniform distribution of the design variables in the design space, (ii) GT stands for the automatic selection of training set based on a Gaussian distribution of the design variables in the design space and (iii) OT stands for the 'optimum' training set selection where the data used is taken in the vicinity of the optimum solution according to the results obtained a posteriori by the conventional ES optimization procedure. The symbol '(c)' denotes that the allowable limits of the constraints have been adjusted, as discussed previously, in order to 'correct' the NN predictions near the feasible region limits, while symbol '(v)' indicates that the final design is violating the constraints and thus it is infeasible. All examples were run on a Silicon Graphics Power Challenge computer.

6.1. Shape optimization examples

Two benchmark test examples [34] have been considered to illustrate the efficiency of the proposed methodology in shape optimization problems with continuous design variables. In both examples plane stress conditions and isotropic material properties are assumed (elastic modulus $E = 210{,}000\ \mathrm{N/mm^2}$ and Poisson's ratio $\nu = 0.3$). For the shape optimization examples the results obtained from a previous study [1] with a mathematical programming (MP) algorithm based on the sequential quadratic programming method (SQP) [35] are also included for comparison. In the tables containing the test results the following additional abbreviations are used: MP–ESA refers to the conventional MP method using a semi-analytical sensitivity analysis scheme, while MP–GFD refers to the conventional MP method using a finite difference approach to perform the sensitivity analysis calculations.

6.1.1. Connecting rod problem

This is a typical structural shape optimization test example. The problem definition is given in Fig. 6(a), whereas the optimized shape is depicted in Fig. 6(b). The linearly varying line load between key points 4 and 6 has a maximum value of $p = 500$ N/mm. The objective is to minimize the volume of the structure subject to a limit on the equivalent stress $\sigma_{\max} = 1{,}200\ \mathrm{N/mm^2}$. Two test cases are considered for this example with a different number of design variables. In the first test case the design model, which makes use of symmetry, consists of *twelve* key points, *four* primary design variables (7, 10, 11, 12) and *six* secondary design variables (7, 8, 9, 10, 11, 12). The stress constraints are imposed as a global constraint for all Gauss points and as key point constraints for key points 2, 3, 4, 5, 6 and 13. The movement directions of the design variables are indicated by the dashed arrows. Key points 8 and 9 are linked to point 7 so that the shape of the arc is preserved throughout the optimization procedure. For this test case the $(\mu + \lambda)$-ES approach is used with $\mu = \lambda = 4$.

In the second test case key points 7, 8 and 9 are not linked thus they can move independently, while three more design variables are introduced in the upper part as key points 12, 13 and 14. The design model now consists of *fifteen* key points and *nine* primary design variables (7, 8, 9, 10, 11, 12, 13, 14, 15) as shown in Fig. 7(a), whereas the optimized shape is depicted in Fig. 7(b). The constraints are again the global stress for all

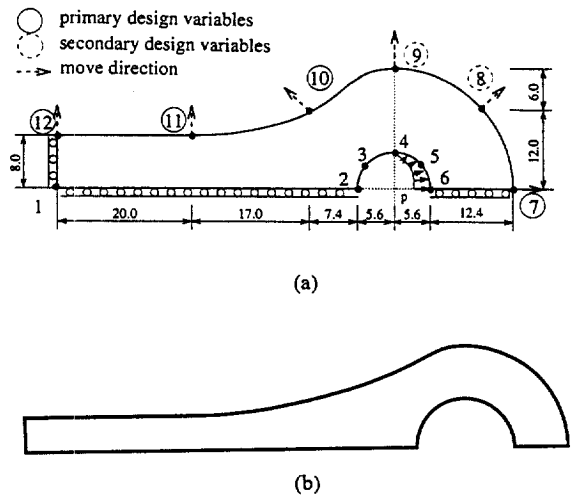


Fig. 6. Connecting rod—Test case 1 (4 design variables): (a) initial shape; (b) final shape.

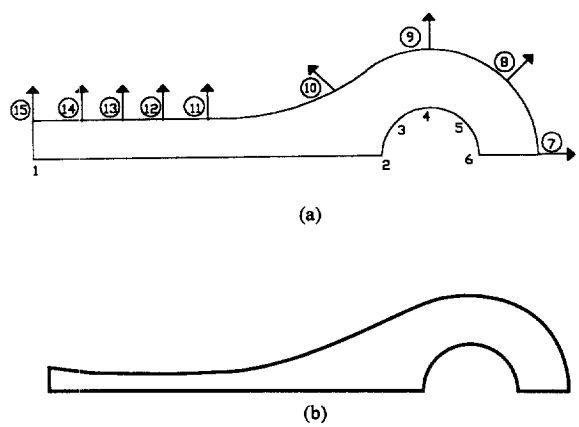


Fig. 7. Connecting rod—Test case 2 (9 design variables): (a) initial shape; (b) final shape.

Gauss points and the stresses on key points 2, 3, 4, 5, 6 and 15. For this test case the $(\mu + \lambda)$-ES approach is used with $\mu = \lambda = 10$.

For the first test case of this example the number of NN input units is equal to the number of design variables, the eight output units correspond to the seven constraint function values plus one for the value of the objective function, whereas the NN architecture used has one hidden layer. The influence of the number of hidden units on the accuracy of the NN prediction for the 'optimum' and for the Gaussian type of NN training set selection is shown in Table 1. For both cases 40 training sets are used. It can be seen that, in both cases, the RMS error and the maximum testing error are reaching a plateau after a certain number of hidden units without any further improvement. This plateau is reached for 12 hidden units which seems to follow the heuristic rule that the number of hidden units should equal the number of input units plus half the sum of input and output units. Therefore, the 4-12-8 NN architecture is chosen and used for the remaining runs. It should be pointed out that the results of the 'optimum' training set selection are used for testing the accuracy and the efficiency of the proposed automated training set selection schemes only, since it is based on the unrealistic assumption that the optimum solution is known.

Table 2 depicts, for the first test case of the connecting rod problem, the performance of the proposed ES–NN methodology, for all types of training set selection schemes, compared to the conventional ES optimization procedure for various numbers of NN training patterns. It can be seen that the 'correction' scheme of the allowable constraint values is more robust than the standard ES–NN version in producing feasible optimum designs. It can also be observed that a reduction in the number of training sets deteriorates the efficiency of the

Table 1
Connecting rod—Test case 1: NN accuracy for different number of hidden units

Hidden nodes	Training set selection			
	'Optimum' selection		Gaussian distribution	
	RMS error	MAX testing error (%)	RMS error	MAX testing error (%)
4	0.035	26.47	0.051	36.17
6	0.032	26.13	0.043	31.41
8	0.021	17.91	0.027	19.23
10	0.014	12.32	0.019	17.89
12	0.013	12.11	0.017	18.01
14	0.014	12.42	0.018	17.75

Table 2
Connecting rod—Test case 1: Performance of the optimization methods

Optimizer type	Optimization steps/ training patterns	ES–NN 'steps'	Computing time (s)				Optimum volume (mm^3)
			Analysis	Training	ES–NN	Total	
MP–ESA	47/–	–	–	–	–	1061	333
MP–GFD	48/–	–	–	–	–	1806	333
ES	53/–	–	–	–	–	1153	337
ES–NN–OT	–/40	49	862.3	181.8	0.9	1045	331(v)
ES–NN–OT(c)	–/40	59	862.3	181.8	1.1	1045	341
ES–NN–UT	–/40	65	862.3	209.4	1.1	1074	332(v)
ES–NN–UT(c)	–/40	68	862.3	209.4	1.2	1074	340
ES–NN–UT	–/32	45	696.4	173.2	0.8	870	325(v)
ES–NN–UT(c)	–/32	49	696.4	173.2	0.9	870	332(v)
ES–NN–GT	–/40	50	862.3	202.7	0.9	1066	328(v)
ES–NN–GT(c)	–/40	61	862.3	202.7	0.9	1066	339
ES–NN–GT	–/32	71	696.4	185.3	1.2	883	327(v)
ES–NN–GT(c)	–/32	79	696.4	185.3	1.3	883	331(v)

method since it leads to infeasible final designs. For this test case a marginal improvement in total computing time required by ES–NN over the standard ES is observed.

For the second test case of this example the number of NN input units is equal to the number of design variables, the eight output units correspond to the seven constraint function values plus one for the value of the objective function, whereas the NN architecture used has one hidden layer. The influence of the number of hidden units on the accuracy of the NN prediction for the 'optimum' and for the Gaussian type of NN training set selection is shown in Table 3. For these two cases 98 and 90 training sets are used, respectively. It can be seen that the RMS error and the maximum testing error are reaching a plateau without any further improvement, as in the previous test case, after a certain number of hidden units. This stationary point is reached for 17 hidden

Table 3
Connecting rod—Test case 2: NN accuracy for different number of hidden units

Hidden nodes	Training set selection			
	'Optimum' selection		Gaussian distribution	
	RMS error	MAX testing error (%)	RMS error	MAX testing error (%)
9	0.0125	25.65	0.025	31.3
11	0.0109	23.75	0.0251	30.7
13	0.0085	25.23	0.0193	27.8
15	0.0048	27.84	0.0187	24.4
17	0.0113	22.30	0.0182	24.1
19	0.012	22.86	0.0193	24.5

units which seems to satisfy the heuristic rule that the number of hidden units should equal the number of input units plus half the sum of input and output units. The 9-17-8 NN architecture is therefore chosen and used for the remaining runs. The results of the 'optimum' training set selection are used as previously for testing the accuracy and the efficiency of the proposed automated training set selection schemes.

Table 4 depicts, for the second test case of the connecting rod problem, the performance of the proposed ES–NN methodology, for all types of training set selection schemes, compared to the conventional ES optimization procedure for various numbers of NN training patterns. It can be seen that the 'correction' scheme of the allowable constraint values is more robust than the standard ES–NN version in producing feasible optimum designs. It can also be observed that a reduction in the number of training sets deteriorates the efficiency of the uniform type of training set selection, whereas for the Gaussian type of training set selection it improves the efficiency with minor violations of the constraints in the final design. For this test case a significant improvement in total computing time required by ES-NN over ES is observed.

In both test cases of this example the proposed methodology compares well with the MP optimization procedure with a semi-analytical sensitivity analysis scheme and outperforms the MP method when a finite difference approach is used to perform the sensitivity analysis calculations.

6.1.2. Square plate with central cut-out problem

The problem definition of this example is given in Fig. 8(a), where due to symmetry only a quarter of the plate is modeled. The optimized shape is depicted in Fig. 8(b). The two exterior sides of the plate are loaded

Table 4
Connecting rod—Test case 2: Performance of the optimization methods

Optimizer type	Optimization steps/ training patterns	ES–NN 'steps'	Computing time (s)				Optimum volume (mm^3)
			Analysis	Training	ES–NN	Total	
MP–ESA	91/–	–	–	–	–	1721	306
MP–GFD	90/–	–	–	–	–	4847	306
ES	133/–	–	–	–	–	2617	305
ES–NN–OT	–/98	128	1822.3	381.8	1.6	2206	300(v)
ES–NN–OT(c)	–/98	139	1822.3	381.8	1.5	2206	308
ES–NN–UT	–/90	109	1691.3	417.1	1.4	2110	277(v)
ES–NN–UT(c)	–/90	143	1691.3	417.1	1.9	2110	307
ES–NN–UT	–/70	98	1416.7	362.1	0.9	1780	242(v)
ES–NN–UT(c)	–/70	123	1416.7	362.1	1.2	1780	287(v)
ES–NN–GT	–/90	117	1691.3	391.5	1.0	2084	285(v)
ES–NN–GT(c)	–/90	153	1691.3	391.5	1.9	2085	309
ES–NN–GT	–/70	111	1416.7	347.4	1.1	1765	265(v)
ES–NN–GT(c)	–/70	132	1416.7	347.4	1.6	1766	300(v)

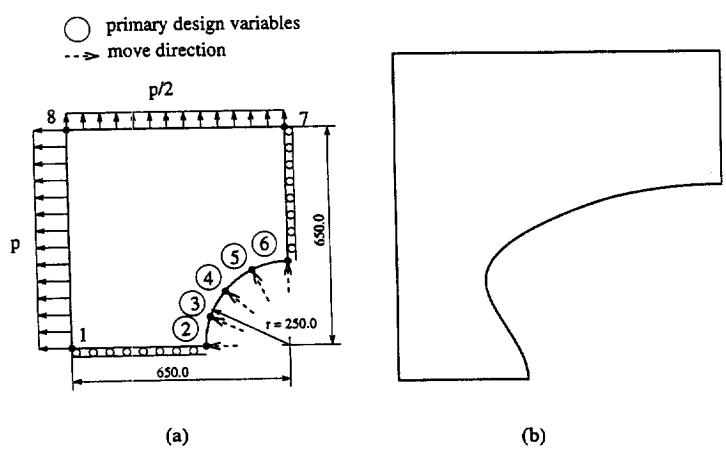


Fig. 8. Square plate—Test case 1 (5 design variables): (a) initial shape; (b) final shape.

with a distributed loading $p = 0.65$ N/mm^2, as shown in Fig. 8(a). The objective is to minimize the volume of the structure subject to a limit to the equivalent stress $\sigma_{max} = 7.0$ N/mm^2. Two test cases are considered for this example with a different number of design variables. The first design model, consists of *eight* key points and *five* primary design variables (2, 3, 4, 5, 6) which can move along radial lines. The movement directions of the design variables are indicated by the dashed arrows. The stress constraints are imposed as a global constraint for all the Gauss points and as key point constraints for key points 2, 3, 4, 5, 6 and 8. For this test case the $(\mu + \lambda)$-ES approach is used with $\mu = \lambda = 5$. In the second test case five more design variables are incorporated in the arc as shown in Fig. 9(a), whereas the final shape is depicted in Fig. 9(b). The design model now consists of *thirteen* key points and *ten* primary design variables (2, 3, 4, 5, 6, 7, 8, 9, 10, 11). The constraints are the global constraint for all Gauss points and the constraints on key points 2, 4, 7, 9, 11 and 13. For this test case the $(\mu + \lambda)$-ES approach is used with $\mu = \lambda = 10$.

For the first test case of this example the number of NN input units is equal to the number of design variables, the eight output units correspond to the seven constraint function values plus one for the value of the objective function, whereas the NN architecture used has one hidden layer. The influence of the number of hidden units on the accuracy of the NN prediction for the 'optimum' and the Gaussian type of NN training set selection is shown in Table 5. For these two training set selection schemes 47 and 50 training sets are used, respectively. It can be seen that the RMS error and the maximum testing error is reaching a plateau, as in the previous example, after a certain number of hidden units. This plateau is reached for 11 hidden units following the heuristic rule that the number of hidden units should equal the number of input units plus half the sum of input and output units. The 5-11-8 NN architecture is therefore chosen and used for the remaining runs.

Table 6 depicts, for the first test case of the square plate problem, the performance of the proposed ES–NN methodology, for all types of training set selection schemes, compared to the conventional ES optimization procedure for various numbers of NN training patterns. It is verified that the 'correction' scheme of the allowable constraint values is more robust than the standard ES–NN version in producing feasible optimum designs. In addition, it can be observed that a reduction in the number of training sets deteriorates the efficiency of the uniform type of training set selection, whereas for the Gaussian type it improves the efficiency with minor

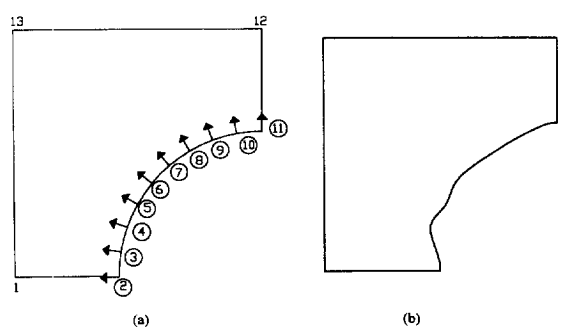


Fig. 9. Square plate—Test case 2 (10 design variables): (a) initial shape; (b) final shape.

Table 5
Square plate—Test case 1: NN accuracy for different number of hidden units

Hidden nodes	Training set selection			
	'Optimum' selection		Gaussian distribution	
	RMS error	MAX testing error (%)	RMS error	MAX testing error (%)
5	0.023	17.31	0.032	23.15
7	0.020	17.01	0.022	17.12
9	0.012	8.31	0.017	10.87
11	0.010	7.57	0.0145	9.67
13	0.010	7.61	0.0143	9.61

Table 6
Square plate—Test case 1: Performance of the optimization methods

Optimizer type	Optimization steps/ training patterns	ES–NN 'steps'	Computing time (s)				Optimum volume (mm^3)
			Analysis	Training	ES–NN	Total	
MP–ESA	72/–	–	–	–	–	1895	280
MP–GFD	75/–	–	–	–	–	4190	279
ES	127/–	–	–	–	–	2141	279
ES–NN–OT	–/47	120	890.7	229.6	1.8	1122	272(v)
ES–NN–OT(c)	–/47	131	890.7	229.6	1.8	1122	281
ES–NN–UT	–/50	112	946.3	238.8	1.8	1238	272(v)
ES–NN–UT(c)	–/50	132	946.3	238.8	2.0	1238	281
ES–NN–UT	–/40	138	674.4	245.1	1.9	921	259(v)
ES–NN–UT(c)	–/40	151	674.4	245.1	2.1	922	271(v)
ES–NN–GT	–/50	116	946.3	254.4	1.9	1203	270(v)
ES–NN–GT(c)	–/50	125	946.3	254.4	2.1	1203	283
ES–NN–GT	–/40	112	674.4	238.8	1.8	915	263(v)
ES–NN–GT(c)	–/40	132	674.4	238.8	2.0	915	275(v)

violations of the constraints in the final design. For this test case a drastic improvement in total computing time required by ES–NN over ES is observed.

For the second test case of this example the number of NN input units is equal to the number of design variables, the eight output units correspond to the seven constraint function values plus one for the value of the objective function, whereas the NN architecture used has one hidden layer. The influence of the number of hidden units on the accuracy of the NN prediction for the 'optimum' and for the Gaussian type of NN training set selection for 117 and 100 training sets, respectively, is shown in Table 7. The plateau for both the RMS error and the maximum testing error is reached for 18 hidden units satisfying again the heuristic rule. The 10-18-8 NN architecture is therefore chosen and used for the remaining runs.

Table 8 depicts, for the second test case of the square plate problem, the performance of the proposed ES–NN methodology for all types of training set selection schemes compared to the conventional ES optimization procedure for various numbers of NN training patterns. The conclusions which can be drawn from this test case are similar to those of the first test case, while the improvement in total computing time required by ES–NN over ES is more evident.

In the first test case of this example the proposed methodology outperforms the MP optimization procedure, whereas in the second test case it provides the optimum design while the MP method fails to converge.

6.2. Sizing optimization examples

Two benchmark test examples of space frames with six and twenty storeys, respectively, have been considered to illustrate the efficiency of the proposed methodology in sizing optimization problems with discrete design variables. In both examples the modulus of elasticity is 200 GPa (29,000 ksi) and the yield stress is

Table 7
Square plate—Test case 2: NN accuracy for different number of hidden units

Hidden nodes	Training set selection			
	'Optimum' selection		Gaussian distribution	
	RMS error	MAX testing error (%)	RMS error	MAX testing error (%)
10	0.020	27.7	0.032	41.1
12	0.017	27.1	0.032	37.1
14	0.019	24.1	0.027	25.6
16	0.014	18.7	0.026	24.2
18	0.011	12.6	0.021	19.7
20	0.012	13.4	0.020	19.8

Table 8
Square plate—Test case 2: Performance of the optimization methods

Optimizer type	Optimization steps/ training patterns	ES–NN 'steps'	Computing time (s)				Optimum volume (mm^3)
			Analysis	Training	ES–NN	Total	
MP–ESA	61/–	–	–	–	–	3675	216(v)
MP–GFD	60/–	–	–	–	–	9512	221(v)
ES	874/–	–	–	–	–	21850	320
ES–NN–OT	–/117	917	2806.8	496.9	7.2	3311	313(v)
ES–NN–OT(c)	–/117	901	2806.8	496.9	6.8	3310	321
ES–NN–UT	–/100	713	2439.7	471.4	5.3	2916	374
ES–NN–UT(c)	–/100	863	3439.7	471.4	7.3	2918	331
ES–NN–UT	–/80	835	2001.7	463.4	6.9	2472	287(v)
ES–NN–UT(c)	–/80	907	2001.7	463.4	8.0	2473	315(v)
ES–NN–GT	–/100	802	2439.7	447.3	6.1	2893	293(v)
ES–NN–GT(c)	–/100	911	2439.7	447.3	8.0	2895	324
ES–NN–GT	–/80	755	2001.7	439.1	5.6	2446	298(v)
ES–NN–GT(c)	–/80	892	2001.7	439.1	7.5	2448	334
ES–NN–GT	–/60	685	1485.3	385.1	5.0	1875	254(v)
ES–NN–GT(c)	–/60	796	1485.3	385.1	6.1	1876	292(v)

$\sigma_y = 250$ MPa (36 ksi). The cross section of each member is assumed to be a W-shape and for each member two design variables are considered as shown in Fig. 10. The objective function of the problems is the weight of the structure. The constraints are imposed on the inter-storey drifts and the maximum non-dimensional ratio q of Eqs. (3) and (4) in each element group which combines axial force and bending moment. The values of allowable axial and bending stresses are $F_a = 150$ MPa and $F_b = 165$ MPa, respectively, whereas the maximum allowable inter-storey drift is limited to 2.2 inches which corresponds to 1.5% of the height of each storey.

6.2.1. Six storey space frame

This example consists of 63 elements with 180 degrees of freedom as shown in Fig. 11. The beams have length $L_1 = 7.32$ m and the columns $L_2 = 3.66$ m. The structure is loaded with a 19.16 kPa gravity load on all floor levels and a lateral load of 109 kN applied at each node in the front elevation along the z direction. The element members are divided into five groups, as shown in Fig. 10, each one having two design variables resulting in a total of ten design variables. The constraints are imposed on the maximum allowable inter-storey drift and the non-dimensional ratio q in each element group. For this test case the $(\mu + \lambda)$-ES approach is used with $\mu = \lambda = 5$.

For this example the number of NN input units is equal to the number of design variables, whereas the twelve output units correspond to the two values of axial force and bending moment for the five element groups, plus one for the value of the inter-storey drift, plus one for the value of the objective function. The NN architecture used has one hidden layer, while the investigation performed on the influence of the number of hidden units is

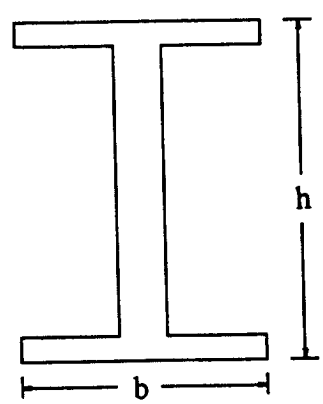


Fig. 10. W-shaped cross-section design variables.

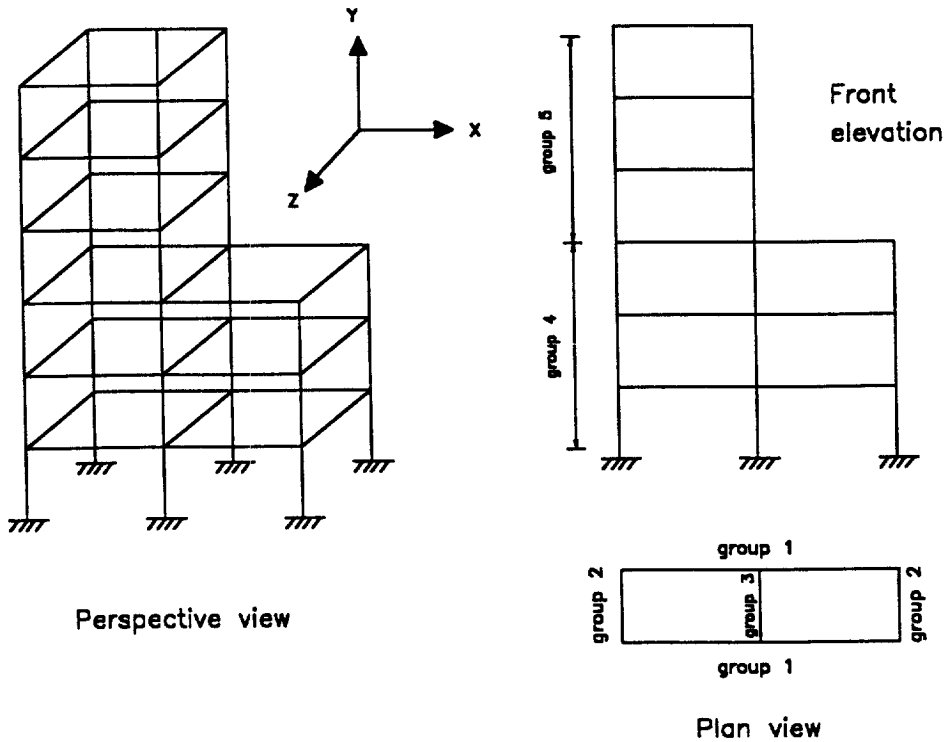


Fig. 11. Six storey space frame.

shown in Table 9. For the 'optimum' and the Gaussian type of NN training set selection 82 and 100 training sets are used, respectively. The plateau for both the RMS error and the maximum error is reached for 18 hidden units which is conforming with the heuristic rule. Thus, the 10-18-12 NN architecture is chosen and used for the remaining runs.

Table 10 depicts the performance of the proposed ES–NN methodology compared to the conventional ES optimization procedure for various numbers of NN training patterns. It can be observed that a marginal reduction in the number of training sets deteriorates the efficiency of the uniform type of training set selection, whereas it improves the efficiency of the Gaussian type with a minor increase of the objective function value in the final design. For this test case a slight increase in total computing time required by ES–NN over ES is observed due to the small size of the problem.

6.2.2. Twenty storey space frame

This example is presented in Fig. 12 (plan a) consisting of 1,020 members and 2,400 degrees of freedom. The loads for this example are taken as uniform vertical forces applied at joints equivalent to uniform load of 4.788

Table 9
Six storey space frame: NN accuracy for different number of hidden units

Hidden nodes	Training set selection			
	'Optimum' selection		Gaussian distribution	
	RMS error	MAX testing error (%)	RMS error	MAX testing error (%)
10	0.030	27.6	0.041	39.1
12	0.022	24.3	0.038	32.8
14	0.019	21.6	0.038	33.2
16	0.016	18.8	0.029	27.1
18	0.010	17.8	0.020	20.7
20	0.020	21.4	0.019	21.3

Table 10
Six storey space frame: Performance of the optimization methods

Analysis type	Number of FE analyses-training patterns	Number of NN analyses	Computing time (s)				Optimum weight (kN)
			Analysis	Training	ES–NN	Total	
ES	281	–	–	–	–	116.3	867
ES–NN–OT	82	275	35.6	313.0	2.7	351.3	908
ES–NN–OT(c)	82	263	35.6	313.0	2.7	351.3	883
ES–NN–UT	100	263	39.3	513.1	2.7	555.1	786(v)
ES–NN–UT(c)	100	255	39.3	513.1	2.6	555	918
ES–NN–UT	80	187	33.1	342.6	1.8	377.5	685(v)
ES–NN–UT(c)	80	232	33.1	342.6	2.4	378.1	758(v)
ES–NN–GT	100	274	39.3	478.0	3.0	520.3	927
ES–NN–UT(c)	100	269	39.3	478.0	2.9	520.2	898
ES–NN–GT	80	274	33.1	321.7	3.0	357.8	940
ES–NN–GT(c)	80	266	33.1	321.7	2.9	357.7	884
ES–NN–GT	60	169	24.8	295.3	1.7	321.8	996
ES–NN–GT(c)	60	211	24.8	295.3	1.7	322.2	961

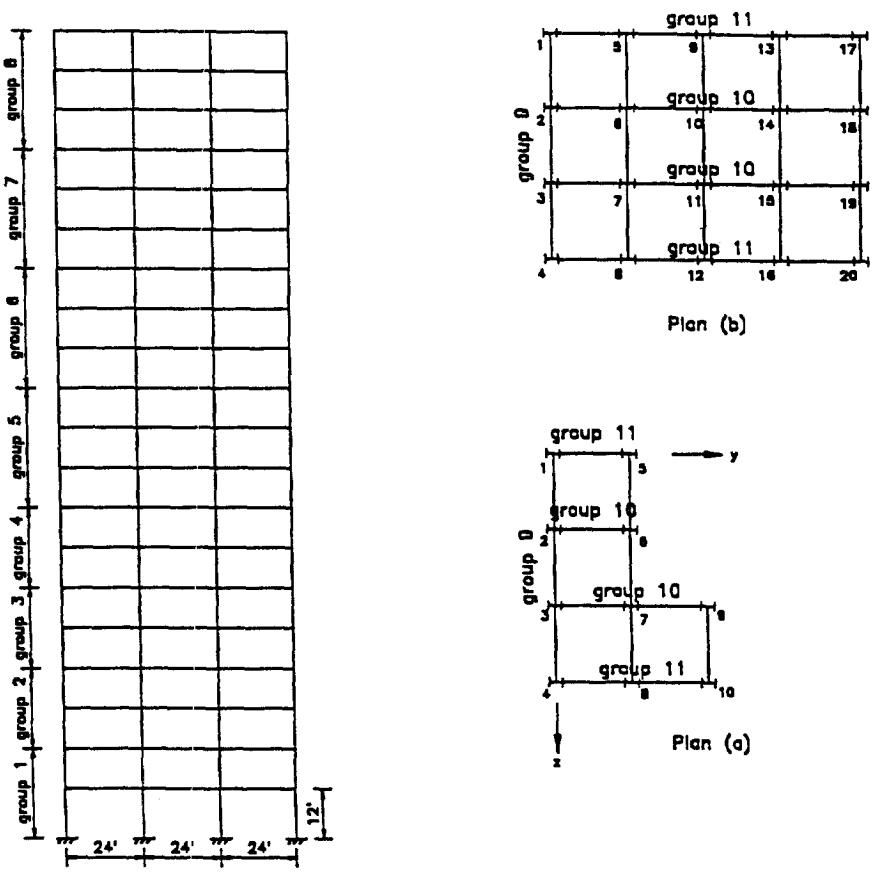


Fig. 12. Twenty storey space frame.

kPa and horizontal forces equivalent to uniform forces of 0.956 kPa on the largest surface. The element members are divided into 11 groups shown in Fig. 12 (plan b) and the total number of design variables is 22. The constraints are imposed on the maximum allowable inter-storey drift and the maximum non-dimensional

Table 11
Twenty storey space frame: NN accuracy for different number of hidden units

Hidden nodes	Training set selection			
	'Optimum' selection		Gaussian distribution	
	RMS error	MAX testing error (%)	RMS error	MAX testing error (%)
20(1)	0.021	17.7	0.037	35.1
22(2)	0.020	17.9	0.033	29.8
24(3)	0.020	17.3	0.038	36.3
26(4)	0.015	11.8	0.025	24.4
28(5)	0.012	10.6	0.018	19.3
30	0.011	9.81	0.021	20.9

ratio q in each element group, as in the previous example. For this test case the $(\mu + \lambda)$-ES approach is used with $\mu = \lambda = 10$.

For this example the number of NN input units is equal to the number of design variables, whereas the 24 output units correspond to the two values of axial force and bending moment for the 11 element groups, plus one for the value of the inter-storey drift, plus one for the value of the objective function. The performance of the 'optimum' and the Gaussian type of NN training set selection with 130 and 220 training sets, respectively, is shown in Table 11. The heuristic rule for choosing the optimum NN configuration is also confirmed in this case and thus the 22-28-24 NN architecture is chosen and used for the remaining runs. Table 12 depicts the performance of the proposed ES–NN methodology compared to the conventional ES optimization procedure for various numbers of NN training patterns, as shown in the second column. Table 12 also presents the influence of the accuracy of the NN training on the results for the 'optimum' training set selection. It can be observed that a moderate reduction in the number of training sets deteriorates the efficiency of the uniform type of training set selection, whereas for the Gaussian type it improves the efficiency with a minor increase of the objective

Table 12
Twenty storey space frame: Performance of the optimization methods

Analysis type	Number of FE analyses-training patterns	Number of NN analyses	Computing time (s)				Optimum weight (kN)
			Analysis	Training	ES–NN	Total	
ES	1566	–	–	–	–	24930	5430
ES–NN–OT	130(1)	1311	2237.3	649.4	10.7	2897	5348(v)
ES–NN–OT	130(2)	1337	2237.3	655.3	10.5	2903	5344(v)
ES–NN–OT	130(3)	1421	2237.3	670.3	11.1	2919	5381(v)
ES–NN–OT	130(4)	1491	2237.3	668.6	11.1	2917	5349(v)
ES–NN–OT	130(5)	1465	2237.3	673.2	11.0	2921	5407(v)
ES–NN–OT(c)	130(1)	1507	2237.3	649.4	12.5	2899	5442
ES–NN–OT(c)	130(2)	1527	2237.3	655.3	12.1	2905	5465
ES–NN–OT(c)	130(3)	1569	2237.3	670.3	12.9	2920	5452
ES–NN–OT(c)	130(4)	1577	2237.3	668.6	12.1	2918	5438
ES–NN–OT(c)	130(5)	1563	2237.3	673.2	12.3	2923	5432
ES–NN–UT	220	1417	3497.2	1007.1	10.7	4515	5138(v)
ES–NN–UT(c)	220	1577	3497.2	1007.1	12.1	4516	5484
ES–NN–UT	180	1357	2387.9	607.1	10.6	3006	5298(v)
ES–NN–UT(c)	180	1470	2387.9	607.1	11.1	3006	5396(v)
ES–NN–GT	220	1609	3497.2	969.1	14.4	4481	5173(v)
ES–NN–GT(c)	220	1554	3497.2	969.1	13.7	4481	5453
ES–NN–GT	180	1593	2865.2	672.1	14.2	3551	5264(v)
ES–NN–GT(c)	180	1611	2865.2	672.1	14.4	3552	5502
ES–NN–GT	150	1203	2387.9	607.1	9.8	3005	4601(v)
ES–NN–GT(c)	150	1379	2387.9	607.1	10.7	3006	5194(v)

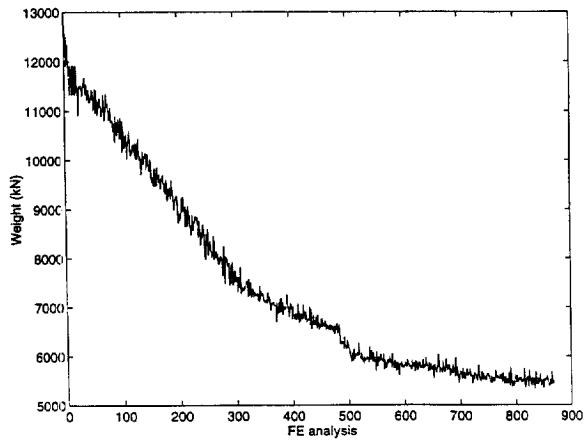


Fig. 13. Iteration history of the objective function.

function value in the final design. For this test case a dramatic improvement in total computing time required by ES–NN over ES is observed. The iteration history of the value of the objective function is shown in Fig. 13.

7. Conclusions

The implementation of a hybrid optimization procedure, based on the combination ES and NN, in shape and sizing structural optimization problems was found to be very effective. The time-consuming requirements of repeated analyses associated with the optimization procedure using ES motivated the use of NN. The computational effort involved in the optimization procedure using ES becomes excessive in large-scale problems and the use of NN to 'predict' the necessary optimization data for ES can practically eliminate any limitation on the size of the problem, while the predicted structural response corresponding to different optimization simulations falls within acceptable tolerances. The methodology presented is an efficient, robust and generally applicable optimization procedure capable of finding the global optimum design of complicated structural optimization problems. Additionally, it was found that the proposed hybrid optimization methodology can reach the optimum for large and computationally intensive problems at a fraction of the computing time required by the standard ES optimization algorithm and the conventional method based on mathematical programming technique.

The study performed on the type of data selection required for NN training showed that the rule of randomly choosing combinations of input data using a Gaussian distribution around the midpoints of the design space is the most successful training set selection scheme which makes the ES–NN methodology robust and efficient. This approach was motivated by the fact that the searching for the optimum and its location usually lies in the region near the midpoints of the design space and was found to be more efficient than the random uniform data distribution and the 'optimum' training set selection based on the unrealistic assumption that the optimum solution is known. In addition, it enables the possibility of using a smaller number of training sets as it was shown in the test examples presented.

Acknowledgments

This work has been supported by HC&M/9203390 project of the EU. The authors wish to thank E. Hinton and J. Sienz of University College of Swansea and G. Thierauf and J. Cai of University of Essen for their cooperation.

References

[1] M. Papadrakakis, Y. Tsompanakis and N.D. Lagaros, Structural shape optimization using evolution strategies, Report ISASR 96-4 (1996).

[2] M. Papadrakakis, Y. Tsompanakis, E. Hinton and J. Sienz, Advanced solution methods in topology optimization and shape sensitivity analysis, J. Engrg. Comput. 13(5) (1996) 57–90.

[3] L.J. Fogel, A.J. Owens and M.J. Walsh, Artificial Intelligence through Simulated Evolution (Wiley, New York, 1966).

[4] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning (Addison-Wesley Publishing Co., Inc., Reading, MA, 1989).

[5] J. Holland, Adaptation in Natural and Artificial Systems (University of Michigan Press, Ann Arbor, 1975).

[6] I. Rechenberg, Evolution Strategy: Optimization of Technical Systems According to the Principles of Biological Evolution (Frommann-Holzboog, Stuttgart, 1973).

[7] H.P. Schwefel, Numerical Optimization for Computer Models (Wiley & Sons, Chichester, UK, 1981).

[8] P. Hajela and L. Berke, Neurobiological computational models in structural analysis and design, Comput. Struct. 41 (1991) 657–667.

[9] L. Berke, S.N. Patnaik and P.L.N. Murthy, Optimum design of aerospace structural components using neural networks, Comput. Struct. 48 (1993) 1001–1010.

[10] A. Berrais, Neural networks in structural engineering: State of the art, in: B.H.V. Topping, ed., Advances in Computational Structures Technology (CIVIL-COMP Press, Edinburgh, 1996) 93–101.

[11] Z. Waszczyszyn, Some recent and current problems of neurocomputing in civil and structural engineering, in: B.H.V. Topping, ed., Advances in Computational Structures Technology (CIVIL-COMP Press, Edinburgh, 1996) 43–58.

[12] D.E. Rummelhart and J.L. McClelland, Parallel Distributed Processing, Vol. 1: Foundations (The MIT Press, Cambridge, 1986).

[13] E. Hinton and J. Sienz, Aspects of adaptive finite element analysis and structural optimization, in: B.H.V. Topping and M. Papadrakakis, eds., Advances in Structural Optimization (CIVIL-COMP Press, Edinburgh, 1994) 1–26.

[14] E. Ramm, K.-U. Bletzinger, R. Reitinger and K. Maute, The challenge of structural optimization, in: B.H.V. Topping and M. Papadrakakis, eds., Advances in Structural Optimization (CIVIL-COMP Press, Edinburgh, 1994) 27–52.

[15] V. Thevendran, N.C. Das Gupta and G.H. Tan, Minimum weight design of multi-bay multi-storey steel frames, Comput. Struct. 43(3) (1992) 495–503.

[16] G.M. Barsan, Optimal design of planar frames based on structural performance criteria, Comput. Struct. 3(2) (1994) 1395–1400.

[17] AISC-Manual of Steel Construction, 9th edition, American Institute of Steel Constructions, Chicago, IL, 1989.

[18] M. Schoenauer, Shape representation for evolutionary optimization and identification in structural mechanics, in: G. Winter, J. Periaux, M. Galan and P. Cuesta, eds., Genetic Algorithms in Engineering and Computer Science (J. Wiley, 1995) 443–464.

[19] F. Hoffmeister and T. Bäck, Genetic algorithms and evolution strategies—similarities and differences, in: H.P. Schwefel and R. Manner, eds., Parallel Problems Solving from Nature (Springer-Verlag, Berlin, Germany, 1991) 455–469.

[20] G. Thierauf and J. Cai, A two level parallel evolution strategy for solving mixed-discrete structural optimization problems. The 21st ASME Design Automation Conference, Boston, MA, September, 1995, 17–221.

[21] G. Thierauf and J. Cai, Structural optimization based on evolution strategy, in: M. Papadrakakis and G. Bugeda, eds., Advanced Computational Methods in Structural Mechanics (CIMNE, Barcelona, 1996) 266–280.

[22] M. Papadrakakis, N.D. Lagaros, G. Thierauf and J. Cai, Advanced solution methods in structural optimization based on evolution strategies, Engrg. Comput. (to be published).

[23] A.D. Belegundu and J.S. Arora, A study of mathematical programming methods for structural optimization, Part I: Theory, Part II: Numerical results, Int. J. Numer. Methods Engrg. 21 (1985) 1583–1623.

[24] P.B. Thanedar, J.S. Arora, C.H. Tseng, O.K. Lim and G.J. Park, Performance of some SQP methods on structural optimization problems, Int. J. Numer. Methods Engrg. 23 (1986) 2187–2203.

[25] L.S. Lasdon, A.D. Warren, A. Jain and R. Ratner, Design and testing of a generalized reduced gradient code for nonlinear programming, ACM Trans. Math. Softw. 4(1) (1978) 34–50.

[26] K. Svanberg, The method of moving asymptotes, a new method for structural optimization, Int. J. Numer. Methods Engrg. 23 (1987) 359–373.

[27] Y.H. Pao, Adaptive Pattern Recognition and Neural Networks (Addison-Wesley Publishing Co, 1989).

[28] R.C. Shieh, Massively parallel structural design using stochastic optimization and mixed neural net/finite element analysis methods, Comput. Syst. Engrg. 5(4–6) (1994) 455–467.

[29] Z.P. Swewczyk and P. Hajela, Neurocomputing strategies in structural design-decomposition based optimization, Struct. Optim. 8 (1994) 242–250.

[30] M. Papadrakakis, V. Papadopoulos and N.D. Lagaros, Structural reliability analysis of elastic-plastic structures using neural networks and Monte Carlo simulation, Comput. Methods Appl. Mech. Engrg. 136 (1996) 145–163.

[31] W.S. Sarle, Neural network implementation in SAS software, SAS Institute Inc, 1994.

[32] Y. Hirose, K. Yamashita and S. Hijiya, Back-propagation algorithm which varies the number of hidden units, Neural Networks 4 (1991) 61–66.

[33] M.J. Embrechts, Metaneural-Version 4.1, Rensselear Polytechnique Institute, 1994.

[34] E. Hinton and J. Sienz, Studies with a robust and reliable structural shape optimization tool, in: B.H.V. Topping, ed., Developments in Computational Techniques for Structural Engineering (CIVIL-COMP Press, Edinburgh, 1995) 343–358.

[35] NAG, Software manual, NAG Ltd, Oxford, UK, 1988.