PERGAMON

Computers and Structures 80 (2002) 571–589

Computers & Structures

www.elsevier.com/locate/compstruc

Structural optimization using evolutionary algorithms

Nikolaos D. Lagaros, Manolis Papadrakakis *, George Kokossalakis

Institute of Structural Analysis & Seismic Research, National Technical University Athens, Zografou Campus, Athens 157 80, Greece

Received 6 February 2001; accepted 9 January 2002

Abstract

The objective of this paper is to investigate the efficiency of various evolutionary algorithms (EA), such as genetic algorithms and evolution strategies, when applied to large-scale structural sizing optimization problems. Both type of algorithms imitate biological evolution in nature and combine the concept of artificial survival of the fittest with evolutionary operators to form a robust search mechanism. In this paper modified versions of the basic EA are implemented to improve the performance of the optimization procedure. The modified versions of both genetic algorithms and evolution strategies combined with a mathematical programming method to form hybrid methodologies are also tested and compared and proved particularly promising. The numerical tests presented demonstrate the computational advantages of the discussed methods, which become more pronounced in large-scale optimization problems. © 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Structural optimization; Genetic algorithms; Evolution strategies; Handling of constraints; Sequential quadratic programming

1. Introduction

Computer algorithms based on the process of natural evolution have been found capable to produce very powerful and robust search mechanisms although the similarity between these algorithms and the natural evolution is based on crude imitation of biological reality. The resulting evolutionary algorithms (EA) are based on a population of individuals, each of which represents a search point in the space of potential solutions of a given problem. These algorithms have some selection process based on the fitness of the individuals and some recombination operators. The best known EA in this class include evolutionary programming (EP) [1,2], genetic algorithms (GA) [3,4] and evolution strategies (ES) [5,6].

Both GA and ES imitate biological evolution in nature and have three characteristics that differ from other conventional optimization algorithms: (i) In place of the usual deterministic operators, they use randomized operators: mutation, selection and recombination. (ii) Instead of a single design point, they work simultaneously with a population of design points in the space of design variables. (iii) They can handle, with minor modifications, continuous, discrete or mixed optimization problems. The second characteristic allows for natural implementation of GA and ES on a parallel computing environment [7,8].

In a gradient-based mathematical programming approach the optimization algorithm proceeds with the following steps: (i) Evaluation of displacements and stresses. (ii) Computation of sensitivities by perturbing each design variable by a small amount. (iii) Solution of the optimization problem and update the design of the structure. These steps are repeated until convergence is achieved. The most time-consuming part of this process is related to the sensitivity analysis phase, which is an important ingredient of all mathematical programming

* Corresponding author. Tel.: +301-772-1694; fax: +301-772-1693.

E-mail address: mpapadra@central.ntua.gr (M. Papadrakakis).

PII: S0045-7949(02)00027-5

optimization methods and may consume a large part of the total computational effort [9]. On the other hand, the application of EA that are based on probabilistic searching, such as GA and ES, do not need gradient information and therefore avoid performing the computationally expensive sensitivity analysis step.

Mathematical programming methods, such as the sequential quadratic programming (SQP) approach, make use of local curvature information derived from linearization of the original functions by using their derivatives with respect to the design variables. The linearization is performed at points obtained in the process of optimization to construct an approximate model of the initial problem. These methods present a satisfactory local rate of convergence, but they cannot assure that the global optimum can be found. They do although assure if the problem is strictly convex. On the other hand, EA are in general more robust and present a better global behaviour than the mathematical programming methods. They may suffer, however, from a slow rate of convergence towards the global optimum and do not guarantee convergence to the global optimum.

Structural optimization problems are characterized by various objective and constraint functions, which are generally non-linear functions of the design variables. These functions are usually implicit, discontinuous and non-convex. The mathematical formulation of structural optimization problems with respect to the design variables, the objective and constraint functions depends on the type of the application. However, all optimization problems can be expressed in standard mathematical terms as a non-linear programming problem (NLP), which in general form can be stated as follows:

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

where, s is the vector of design variables, $F(s)$ is the objective function to be minimized, $h_j(s)$ are the behavioral constraints, s_i^{l} and s_i^{u} are the lower and the upper bounds of a typical design variable s_i. Equality constraints are rarely imposed in this type of problems except in some cases for design variable linking. Whenever they are used they are treated for simplicity as a set of two inequality constraints.

In this work the efficiency of various EA is investigated in structural sizing optimization problems. Furthermore, in order to benefit from the advantages of both methodologies, combinations of EA with SQP are also examined in an attempt to increase further the robustness as well as the computational efficiency of the optimization procedure. The numerical tests presented demonstrate the computational advantages of the discussed methods, which become more pronounced in large-scale and computationally intensive optimization problems.

2. Genetic algorithms

GA are probably the best-known EA, receiving substantial attention in recent years. The first attempt to use EA took place in the sixties by a team of biologists [10] and was focused in building a computer program that would simulate the process of evolution in nature. However, the GA model used in this study and in many other structural design applications refers to a model introduced and studied by Holland and co-workers [4]. In general the term genetic algorithm refers to any population-based model that uses various operators (selection–crossover–mutation) to evolve. In the basic genetic algorithm each member of this population will be a binary or a real valued string, which is sometimes referred to as a *genotype* or, alternatively, as a *chromosome*.

2.1. The basic genetic algorithms

2.1.1. The three main steps of the basic GA

Step 0 initialization: The first step in the implementation of any genetic algorithm is to generate an initial population. In most cases the initial population is generated randomly. In this study in order to perform a comparison between various optimization techniques the initial population is fixed and is chosen in the neighborhood of the initial design used for the mathematical programming method. After creating an initial population, each member of the population is evaluated by computing the representative objective and constraint functions and comparing it with the other members of the population.

Step 1 selection: Selection operator is applied to the current population to create an intermediate one. In the first generation the initial population is considered as the intermediate one, while in the next generations this population is created by the application of the selection operator.

Step 2 generation (crossover–mutation): In order to create the next generation, crossover and mutation operators are applied to the intermediate population to create the next population. Crossover is a reproduction operator, which forms a new chromosome by combining parts of each of the two parental chromosomes. Mutation is a reproduction operator that forms a new chromosome by making (usually small) alterations to the values of genes in a copy of a single parent chromosome. The process of going from the current population to the next population constitutes one generation in the evolution process of a genetic algorithm. If the termination criteria are satisfied the procedure stops, otherwise, it returns to step 1.

2.2. Micro genetic algorithms

The micro genetic algorithm (μGA) was introduced by Krishnakumar [11] and applied to simple mathematical test functions and to the wind shear optimal guidance problem. The main objective of this scheme is to reduce the size of the population compared to the basic one. This corresponds, in the case of structural optimization problems discretized with finite elements, to less finite element analyses per generation. It is a known fact that GA generally exhibit poor performance with very small population due to insufficient information processed and premature convergence to non-optimal results. A remedy to this problem, suggested by Goldberg [12], could be to restart the evolution process in case of nominal convergence with a new initial population which will include the best solution already achieved. Based on this suggestion Krishnakumar proposed the μGA which can be described by the following steps:

Step 0 initialization: The first step generates a population of size five either randomly or by generating four strings randomly and by selecting one good string from any previous search, or according to the experience of the designer.

Step 1 fitness evaluation: In this step the fitness of each individual is evaluated and the best string is determined. The best string is labeled as string five and it is carried to the next generation (elitist strategy). In this way there is a guarantee that the information about good strings are not lost.

Step 2 generation: According to the previous step the best individual of the current generation is carried out to the next one. The remaining four members of the next generation are chosen according to the tournament selection operator. After the selection operator is terminated the crossover operator is applied.

Step 3 convergence check: If the termination criteria is satisfied the process ends, otherwise check for nominal convergence which is measured by bit wise convergence in case of binary coding or by comparing the design variables in case of real valued strings. If converged go to step 0, else return to Step 1.

A modified version of μGA is tested in this study, where only feasible designs are accepted for the evolution process. This version, which resembles the death penalty treatment of the constraints adopted by ES, is abbreviated to m μGA.

3. Methods for handling the constraints

Although genetic algorithms were initially developed to solve unconstrained optimization problems, during the last decade several methods have been proposed for handling constrained optimization problems as well. The methods based on the use of penalty functions are employed in the majority of cases for treating constraint optimization problems with GA. In this study methods belonging to this category have been implemented and will be briefly described in the following section.

3.1. Method of static penalties

In the method of static penalties the objective function is modified as follows:

$$F'(s) = \begin{cases} F^{(n)}(s), & \text{if } s \in \mathscr{F} \\ F^{(n)}(s) + p \cdot \mathrm{viol}^{(n)}(s), & \text{otherwise} \end{cases} \tag{2}$$

where p is the static penalty parameter, $\mathrm{viol}^{(n)}(s)$ is the sum of the violated constraints and $F^{(n)}(s)$ is the objective function to be minimized, both normalized in [0,1], while $\mathscr{F}$ is the feasible region of the design space.

$$\mathrm{viol}(s) = \sum_{j=1}^{m} h_j(s) \tag{3}$$

The sum of the violated constraints is normalized before it is used for the calculation of the modified objective function. The main advantage of this method is its simplicity. However, there is no guidance on how to choose the single penalty parameter p. If it is chosen too small the search will converge to an infeasible solution, otherwise, if it is chosen too large, a feasible solution may be located but it would be far from the global optimum. A large penalty parameter will force the search procedure to work away from the boundary where the global optimum is usually located and divides the feasible region from the infeasible one.

3.2. Method of dynamic penalties

The method of dynamic penalties was proposed by Joines and Houck [13] and applied to mathematical test functions. As opposed to the previous method, dynamic penalties are implemented in this case. Individuals are evaluated (at the generation g) by the following formula

$$F'(s) = F^{(n)}(s) + (c \cdot g)^{\alpha} \mathrm{viol}^{(n)}(s) \tag{4}$$

$$\mathrm{viol}(s) = \sum_{j=1}^{m} h_j^{\beta}(s) \tag{5}$$

where c, α and β are constants. A reasonable choice for these parameters was proposed as follows: $c = 0.5$–2.0, $\alpha = \beta = 1$ or 2. For high generation number, however, the $(c \cdot g)^{\alpha}$ component of the penalty term takes extremely large values which makes even the slightly violated designs not to be selected in subsequent generations. Thus, the system has little chances to escape from local optima. In most experiments reported by

Michalewicz [14] the best individual was found in early generations.

3.3. Augmented Lagrangian method

The Augmented Lagrangian method (AL-GA) was proposed by Adeli and Cheng [15,16]. According to this method the constrained problem is transformed to an unconstrained problem, by introducing two sets of penalty coefficients $\gamma[(\gamma_1,\gamma_2,\ldots,\gamma_{M+N})]$ and $\mu[(\mu_1,\mu_2,\ldots,\mu_{M+N})]$. The modified objective function, for the generation g, is defined as follows:

$$F'(s,\gamma,\mu)=\frac{1}{L_{\mathrm{f}}}F(s)+\frac{1}{2}\left\{\sum_{j=1}^{N}\gamma_j^{(g)}[(q_j-1+\mu_j^{(g)})^{+}]^2+\sum_{j=1}^{M}\gamma_{j+N}^{(g)}\left[\left(\frac{|d_j|}{|d_j^{\mathrm{a}}|}-1+\mu_{j+N}^{(g)}\right)^{+}\right]^2\right\} \tag{6}$$

where L_{f} is a factor for normalizing the objective function; q_j is a non-dimensional ratio related to the stress constraints of the jth element group (see Eqs. (18) and (19)); d_j is the displacement in the direction of the jth examined degree of freedom, while d_j^{a} is the corresponding allowable displacement; N, M correspond to the number of stress and displacement constraint functions, respectively. Furthermore

$$(q_j-1+\mu_j^{(g)})^{+}=\max(q_j-1+\mu_j^{(g)},0) \tag{7}$$

$$\left(\frac{|d_j|}{|d_j^{\mathrm{a}}|}-1+\mu_{j+N}^{(g)}\right)^{+}=\max\left(\frac{|d_j|}{|d_j^{\mathrm{a}}|}-1+\mu_{j+N}^{(g)},0\right) \tag{8}$$

The penalty coefficients are updated at each generation according to the expressions $\gamma_j^{(g+1)}=\beta\cdot\gamma_j^{(g)}$ and $\mu_j^{(g)}=\mu_j^{(g)}/\beta$, where $\mu_j^{(g+1)}=\mu_j^{(g)}+\max[\mathrm{con}_{j,\mathrm{ave}}^{(g)},-\mu_j^{(g)}]$ and $\mathrm{con}_{j,\mathrm{ave}}^{(g)}$ is the average value of the jth constraint function for the gth generation, while the initial values of γ's and μ's are set equal to three and zero, respectively. Coefficient β is taken equal to ten as recommended by Belegundu and Arora [17].

3.4. Segregated genetic algorithm

The basic idea of the segregated GA (S-GA) [18] is to use, as in the method of static penalties, two static penalty parameters instead of one. The two values of the penalty parameters are associated with two populations that have a different level of satisfaction of the constraints. Each of the groups corresponds to the best performing individuals with respect to the associated penalty parameter. The S-GA can be described as follows:

Step 0 initialization: Random generation of $2N$ designs. The objective functions of the designs $1,2,\ldots,N$ are evaluated using the p_h penalty parameter, while the remaining designs $N+1,\ldots,2N$ are evaluated using the p_ℓ penalty parameter.

Step 1 selection: An intermediate population of size N is created by selecting the best individuals from the two populations.

Step 2 generation: Generate N offsprings using the basic operators mutation and crossover. The parents are evaluated using the p_h penalty parameter while the offsprings using the p_ℓ. The process is then repeated by returning to *Step 1*.

This version was used in [18] for the minimal weight design problem of a composite laminated plate.

4. Evolution strategies

4.1. Basic evolution strategies

In the majority of cases ES were applied to continuous 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 Thierauf and Cai [19] have proposed a modified ES algorithm. The basic differences between discrete and continuous ES are focused on the mutation and the recombination operators. The multi-membered ES adopted in the current study uses three operators: recombination, mutation and selection operators that can be included in the algorithm 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 $\tilde{s}=[\tilde{s}_1,\tilde{s}_2,\ldots,\tilde{s}_n]^{\mathrm{T}}$ is first built by means of recombination. For discrete problems the following recombination cases can be used

$$\tilde{s}_i=\begin{cases}s_{a,i}\text{ or }s_{b,i}\text{ randomly} & (\mathrm{A})\\ s_{m,i}\text{ or }s_{b,i}\text{ randomly} & (\mathrm{B})\\ s_{bj,i} & (\mathrm{C})\\ s_{a,i}\text{ or }s_{bj,i}\text{ randomly} & (\mathrm{D})\\ s_{m,i}\text{ or }s_{bj,i}\text{ randomly} & (\mathrm{E})\end{cases} \tag{9}$$

$\tilde{s}_i$ is the ith component of the temporary parent vector $\tilde{s}$, $s_{a,i}$ 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. The vector s_m is not randomly chosen but is the best of the μ parent vectors in the current generation. In case C of Eq. (9), $\tilde{s}_i=s_{bj,i}$ 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.

Let as consider the temporary parent $s_{\mathrm{p}}^{(g)}$ of the generation g that produces an offspring $s_{\mathrm{o}}^{(g)}$ through the mutation operator as follows

$$s_{\mathrm{o}}^{(g)} = s_{\mathrm{p}}^{(g)} + z^{(g)} \tag{10}$$

where $z^{(g)} = [z_1^{(g)}, z_2^{(g)}, \ldots, z_n^{(g)}]^{\mathrm{T}}$ is a random vector. Mutation is understood to be random, purposeless events, which occur very rarely. 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 [19] 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 $n-\ell$, components of the randomly changed vector $z^{(g)}$ will have zero value. In other words, the terms of vector $z^{(g)}$ are derived from

$$z_i^{(g)} = \begin{cases} (\kappa+1)\delta s_i & \text{for } \ell \text{ randomly chosen components} \\ 0 & \text{for } n-\ell \text{ other components} \end{cases} \tag{11}$$

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

$$p(\kappa) = \frac{(\gamma)^{\kappa}}{\gamma!}\mathrm{e}^{-\gamma} \tag{12}$$

γ is the standard deviation as well as the mean value of the random number κ. The choice of ℓ depends on the size of the problem and it is usually taken as 1/5 of the total number of design variables. The ℓ components are selected using uniform random distribution in every generation according to Eq. (11).

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 produce λ offsprings $(\mu \leqslant \lambda)$ and the selection process defines a new population of μ individuals from the set of λ offsprings only.

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 $4n\mu/\lambda$ generations remains unchanged, (ii) when the mean value of the objective values from all parent vectors in the last $2n\mu/\lambda$ generations has not been improved by less than a given value $\varepsilon_{\mathrm{b}}(=0.0001)$, (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 $\varepsilon_{\mathrm{c}}(=0.0001)$, (iv) when the ratio μ_b/μ has reached a given value $\varepsilon_{\mathrm{d}}(=0.5\text{–}0.8)$ where μ_b is the number of the parent vectors in the current generation with the best objective function value.

4.2. Contemporary ES—the (μ,λ,θ) evolution strategies

This is a more general ES version, which was proposed by Schwefel and Rudolph [20] for application in continuous problems but has not been applied either to continuous or to discrete optimization problems [21]. The two schemes of the multi-membered evolution strategy, namely the $(\mu+\lambda)$-ES and the (μ,λ)-ES, differ in the way the parents of a new generation are selected. So far, only empirical results have shown that the 'plus' version performs better in structural optimization problems [7,19].

The (μ,λ)-ES version is in danger to diverge because the so far best position is not preserved within the generation cycle (the so-called non-elitist strategy). The 'comma' version implies that each parent can have children only once (duration of life: one generation or one reproduction cycle), whereas in the 'plus' version individuals may live eternally if no child achieves a better or at least the same improvement in the objective function value. The contemporary ES (C-ES) introduce a maximal life span of $\theta \geqslant 1$ reproduction cycles which gives the 'comma' scheme for $\theta = 1$ and the 'plus' one for $\theta = \infty$. If $\mu \geqslant 1$ is the number of parents, $\lambda > \mu$ is the number of offspring, then ρ with $1 \leqslant \rho \leqslant \mu$ is the number of ancestors for each descendant. This ES version differs in two points from the basic one: (i) free number of parents are involved in reproduction ranging from 1 to μ, (ii) a finite number of reproduction cycles per individual is performed, not one (1) or infinite (∞) for the 'comma' and the 'plus' schemes, respectively. The selection, mutation and recombination operators used in the C-ES are the same as described in the section of the basic evolution strategies.

4.3. Adaptive ES

The handling of the constraints by the basic ES is based on the death penalty approach [22], where every infeasible design point is discarded. Thus the process is directed to search only in the feasible region of the design space. Due to this approach many designs that are examined by the optimizer during the search process and are close to the acceptable design space are rejected leading to the loss of valuable information. The idea introduced in this work is to use soft constraints during the first stages of the search and as the search approaches the region of the global optimum the

constraints to become more severe until they reach their real values.

The implementation of adaptive ES (A-ES) in a structural optimization problem is straightforward and follows the same steps described in the section of the basic ES. The ES optimization procedure starts with a population of parent vectors, while a level of violation of the constraints is determined. If any of these parents corresponds to an infeasible design lying outside the extended design space then this parent is modified until it becomes 'feasible'. Then the offsprings are generated and they are also checked if they are in the 'feasible' region according the current level of violation. In every generation the values of the objective function are compared between the parent and the offspring vectors and the worst vectors are rejected, while the remaining ones are considered to be the parent vectors of the new generation. This procedure is repeated until the chosen termination criterion is satisfied.

In this adaptive scheme a nominal convergence check is adopted for the determination of the level of violation of constraints. Nominal convergence occurs when the mean value of the objective function of the designs of the current population is relatively close to the best design achieved until the current generation, according to the expression

$$\frac{\overline{F}^{(g)} - F^{(g)}_{\text{best}}}{\overline{F}^{(g)}} \leqslant \varepsilon_{\text{ad}} \tag{13}$$

where $\overline{F}^{(g)}$ is the mean objective function value and $F^{(g)}_{\text{best}}$ is the best objective function value of all parents in the gth generation, where $\varepsilon_{\text{ad}} = 0.05$.

The A-ES steps can be stated as follows:

1. *Initialization step:* Selection of s_i, $(i = 1, 2, \ldots, \mu)$ parent vectors of the design variables and the percentage of violation of the constraints v_0 (usually taken between 20% and 50%).
2. *Analysis step:* Solve $K(s_i)u_i = f \quad (i = 1, 2, \ldots, \mu)$, where K is the stiffness matrix of the structure and f is the loading vector.
3. *Constraints check:* All parent vectors become "feasible", within the prescribed level of constraints violation v_0.
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. *Nominal convergence check:* If nominal convergence has occurred the level of violation v_g becomes more severe by reducing its value by the quantity b (usually 0.1 or 0.2).
7. *Constraints check:* If satisfied according to the current level of violation v_g continue, else change s_j and return to *Step 4*.
8. *Selection step:* Selection of the next generation parents according to $(\mu + \lambda)$ or (μ, λ) selection schemes.
9. *Convergence check:* If satisfied stop, else return to *Step 3*.

4.4. An academic example

As an example for explaining the process of ES we consider the three-bar truss shown in Fig. 1, where the minimum volume is required to support a force P.

This structure has been used as a test bed in the literature of structural optimization [23] and must satisfy various constraints, such as member crushing, member buckling and failure by excessive deflection of node 4. The final design of the structure must be symmetric. Therefore, the following design variables are defined: A_1 = cross-sectional area of material for members 1 and 3, and A_2 = cross-sectional area of material for member 2. The relative merit of any design for the problem is measured in its material volume. Thus, the total material volume for the structure serves as an objective function:

$$\text{volume} = L(2\sqrt{2}A_1 + A_2)$$

where L is defined in Fig. 1.

To define the constraint functions for the problem, stresses and deflections for the structure are calculated. Using analysis procedures for statically indeterminate structures, horizontal and vertical displacements u and v, respectively, of the node 4 are given by

$$u = \frac{\sqrt{2}LP_u}{A_1E}$$

$$v = \frac{\sqrt{2}LP_v}{(A_1 + \sqrt{2}A_2)E}$$

where E is the modulus of elasticity for the material, while P_u and P_v are the horizontal and vertical components of the load P, respectively: $P_u = P\cos\theta$, $P_v =$

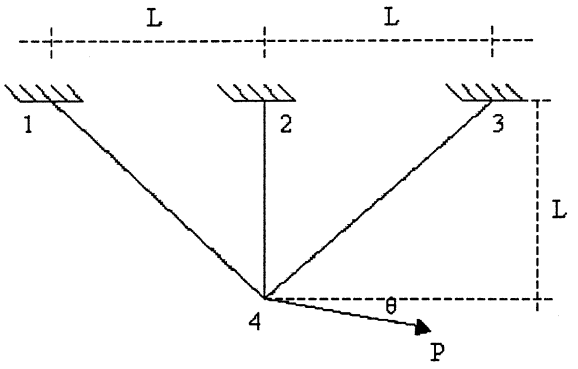


Fig. 1. Symmetric three-bar truss.

$P\sin\theta$. Stresses σ_1, σ_2 and σ_3 in members 1–3 under the load P are computed from member forces as follows:

$$\sigma_1 = \frac{1}{\sqrt{2}}\left[\frac{P_u}{A_1} + \frac{P_v}{(A_1 + \sqrt{2}A_2)}\right] \quad \text{(a)}$$

$$\sigma_2 = \frac{\sqrt{2}P_v}{(A_1 + \sqrt{2}A_2)} \quad \text{(b)}$$

$$\sigma_3 = \frac{1}{\sqrt{2}}\left[\frac{P_v}{(A_1 + \sqrt{2}A_2)} - \frac{P_u}{A_1}\right] \quad \text{(c)}$$

Note from Eqs. (a) and (c) that σ_1 is always greater than σ_3. Therefore, we need to impose constraints only on σ_1 and σ_2. If σ_a is an allowable stress for the material, then

$$\sigma_1 \leqslant \sigma_a \quad \text{and} \quad \sigma_2 \leqslant \sigma_a$$

Horizontal and vertical deflections of the node 4 must be within the specified limits Δu and Δv, respectively. Thus $u \leqslant \Delta u$ and $v \leqslant \Delta v$.

To impose buckling constraints for members under compression, the dependence of moment of inertia I on the cross-sectional area of the members must be specified. A form with quite general applicability is $I = \beta A^2$, where A is the cross-sectional area and is a non-dimensional constant. This relation follows if the shape of the cross-section is fixed and all its dimensions are varied in the same proportion. The axial force for the ith member is $F_i = A_i\sigma_i$, where $i = 1, 2, 3$ with tensile force taken as positive. Members of the truss are considered bars with pin ends. Therefore, the buckling load for the ith member is

$$P_b = \frac{\pi^2 E_i I_i}{\ell_i^2}$$

where ℓ_i is the length of the ith member. Buckling constraints are expressed as

$$-F_i \leqslant \frac{\pi^2 E_i I_i}{\ell_i^2}$$

The negative sign for F_i is used to make the left-hand side of the constraints negative when the member is in tension since there is no need to impose buckling constraints for members in tension. Substituting various quantities, member buckling constraints take the form

$$-\frac{1}{\sqrt{2}}\left[\frac{P_u}{A_1} + \frac{P_v}{(A_1 + \sqrt{2}A_2)}\right] \leqslant \frac{\pi^2 E\beta A_1}{\ell_1^2}$$

$$-\frac{\sqrt{2}P_v}{(A_1 + \sqrt{2}A_2)} \leqslant \frac{\pi^2 E\beta A_2}{\ell_2^2}$$

Table 1
Design data for the three-bar structure

Allowable stress	Members 1 and 3: $\sigma_{1a} = \sigma_{3a} = 34.5$ MPa Member 2: $\sigma_{2a} = 137.9$ MPa
Allowable displacements	$\Delta u = 0.0127$ cm $\Delta v = 0.0127$ cm
Modulus of elasticity	$E = 69.0E + 03$ MPa
Constant (assuming the sections to be thin)	$\beta = 1.0$
No. of design variables	$n = 2$
Lower limits of the design variables	(5.07, 5.07) cm^2
Upper limits of the design variables	(101.4, 101.4) cm^2
Parameter	$L = 2.54$ cm
Load	$P = 178$ kN
Angle	$\theta = 45°$

$$-\frac{1}{\sqrt{2}}\left[\frac{P_v}{(A_1 + \sqrt{2}A_2)} - \frac{P_u}{A_1}\right] \leqslant \frac{\pi^2 E\beta A_1}{\ell_1^2}$$

We will describe now step by step the ES generation of the first population from the initial one for the design data shown in Table 1. The members of the truss are taken to be tabular sections.

The selection scheme used in this application is the $(5 + 5)$ which means $\mu = 5$ parents produce $\lambda = 5$ offsprings selected according to the $(\mu + \lambda)$ scheme. The current difference between two adjacent values δs_i of Eq. (11) is taken 2.535 cm^2 and $\ell = 1$ (Eq. (11)). The recombination type D of Eq. (9) is used according to which the best member of the parent population is recombined with a randomly chosen member of the parent population. Termination criterion (ii) of ES is implemented according to which the procedure is terminated when the mean value of the objective values from all parent vectors in the last four generation has not been improved by less than the given value $\varepsilon_b = 0.001$.

Initial population

Member of the population	Design (A_1, A_2) (cm^2)	Volume (l)
1	(50.7, 50.7)	24.0
2	(71.0, 65.9)	33.0
3	(71.0, 71.0)	33.7
4	(40.6, 60.8)	21.7
5	(40.6, 35.5)	18.6

- Recombine member 5 with member 1 to obtain the first temporary offspring vector (40.6, 50.7). The mutation operator is performed in one of the two design variable since $\ell = 1$.

  To randomly select which one of the two design variable is to be mutated the following procedure

is followed: Generate a random number that follows the Gauss distribution in the (0,1): $r = 0.71471$.
Transform of the random variable r to an integer one:
$i1 = \text{int}(r \times (n+1)) + 1 = 3$
if $(i1.\text{ge}.(n+1))i1 = n+1$
$i1 = i1 - (n+2)/2 = 1$ if $(i1.\text{gt}.0)x_i = x_i + z_i$ else $x_i = x_i - z_i$
$i2 = \text{abs}(i1) = 1$ which means that the first design variable is to be mutated.
Mutate the design variable A_1 with $\kappa = 0$ [24] (Eq. (12)) to produce the first feasible offspring (43.1, 50.7) with objective function value equal to 21.4 l.

- Recombine member 5 with member 5 to obtain the second temporary offspring vector (40.6, 35.5). Mutate the design variable A_2 with $\kappa = 0$ to produce the corresponding feasible offspring (40.6, 38.0) with objective function value equal to 18.9 l.
- Recombine member 5 with member 4 to obtain the third temporary offspring vector (40.6, 60.8). Mutate the design variable A_1 with $\kappa = 0$ to produce the corresponding feasible offspring (38.0, 60.8) with objective function value equal to 20.9 l.
- Recombine member 5 with member 3 to obtain the fourth temporary offspring vector (71.0, 35.5). Mutate the design variable A_1 with $\kappa = 0$ to produce the corresponding feasible offspring (68.4, 35.5) with objective function value equal to 28.4 l.
- Recombine member 5 with member 1 to obtain the fifth temporary offspring vector (50.7, 35.5). Mutate the design variable A_2 with $\kappa = 1$ to produce the corresponding feasible offspring (50.7, 30.4) with objective function value equal to 21.5 l.

If an offspring happens to be infeasible then the recombination of the best member with a randomly chosen member is performed followed by the mutation step until a feasible design is achieved.

Offspring population

Member of the population	Design (A_1, A_2) (cm^2)	Volume (l)
1	(43.1, 50.7)	21.4
2	(40.6, 38.0)	18.9
3	(38.0, 60.8)	20.9
4	(68.4, 35.5)	28.4
5	(50.7, 30.4)	21.5

We then select among the $(\mu + \lambda)$ population pool the best μ individuals. Thus, the second population is as follows:

Second population

Member of the population	Design (A_1, A_2) (cm^2)	Volume (l)
1	(43.1, 50.7)	21.4
2	(40.6, 38.0)	18.9
3	(38.0, 60.8)	20.9
4	(68.4, 35.5)	28.4
5	(40.6, 35.5)	18.6

The procedure is repeated until the termination criteria is satisfied, for the current test example this is achieved for the total number of generations 14 and 86 total number of FE analyses. The best design achieved is the (30.4, 22.8) with objective function value equal to 13.5 l.

5. Gradient-based sizing optimization

5.1. Sensitivity analysis

Sensitivity analysis is the most important and time-consuming part of a gradient-based sizing optimization procedure. The methods for sensitivity analysis can be divided into discrete and variational methods [25]. In the discrete approach the derivatives or the sensitivities of the objective and constraint functions are evaluated using the finite element equations of the discretized structure. Generally the effort required for the computer implementation of the discrete methods is less than the effort required for the implementation of the variational methods.

A further classification of the discrete methods is the following [26]:

(i) *Global finite difference method:* A full finite element analysis has to be performed for each design variable and the accuracy of the method depends strongly on the value of the perturbation of the design variables. The global finite difference method (GFD) scheme is usually sensitive to the accuracy of the computed perturbed displacement vectors which is dependent on the magnitude of the perturbation of the design variables. The magnitude of this perturbation is usually taken between 10^{-3} and 10^{-5}.
(ii) *Semi-analytical method:* The stiffness matrix of the initial finite element solution is retained during the computation of the sensitivities. This provides an improved efficiency over the finite difference method by a relatively small increase in the algorithmic complexity. The accuracy problem involved in the numerical differentiation can be overcome by using the "exact" semi-analytical (ESA) method which needs more programming effort than the simple

semi-analytical one but it is computationally more efficient [27,28].

(iii) *Analytical method:* The finite element equations, the objective and constraint functions are differentiated analytically.

The decision on which method to implement depends strongly on the type of problem, the structure of the computer program and the access to the source code. The implementation of analytical and semi-analytical methods is more complex and requires access to the source code, whereas when a finite difference method is applied the formulation is much simpler and the sensitivity coefficients can be easily evaluated even with general purpose commercial codes. In the present investigation both the global finite difference method and the semi-analytical method have been used.

5.2. Sequential quadratic programming

SQP methods are the standard general purpose mathematical programming algorithms for solving NLP optimization problems [29]. Such methods make use of local curvature information derived from linearization of the original functions, by using their derivatives with respect to the design variables at points obtained in the process of optimization. Thus a quadratic programming (QP) model (or subproblem) is constructed from the initial NLP problem. A local minimizer is found by solving a sequence of these QP subproblems using a quadratic approximation of the objective function. Each subproblem has the following form

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}p^{\mathrm{T}}Hp + g^{\mathrm{T}}p \\ \text{with} & Ap + h(s) \leqslant 0 \\ & \bar{s}_\ell \leqslant p \leqslant \bar{s}_u \end{array} \tag{14}$$

where p is the search direction subjected to upper and lower bounds, g is the gradient of the objective function, A is the Jacobian of the constraints, usually the *active* ones only (i.e. those that are either violated, or not far from being violated), $\bar{s}_\ell = s_\ell - s$, $\bar{s}_u = s_u - s$ and H is an approximation of the Hessian matrix of the Lagrangian function

$$L(s,\lambda) = F(s) + \lambda h(s) \tag{15}$$

In Eq. (15) λ are the Lagrange multipliers under the non-negativity restriction ($\lambda \geqslant 0$) for the inequality constraints. In order to construct the Jacobian and the Hessian matrices of the QP subproblem the derivatives of the objective and constraint functions are required. These derivatives are computed during the sensitivity analysis phase.

There are two ways to solve this QP subproblem, either with a primal [30], or with a dual [31] formulation. In the present study a primal algorithm is employed based on an SQP algorithm from the NAG library [32]. The primal algorithm is divided into three phases: (i) the solution of the QP subproblem to obtain the search direction, (ii) the line search along the search direction, (iii) the update of the Hessian matrix H.

Once the direction vector is found a line search is performed, involving only the non-linear constraints, in order to produce a "sufficient decrease" to the merit function φ. This merit function is an augmented Lagrangian function of the form [31]

$$\varphi = F(s) - \sum_i \lambda_i (g_i(s) - \gamma_i) + \frac{1}{2}\sum_i p_i (g_i(s) - \gamma_i)^2 \tag{16}$$

where γ_i are the non-negative slack variables of the inequality constraints derived from the solution of the QP subproblem. These slack variables allow the active inequality constraints to be treated as equalities and avoid possible discontinuities. Finally, p_i are the penalty parameters which are initially set to zero and in subsequent iterations are increased whenever this is necessary in order to control the violation of the constraints and to ensure that the merit function follows a descent path [30].

To implement SQP in discrete optimization problems, a continuous SQP step is performed first at each step. Then the current continuous design reached is projected to the nearest discrete values of the design space. The current continuous design is considered as a lower bound of the projected discrete one [33].

6. The hybrid approach

Hybrid methods which combine evolutionary computation techniques with deterministic procedures for numerical optimization problems have been recently investigated. Papadrakakis et al. [34] used evolution strategies with the SQP method, while Waagen et al. [35] combined EP with the direction set method of Hooke and Jeeves [36]. The hybrid implementation proposed in [34] was found very successful on shape optimization test examples, while the method proposed in [35] was applied to unconstrained mathematical test functions. Myung et al. [37] considered a similar to Waagen et al. approach, but they experimented with constrained mathematical test functions. Myung et al. combined a floating-point EP technique, with a method-developed by Maa and Shanblatt [38] applied to the best solution found by the EP technique. The second method iterates until the system defined by the combination of the objective function, the constraint functions and the design variables reach equilibrium.

A characteristic property of the SQP based optimizers is that they capture very fast the right path to the

nearest optimum, irrespective of its nature a local or global optimum. However, after locating the area of this optimum it might oscillate until all constraints are satisfied since it is observed that even small constraint violations often slow down the convergence rate of the method. On the other hand EA proceed with slower rate, due to their random search, but the absence of strict mathematical rules, which govern the convergence rate of the mathematical programming methods, make EA less vulnerable to local optima and therefore it is much more likely to converge towards the global optimum in non-convex optimization methods. These two facts give the motivation to combine EA with MP methodologies. Between the two EA examined in this study the genetic algorithms seems to be faster than evolution strategies since they do not always operate on the feasible region of the design space as evolution algorithms. However, they are most often found unable to converge to feasible designs.

In order to benefit from the advantages of both methodologies (EA and SQP) a hybrid approach is proposed, which combines the two methods in an effort to increase the robustness and the computational efficiency of the optimization procedure. The optimization process is divided into two separate phases. During the first phase, the evolution algorithm optimizes the objective function. After the termination of this phase, the second phase starts by applying the SQP to the best solution found during the first phase.

The procedure of EA is first used in order to locate the region where the global optimum lies, then the SQP is activated in order to exploit its higher rate of convergence in the neighborhood of the optimum. The switch from EA to SQP is performed when EA reaches the vicinity of an optimum that is considered a good one. This approach appears to be more rational in the general case when more complex and non-convex design problems are to be solved with many local optima. In the case of GA-SQP the final design achieved by GA can be tolerated to be infeasible since the SQP will eventually locate a feasible design.

In the present study two hybrid methodologies are examined which are based on the combination of GA and ES with the SQP algorithm. The transition from one algorithm to the other is performed when there is a small difference between the best designs of two consecutive generations

$$\left|\frac{f_{j+1}-f_j}{f_j}\right| \leqslant \varepsilon \tag{17}$$

It was found in the test cases performed in this study and in shape optimization problems [34] that a reasonable value for ε could be taken in the vicinity of 10%.

7. Numerical tests

The optimization of two space frame structures is selected to illustrate the efficiency of the presented methodologies in structural design problems. In both test examples the modulus of elasticity E is taken 200 GPa and the yield stress σ_y is 250 MPa. The cross-section of each member is assumed to be a I-shape and for each member two design variables are considered as shown in Fig. 2. The values of b and h are selected from an integer design space, while t and w are given as follows: $f = 0.06h + 0.10(b-10)$, $w = 0.625f$. Those two expressions make sure that the web thickness is less than b, the opposite of which would have been not acceptable. The objective function of the problems is the weight of the structure. The constraints are the member stresses and the inter-storey drifts. For rigid frames in rolled I-shapes, under allowable stress design requirements specified by Eurocode 3 [39], the stress constraints are defined by the non-dimensional ratio q of interaction formulas

$$q = \frac{f_a}{F_a} + \frac{f_b^y}{F_b^y} + \frac{f_b^z}{F_b^z} \leqslant 1.0 \quad \text{if } \frac{f_a}{F_a} \leqslant 0.15 \tag{18}$$

and

$$q = \frac{f_a}{0.60\sigma_y} + \frac{f_b^y}{F_b^y} + \frac{f_b^z}{F_b^z} \leqslant 1.0 \quad \text{if } \frac{f_a}{F_a} > 0.15 \tag{19}$$

where f_a is the computed compressive axial stress, f_b^y, f_b^z are the computed bending stresses for y- and z-axis, respectively. F_a is the allowable compressive axial stress,

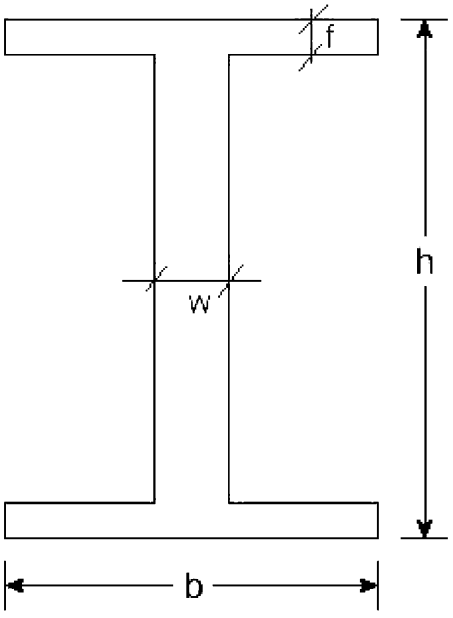


Fig. 2. I-shaped cross-section design variables.

F_b^y, F_b^z are the allowable bending stresses for y- and z-axis, respectively, and σ_y is the yield stress of the steel. The allowable inter-storey drift is limited to 1.5% of the height of each storey. One load case is considered in all examples.

The termination criterion adopted for both GA and ES is the same for comparison reasons. If no improvement of the best value of the objective function has occurred in the last six generations, then the optimization procedure is terminated.

Example 1: The first example is a six-storey space frame, first studied by Orbinson et al. [40], with 63 elements and 180 degrees of freedom. The beams length is $L_1 = 7.32$ m and the columns height is $L_2 = 3.66$ m. The loads consisting of 17 kPa gravity load on all floor levels and a lateral load of 100 kN applied at each node in the front elevation in the z direction. The element members are divided into five groups shown in Fig. 3 and the total number of design variables is 10. The initial design used in this example was chosen away from the optimum corresponding to the weight of 2846 kN for every test. Table 2 shows the performance of the two types of the basic multi-membered ES, namely $(\mu + \lambda)$-ES and (μ, λ)-ES for $\mu = \lambda = 5$ and $\mu = \lambda = 10$. The best design achieved by these tests is used as the basis for comparison in subsequent applications of the optimization methods examined.

In Figs. 4–7 various techniques for handling the constraints of the genetic algorithms are presented. The

Table 2
Test example 1—performance of the two selection schemes $(\mu + \lambda)$-ES and (μ, λ)-ES

Optimizer	Weight (kN)	FE analyses	Time (s)
$(5 + 5)$-ES	675	416	177
(5,5)-ES	677	430	183
$(10 + 10)$-ES	668	847	363
(10,10)-ES	661	834	357

performance of the methods for handling the constraints is measured with two parameters: the weight achieved and the average level of violation of the constraint functions. The average percentage violation of the constraint functions is abbreviated to *Avg. Const. Violation* (%). Fig. 4 depicts the performance of GA with the method of static penalties for handling the constraints. It can be seen that the performance of the method is very sensitive to the value of the penalty parameter, while there is not a rule of thumb on how to choose this single penalty parameter. If it is chosen too small the search will converge to an infeasible solution otherwise if it is chosen too large a feasible solution may be located but it would be far from the global optimum. Fig. 5 presents the performance of GA with the method of dynamic penalties for handling the constraints. It appears that the constraint violation percentage in this case is equal to zero for all cases considered, since for high generation number the $(c \cdot g)^{\alpha}$ component of the penalty term takes

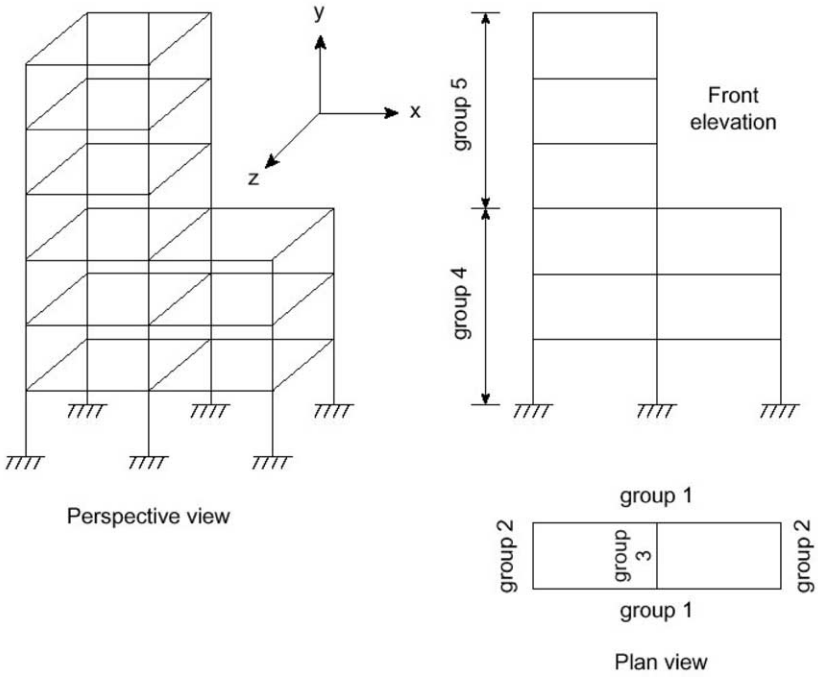
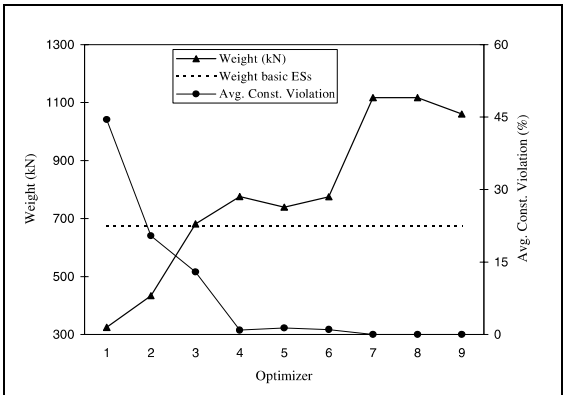
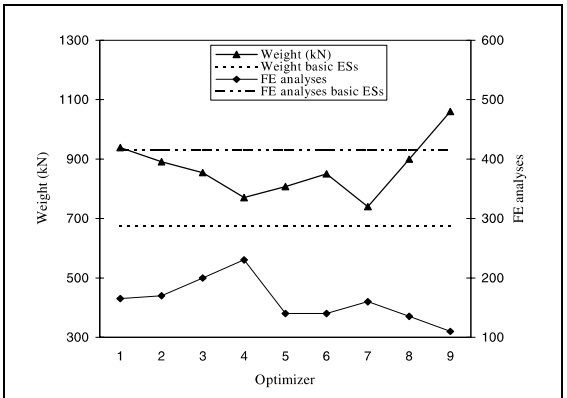


Fig. 3. Six storey space frame.



Optimizer	Penalty Parameter p	FE analyses
1	0.2	275
2	1	330
3	5	115
4	50	230
5	100	275
6	200	150
7	500	85
8	3000	85
9 (μGA)	100	110

Fig. 4. Test example 1—performance of GA with static penalties.

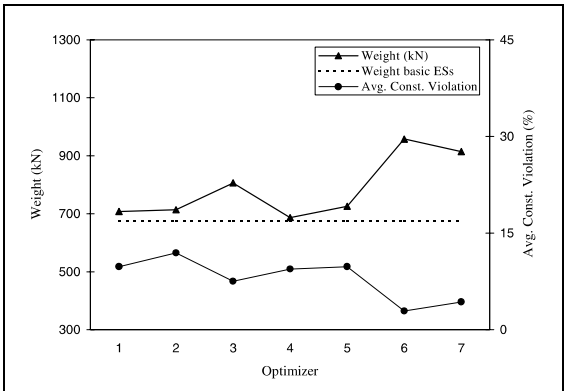


Optimizer	Parameters (c, β)
1	(0.5, 1)
2	(1, 1)
3	(0.5, 2)
4	(1, 2)
5	(1.2, 2)
6	(1.5, 2)
7	(2, 2)
8	(2.1, 2)
9 (μGA)	(2, 2)

Fig. 5. Test example 1—performance of GA with dynamic penalties ($\alpha = 2$, zero constr. violation).

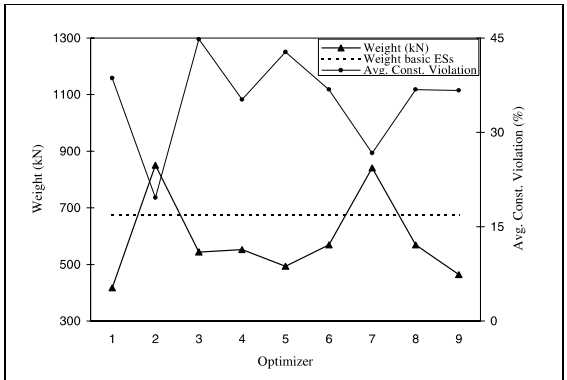
large values, which makes even the slightly violated designs not to be selected in subsequent generations. Thus, the system has little chances to escape from local optima.

Fig. 6 presents the performance of the segregated GA (S-GA) method for handling the constraints for different values of penalty parameters p_h, p_ℓ. The results indicate



Optimizer	Penalty Parameters (p_h, p_ℓ)	FE analyses
1	(5, 0.5)	110
2	(10, 0.5)	130
3	(20, 0.5)	100
4	(3, 1)	135
5	(25, 1)	85
6	(50, 1)	70
7 (μS-GA)	(4, 0.4)	80

Fig. 6. Test example 1—performance of S-GA with p_h, p_ℓ scheme.



Optimizer	Penalty Parameters (p_ℓ, p_h)	FE analyses
1	(0.4, 1)	115
2	(0.4, 50)	50
3	(0.5, 5)	85
4	(0.5, 10)	75
5	(0.5, 25)	75
6	(0.5, 15)	75
7	(0.3, 100)	50
8	(0.6, 20)	75
9	(1, 25)	75

Fig. 7. Test example 1—performance of S-GA with p_ℓ, p_h scheme.

that the optimum designs achieved are much closer to the global one, compared to the other constraint handling techniques, but are still in many cases local optima. A general comment that can be deduced from Figs. 4–6 for micro GA is that μ-GA show premature convergence to worse local optima compared to the corresponding basic GA. Fig. 7 depicts the performance of the S-GA method when the parents are evaluated using the p_ℓ penalty parameter and the offsprings using the p_h.

Tables 3 and 4 contain the results of the Augmented Lagrangian GA method (AL-GA). The constraint violation percentage is equal to zero for all tests considered. Table 3 depicts the results of the method with the termination criteria suggested by Adeli and Cheng in Ref. [15], while Table 4 includes the results obtained for the termination criteria used for the rest of the methods. For the adopted termination criterion we have examined four different cases, according to the allowable number of generations with no improvement of the objective function. It can be seen that both termination criteria converge to the same result which appears to be a local minima compared to the minimum achieved by the ES as shown in Table 2. It can also be seen that the Augmented Lagrangian GA method is not affected by the value of normalization parameter L_f.

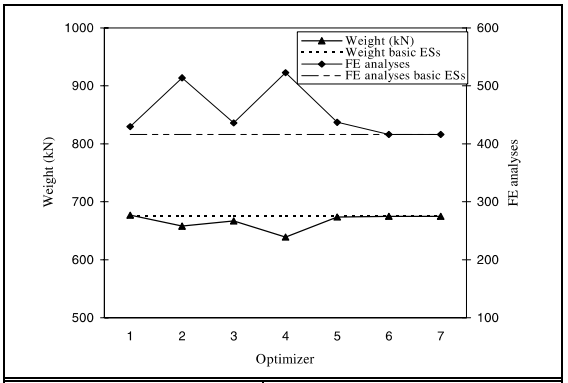
Table 3
Test example 1—performance of the AL-GA

L_f	Weight (kN)	FE analyses	Time (s)
100	1010	145	62
500	1010	145	62
1000	1010	145	62

Termination criterion of Ref. [15].

Table 4
Test example 1—performance of the AL-GA with the termination criterion adopted in this work

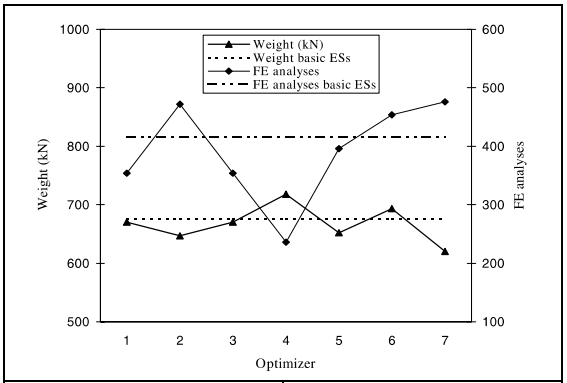
L_f	Weight (kN)	FE analyses	Time (s)
Case a: $6\mu/\lambda$			
100	1060	110	48
500	1060	110	48
1000	1060	110	48
Case b: $12\mu/\lambda$			
100	1042	140	60
500	1042	140	60
1000	1042	140	60
Case c: $18\mu/\lambda$			
100	1010	145	62
500	1010	145	62
1000	1010	145	62
Case d: $24\mu/\lambda$			
100	1010	145	62
500	1010	145	62
1000	1010	145	62



Optimizer	Parameters (μ,λ,θ)
1	(5,5,1)
2	(5,5,2)
3	(5,5,4)
4	(5,5,6)
5	(5,5.8)
6	(5,5,10)
7	(5,5,15)

Fig. 8. Test example 1—performance of the C-ES.

Figs. 8 and 9 show the performance of the two modified versions of the evolution strategies, namely the contemporary ES (C-ES) and the adaptive ES (A-ES), respectively. For this test example the reduction parameter b used in the A-ES method is equal to 0.1, while the parameters α_i and α_f denote the initial and final level



Optimizer	Parameters (ε_{ad}, α_i, α_f)
1	(0.005, 1.2, 1.0)
2	(0.005, 1.6, 1.6)
3	(0.01, 1.2, 1.0)
4	(0.03, 1.4, 1.0)
5	(0.03, 1.6, 1.5)
6	(0.05, 1.2, 1.0)
7	(0.05, 1.6, 1.0)

Fig. 9. Test example 1—performance of the A-ES ($b = 0.1$).

of violation of the constraints. The initial parameter α_i for the normalized constraint functions (Eqs. (18) and (19)) is taken between 1.2 and 1.6, which correspond to a percentage violation of the constraints between 20% and 60%. In the case of Fig. 8 all designs achieved are feasible, while in the case of Fig. 9 the designs achieved by the optimizers 2 and 5 are not feasible. The results of Figs. 8 and 9 indicate that the two new versions of ES manage to converge to better designs than the basic ES for a number of different parameters used at a marginal increase of computational effort. The C-ES method appears to be slightly more robust than the adaptive one, which may converge to infeasible designs when the parameter α_f is not equal to one. The A-ES, however, manage to converge to the best optimum design of 620 kN in 470 FE analyses for $\varepsilon_{ad} = 0.005$ and $\alpha_i = 1.6$.

The performance of the hybrid approaches is depicted in Table 5. The results indicate that despite the fact that all GA methods, except of AL-GA, converge to infeasible designs (the letter 'v' denotes constraint violation) the SQP optimizer which follows manages to reach a feasible one at a competitive time. The best design found by A-ES-SQP and AL-GA-SQP optimizer is 20% better than the design achieved by SQP requiring 35% and 25% less computational effort, respectively. We also examined two EA hybrid methods by combining AL-GA with ES and vice versa. Both perform well in terms of the design achieved and the required computing time. Furthermore, the modified version of the μGA, which excludes all the infeasible designs gave the same solution as the (μ, λ)-ES with 25% less computational effort, while the modified micro GA (m μGA) achieves better performance in terms of computing time compared to the $(\mu + \lambda)$-ES method. Comparing the performance of the two sensitivity analysis methods it can be seen that the ESA sensitivity analysis method is faster than GFD method. For this test example the ESA and GFD sensitivity analysis methods are used to compute the sensitivities and magnitude of perturbation is equal to 10^{-5}.

Example 2: The second example is the 20-storey space frame, studied by Papadrakakis and Papadopoulos [41], with 1020 members and 2400 degrees of freedom. The loads considered here are uniform vertical forces applied at joints equivalent to uniform load of 4.8 kPa and horizontal forces equivalent to uniform forces of 1.0 kPa on the largest surface. The element members are divided into 11 groups shown in Fig. 10 and the total number of design variables is 22. The initial design used in this example was chosen away from the optimum corresponding to the weight of 42,248 kN for every test.

Table 6 shows the performance of the two types of the multi-membered ES, namely $(\mu + \lambda)$-ES and (μ, λ)-ES for $\mu = \lambda = 5$ and $\mu = \lambda = 10$. The $(10 + 10)$ version of ES manages to converge to the best design, which is used as reference design. In Figs. 11 and 12 various techniques for handling the constraints by the GA are presented. The average percentage violation of the constraint functions is abbreviated as previously to *Avg. Const. Violation* (%). It can be seen that all feasible solutions achieved appear to be local optima although for this example the GA with dynamic penalties showed a rather more robust behaviour.

Tables 7 and 8 contain the results of the Augmented Lagrangian GA method (AL-GA). The constraint violation percentage is equal to zero for all tests considered. Table 7 depicts the results of the method with the ter-

Table 5
Test example 1—combination GA-SQP and ES-SQP

Optimizer	Initial design for second optimizer	Final design (kN)	Sensitivity analysis	FE analyses		Time (s)		
				EA	SQP	EA	SQP	Total
SQP	–	795	GFD	–	272	–	340	340
SQP	–	795	ESA	–	271	–	204	204
S-GA-SQP	708v	672	GFD	110	86	47	107	154
S-GA-SQP	708v	672	ESA	110	86	47	66	113
S-GA-SQP	635v	675	GFD	120	155	51	191	242
S-GA-SQP	635v	675	ESA	120	156	51	113	164
AL-GA-SQP	1010	663	GFD	145	121	62	155	217
AL-GA-SQP	1010	663	ESA	145	121	62	95	157
C-ES-SQP	912	681	GFD	212	114	90	142	232
C-ES-SQP	912	681	ESA	212	114	90	88	178
A-ES-SQP	829	663	GFD	136	96	58	120	178
A-ES-SQP	829	663	ESA	136	96	58	74	132
AL-GA-ES	1010	663	–	145 + 207	–	151	–	151
ES-AL-GA	829	675	–	136 + 257	–	177	–	177
m μGA	–	677	–	319	–	154	–	154

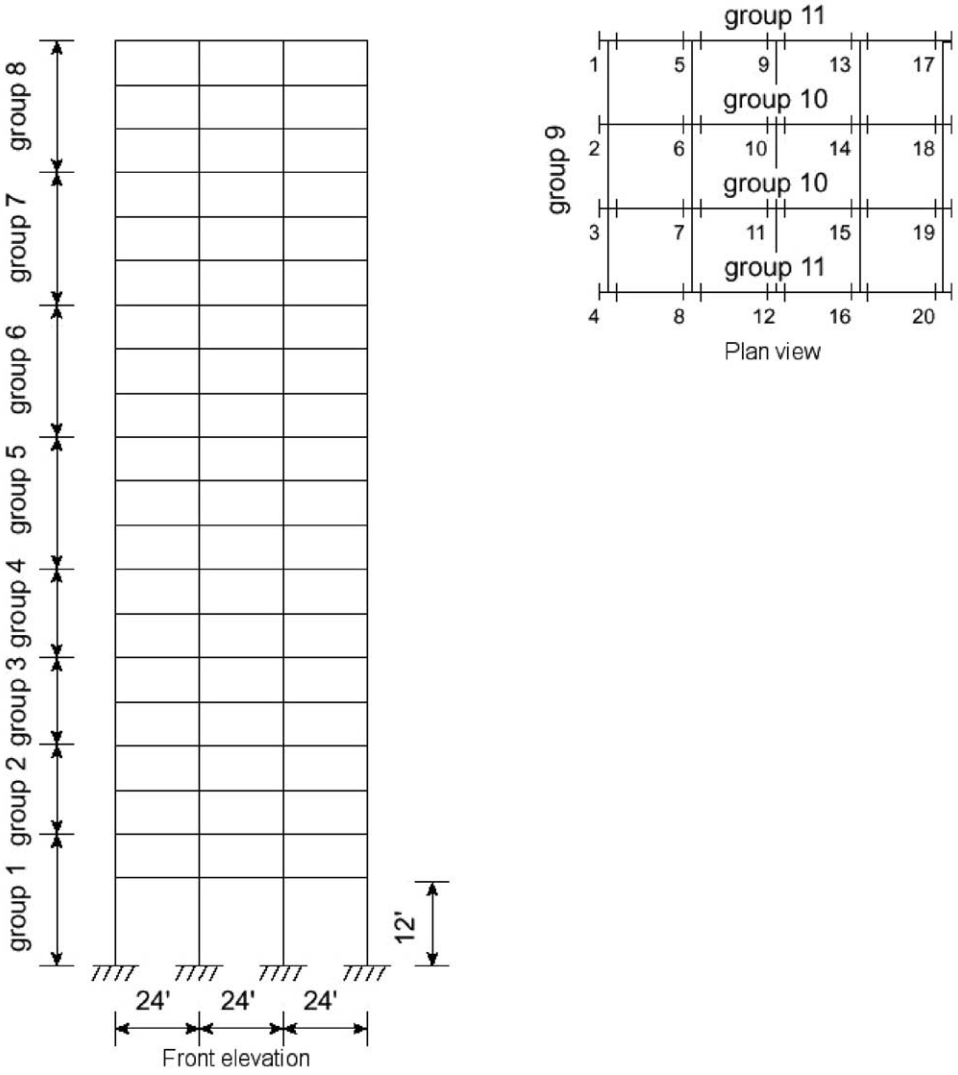


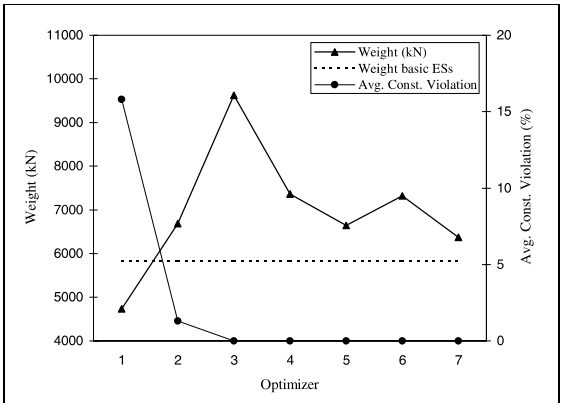
Fig. 10. Twenty storey space frame.

Table 6
Test example 2—performance of the two selection schemes $(\mu + \lambda)$-ES and (μ, λ)-ES

Optimizer	Weight (kN)	FE analyses	Time (s)
(5 + 5)-ES	6028	240	3708
(5,5)-ES	6085	452	6984
(10 + 10)-ES	5819	422	6519
(10,10)-ES	5877	727	11,230

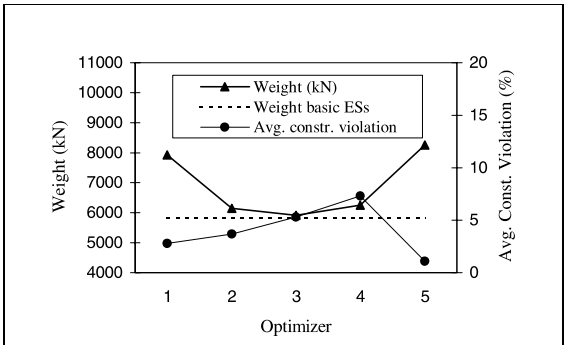
mination criteria suggested by Adeli and Cheng in Ref. [15], while Table 8 includes the results obtained for the termination criteria used for the rest of the methods. For the adopted termination criterion we have examined four different cases according to the allowable number of generations with no improvement of the objective function value. It can be seen that both termination criteria converge to the same result which appears to be a local minimal compared to the minimum achieved by the ES as shown in Table 6. It can also be seen, as in the previous example, that the Augmented Lagrangian GA method is not affected by the value of normalization parameter L_f.

Figs. 13 and 14 depict the performance of the two versions of the evolution strategies, namely the contemporary ES (C-ES) and the adaptive ES (A-ES), respectively, all designs depicted in those two figures are feasible. A comparison of the results of Table 6 and those depicted in Figs. 13 and 14 indicates that the two new methodologies outperform in most cases the basic



Optimizer	Parameters	FE analyses
1	p=1	360
2	p=50	290
3	p=500	120
4	c=0.5	220
5	c=1	260
6	c=1.5	150
7	c=2	190

Fig. 11. Test example 2—optimizer 1–3: GA with static penalties; optimizer (4–7) GA and dynamic penalties ($\alpha = \beta = 2$).



Optimizer	Penalty Parameters (p_h, p_ℓ)	FE analyses
1	(5, 0.5)	65
2	(10,0.5)	141
3	(2,1)	147
4	(25,1)	93
5	(50,1)	77

Fig. 12. Test example 2—performance of S-GA with p_ℓ, p_h scheme.

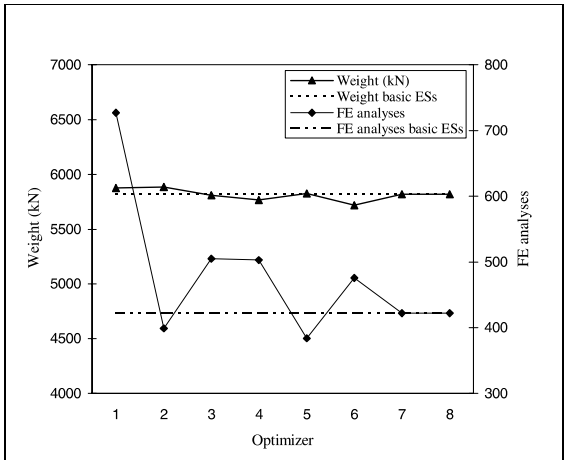
Table 7
Test example 1—performance of the AL-GA

L_f	Weight (kN)	FE analyses	Time (s)
100	7015	195	3097
500	7015	195	3097
1000	7015	195	3097

Termination criterion of Ref. [15].

Table 8
Test example 1—performance of the AL-GA with the termination criterion adopted in this work

L_f	Weight (kN)	FE analyses	Time (s)
Case a: $6\mu/\lambda$			
100	8073	120	1917
500	8073	120	1917
1000	8073	120	1917
Case b: $12\mu/\lambda$			
100	7397	140	2231
500	7397	140	2231
1000	7397	140	2231
Case c: $18\mu/\lambda$			
100	7015	195	3097
500	7015	195	3097
1000	7015	195	3097
Case d: $24\mu/\lambda$			
100	7015	195	3097
500	7015	195	3097
1000	7015	195	3097

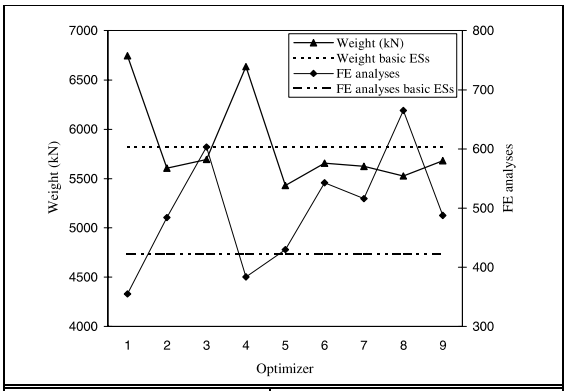


Optimizer	Parameters (μ,λ,θ)
1	(10,10,1)
2	(10,10,2)
3	(10,10,4)
4	(10,10,6)
5	(10,10,8)
6	(10,10,10)
7	(10,10,15)
8	(10,10,20)

Fig. 13. Test example 2—performance of the C-ES.

version of evolution strategies in terms of the achieved optimum weight at the expense of more computing time.

The performance of the hybrid approaches is depicted in Tables 9–11. Three initial designs are consid-



Optimizer	Parameters (ε_{ad}, α_i, α_f)
1	(0.01, 1.2, 1.0)
2	(0.01, 1.4, 1.0)
3	(0.01, 1.6, 1.0)
4	(0.03, 1.2, 1.0)
5	(0.03, 1.4, 1.0)
6	(0.03, 1.6, 1.0)
7	(0.05, 1.2, 1.0)
8	(0.05, 1.4, 1.0)
9	(0.05, 1.6, 1.0)

Fig. 14. Test example 2—performance of the A-ES ($b = 0.1$).

ered, one close, one medium and one away from the optimum. It can be seen that the computing time spent by the optimizers is affected by the initial design, especially in the case of the SQP approach. Furthermore, GA-SQP, ES-SQP and AL-GA-SQP optimizers manage to converge to final designs with 10% less weight than the SQP optimizer with less computational effort. We have also examined a EA hybrid method by combining AL-GA with ES, which perform well in terms of the design achieved and the required computing time. For this test example the ESA and GFD sensitivity analysis methods are used to compute the sensitivities and magnitude of perturbation is equal to 10^{-5}.

8. Conclusions

In this work two versions of evolution strategies as well as various methods for handling the constraints in GA have been discussed and compared in structural sizing optimization problems with the standard mathematical programming approach of SQP. The techniques for handling the constraints in GA are based on the use of penalty functions, which transform the constraint optimization problem into an unconstraint one. Techniques for handling the constraints based on static and dynamic penalties as well as the micro GA (μGA) and segregated GA (S-GA) are examined and tested. It appears that the efficiency of GA in structural sizing optimization problems is sensitive to the values of the characteristic parameters for handling the constraints. The computational effort that is required by GA is less than the corresponding effort by ES but they are hindered by premature convergence either to non-optimal or to infeasible designs. The contemporary ES (C-ES) and the adaptive ES (A-ES), did manage to improve the final design achieved by the basic ES for a number of values of the characteristic parameters at almost no increase of the computational effort.

The proposed hybrid optimization approach proved to be a robust and efficient method for structural optimization. Both GA-SQP and ES-SQP manage to converge to better designs than those achieved by ES or SQP alone at a reduced computational effort compared to the SQP procedure. The combination of GA with SQP is particularly promising in bad initial designs due

Table 9
Test example 2—hybrid methods

Optimizer	Initial design for second optimizer	Final design (kN)	Sensitivity analysis	Time (s)		Time (s)		
				EA	SQP	EA	SQP	Total
SQP	–	6427	GFD	–	435	–	21,932	21,932
SQP	–	6427	ESA	–	438	–	13,655	13,655
S-GA-SQP	7928*v*	5764	GFD	65	116	1100	5166	6266
S-GA-SQP	7928*v*	5764	ESA	65	117	1100	3608	4708
AL-GA-SQP	7015	6427	GFD	195	99	3301	4415	7716
AL-GA-SQP	7015	6427	ESA	195	99	3301	3077	6378
C-ES-SQP	7132	5834	GFD	193	152	3266	6770	10,036
C-ES-SQP	7132	5834	ESA	193	152	3266	4687	7953
A-ES-SQP	6531	5713	GFD	107	111	1811	4943	6754
A-ES-SQP	6531	5713	ESA	107	110	1811	3392	5202
AL-GA-ES	7015	5819	–	195 + 65	–	4401	–	4401
m μGA	–	5772	–	395	–	6117	–	6117

Bad initial design.

Table 10
Test example 2—hybrid methods

Optimizer	Initial design for second optimizer	Final design (kN)	Sensitivity analysis	Time (s)		Time (s)		
				EA	SQP	EA	SQP	Total
SQP	–	6119	GFD	–	385	–	17,519	17,519
SQP	–	6119	ESA	–	387	–	12,097	12,097
S-GA-SQP	7669	5695	GFD	40	169	681	7391	8071
S-GA-SQP	7669	5695	ESA	40	169	681	5291	5972
AL-GA-SQP	6816	5695	GFD	155	98	2639	4361	7000
AL-GA-SQP	6816	5695	ESA	155	98	2639	3057	5696
C-ES-SQP	6835	5764	GFD	171	111	2901	4943	7844
C-ES-SQP	6835	5764	ESA	171	111	2901	3469	6370
A-ES-SQP	6319	5764	GFD	91	67	1552	2983	4535
A-ES-SQP	6319	5764	ESA	91	67	1552	2089	3641
AL-GA-ES	6816	5430	–	155 + 73	–	3866	–	3866
m μGA	–	5472	–	339	–	5243	–	5243

Medium initial design.

Table 11
Test example 2—hybrid methods

Optimizer	Initial design for second optimizer	Final design (kN)	Sensitivity analysis	Time (s)		Time (s)		
				EA	SQP	EA	SQP	Total
SQP	–	6119	GFD	–	259	–	11,508	11,508
SQP	–	6119	ESA	–	259	–	8049	8049
S-GA-SQP	7669	5695	GFD	35	169	590	7391	7981
S-GA-SQP	7669	5695	ESA	35	169	590	5291	5881
AL-GA-SQP	6816	5695	GFD	115	98	1934	4361	6295
AL-GA-SQP	6816	5695	ESA	115	98	1934	3057	4991
C-ES-SQP	6835	5764	GFD	158	111	2685	4943	7628
C-ES-SQP	6835	5764	ESA	158	111	2685	3469	6154
A-ES-SQP	6275	5764	GFD	69	67	1198	2983	4181
A-ES-SQP	6275	5764	ESA	69	67	1198	2089	3287
AL-GA-ES	6816	5430	–	115 + 73	–	3161	–	3161
m μGA	–	5472	–	339	–	5243	–	5243

Good initial design.

to the fast convergence of GA towards the neighborhood of the optimum and the property of SQP to compute quickly the nearest optimum once in the neighborhood of the solution. However, the proposed adaptive ES when coupled with SQP and the combination of the Augmented Lagrangian GA, as the first stage optimizer followed by ES, proved to be the more efficient optimization algorithms.

References

[1] Fogel LJ, Owens AJ, Walsh MJ. Artificial intelligence through simulated evolution. New York: Wiley; 1966.
[2] Fogel DB. Evolving artificial intelligence. PhD thesis, University of California, San Diego, 1992.
[3] Goldberg DE. Genetic algorithms in search, optimization and machine learning. Reading, Massachusetts: Addison-Wesley Publishing Co., Inc.; 1989.
[4] Holland J. Adaptation in natural and artificial systems. Ann Arbor, MI: University of Michigan Press; 1975.
[5] Rechenberg I. Evolution strategy: optimization of technical systems according to the principles of biological evolution. Stuttgart: Frommann-Holzboog; 1973 (in German).
[6] Schwefel HP. Numerical optimization for computer models. Chichester, UK: Wiley & Sons; 1981.
[7] Papadrakakis M, Lagaros ND, Thierauf G, Cai J. Advanced solution methods in structural optimization

based on evolution strategies. J Eng Comput 1998;15(1):12–34.
[8] Adeli H, Cheng NT. Concurrent genetic algorithms for optimization of large structures. J Aerospace Eng ASCE 1994;7(3):276–96.
[9] Papadrakakis M, Tsompanakis Y, Hinton E, Sienz J. Advanced solution methods in topology optimization and shape sensitivity analysis. J Eng Comput 1996;13(5):57–90.
[10] Barricelli NA. Numerical testing of evolution theories. ACT A Biotheoretica 1962;16:69–126.
[11] Krishnakumar K. Micro genetic algorithms for stationary and non stationary function optimization. SPIE Proceedings Intelligent control and adaptive systems, vol. 1196, 1989.
[12] Goldberg DE. Sizing populations for serial and parallel genetic algorithms. TCGA Report No. 88004, University of Alabama, 1988.
[13] Joines J, Houck C. On the use of non-stationary penalty functions to solve non-linear constrained optimization problems with GA. In: Michalewicz Z, Schaffer JD, Schwefel H-P, Fogel DB, Kitano H, editors. Proceedings of the First IEEE International Conference on Evolutionary Computation. IEEE Press; 1994. p. 579–84.
[14] Michalewicz Z. Genetic algorithms, numerical optimization and constraints. In: Eshelman LJ, editor. Proceedings of the 6th International Conference on Genetic Algorithms. Los Altos, CA: Morgan Kaufmann; 1995. p. 151–8.
[15] Adeli H, Cheng NT. Integrated genetic algorithm for optimization of space structures. J Aerospace Eng ASCE 1993;6(4):315–28.
[16] Adeli H, Cheng NT. Augmented Lagrangian genetic algorithm for structural optimization. J Aerospace Eng ASCE 1994;7(1):104–18.
[17] Belegundu AD, Arora JS. A computational study of transformation methods for optimal design. AIAA J 1984;22(4):535–42.
[18] Le Riche RG, Knopf-Lenoir C, Haftka RT. A segregated genetic algorithm for constrained structural optimization. In: Eshelman LJ, editor. Proceedings of the 6th International Conference on Genetic Algorithms, 1995. p. 558–65.
[19] Cai J, Thierauf G. Discrete structural optimization using evolution strategies. In: Topping BHV, Khan AI, editors. Neural networks and combinatorial in civil and structural engineering. Edinburgh: Civil–Comp Ltd; 1993. p. 95–100.
[20] Schwefel H-P, Rudolph G. Contemporary evolution strategies. In: Morgan F, Moreno A, Merelo JJ, Chacion P, editors. Advances in artificial life, Proceedings of the Third European Conference on Artificial Life Granada, Spain, 4–6 June. Berlin: Springer; 1995. p. 893–907.
[21] Rudolph G. Personal communication, 1999.
[22] Back T, Hoffmeister F, Schwefel H-P. A survey of evolution strategies. In: Belew RK, Booker LB, editors. Proceedings of the 4th International Conference on Genetic Algorithms. Los Altos, CA: Morgan Kaufmann; 1991. p. 2–9.
[23] Schmit LA. Structural design by systematic synthesis. Proceedings of the second ASCE Conference on Electronic Computations, Pittsburgh, 1960. p. 105–22.
[24] Naylor TH, Balintfy JL. In: Computer simulation techniques. New York: John Wiley & Sons; 1966. p. 114.
[25] Bletzinger KU, Kimmich S, Ramm E. Efficient modelling in shape optimal design. Comput Syst Eng 1991;2(5/6):483–95.
[26] Hinton E, Sienz J. Aspects of adaptive finite element analysis and structural optimization. In: Topping BHV, Papadrakakis M, editors. Advances in Structural Optimization. Edinburgh: Civil–Comp Press; 1994. p. 1–26.
[27] Olhoff N, Rasmussen J, Lund E. Method of exact numerical differentiation for error estimation in finite element based semi-analytical shape sensitivity analyses. Special Report No. 10, Institute of Mechanical Engineering, Aalborg University, Aalborg, DK, 1992.
[28] Hinton E, Sienz J. Studies with a robust and reliable structural shape optimization tool. In: Topping BHV, editor. Developments in computational techniques for structural engineering. Edinburgh: Civil-Comp Press; 1995. p. 343–58.
[29] Gill PE, Murray W, Wright WH. Practical optimization. London: Academic Press; 1981.
[30] Gill PE, Murray W, Saunders MA, Wright MH. User's guide for NPSOL (Version 4.0): A Fortran Package for Non-linear Programming. Technical Report SOL 86-2, Department of Operations Research, Stanford University, 1986.
[31] Fleury C. Dual methods for convex separable problems. In: Rozvany GIN, editor. NATO/DFG ASI optimization of large structural systems. Berchtesgaden, Germany, Dordrecht, Netherlands: #132;lands; 1993. p. 509–30.
[32] NAG, Software manual. NAG Ltd, Oxford, UK, 1988.
[33] Thanedar PB, Vanderplaats GN. Survey of discrete variable optimization for structural design. J Struct Eng ASCE 1995;121(2):301–6.
[34] Papadrakakis M, Tsompanakis Y, Lagaros ND. Structural shape optimization using evolution strategies. Eng Optim 1999;31:515–40.
[35] Waagen D, Diercks P, McDonnell J. The stochastic direction set algorithm: a hybrid technique for finding function extrema. In: Fogel DB, Atmar W, editors. Proceedings of the 1st Annual Conference on Evolutionary Programming. Evolutionary Programming Society; 1992. p. 35–42.
[36] Hooke R, Jeeves TA. Direct search solution of numerical and statistical problems. J ACM 1961;8.
[37] Myung H, Kim J-H, Fogel D. Preliminary investigation into a two-stage method of evolutionary optimization on constrained problems. In: McDonnell JR, Reynolds RG, Fogel DB, editors. Proceedings of the 4th Annual Conference on Evolutionary Programming. Cambridge, MA: MIT Press; 1995. p. 449–63.
[38] Maa C, Shanblatt M. A two-phase optimization neural network. IEEE Trans Neural Networks 1992;3(6):1003–9.
[39] Eurocode 3, Design of steel structures, Part1.1: General rules for buildings, CEN, ENV 1993-1-1/1992.
[40] Orbinson JG, McGuire W, Abel JF. Yield surface applications in non-linear steel frames analysis. Comput Meth Appl Mech Eng 1982;33:557–73.
[41] Papadrakakis M, Papadopoulos V. A computationally efficient method for the limit elasto plastic analysis of space frames. Computat Mech 1995;16(2):132–41.