Pergamon

Ocean Engng, Vol. 25, No. 9, pp. 793–811, 1998

Printed in Great Britain
0029–8018/98 $19.00 + 0.00

PII: S0029–8018(97)10011–7

A MIXED CONTINUOUS AND DISCRETE NONLINEAR CONSTRAINED ALGORITHM FOR OPTIMIZING SHIP HULL STRUCTURAL DESIGN

Oscar B. Augusto* and Alexandre Kawano

Universidade de Sao Paulo-Departamento de Engenharia Naval e Oceanica, Av. Prof Mello Moraes, 05508-900-Sao Paulo, Sao Paulo, Brazil

(*Received* 9 *July* 1997; *accepted in final form* 23 *July* 1997)

Abstract—A nonlinear search algorithm for optimizing constrained design of ship structures is presented. The decision variables can be continuous or discrete and the constraints can be homogeneous or inequality nonlinear functions of those variables. The algorithm does not use gradients; therefore, it can work with non-systematized functions such as tables or another class of design routine. It was tested in the structural design of a Patrol Boat and has proved to be a powerful tool decreasing the time expended in preliminary design when it is done by the conventional spiral approach. © 1998 Elsevier Science Ltd. All rights reserved

1. INTRODUCTION

The ability of engineers to produce optimal designs has been severely limited by the techniques available for design optimization. Typically, much of the development effort has focused on simulation programs to evaluate design parameters, that is, analysis. One problem currently being addressed is how to use the information provided by the simulations in the iterative process of searching the parameter space for better designs. Given an evaluation of a setting of the parameters governing a design, on what basis should the choice be made of the "best" parameters to evaluate next? Because design spaces are combinatorially explosive and the time in which to develop a new design is limited, relatively few design points can be evaluated. Furthermore, since simulation programs rather than explicit analytic functional forms are typically used to model the design, the engineer's understanding of the parameter search space is more and more limited. This lack of knowledge is the reason that we need automated procedures for iterative design.

Most automated design processes are based on optimization. The optimization is purely a mathematical process and therefore it can and should be a fully automated process. That is, in the structural design process, the optimization step should be simply a "black box" which performs a specific task: it accepts as input an objective function and a set of constraints, and it returns as output the specific optimal values of the design variables, the values that maximize the objective while satisfying all of the constraints. Since there is only one optimum point in the design space, the optimization task is straightforward

*Author to whom correspondence should be addressed

and unambiguous. The main requirement is that the method must find the optimum rapidly and efficiently. The particular manner in which it does that is not important, and any optimization method that meets the requirement can be used in the design process.

2. THE PROBLEM FORMULATION

A mathematical programming problem can be formulated in a fairly standardized manner which has evolved in the literature. The use of a standard form is particularly desirable from an applications viewpoint since optimization programs can then be applied to many different design problems without change.

2.1. *Design variables*

The design or decision variables are those items which specify the design and need to be established. They form a vector of n, independent design variables to determine

$$\boldsymbol{X} = (x_1, x_2, x_3, \ldots x_n)^{\mathrm{T}} \tag{1}$$

These will be considered deterministic quantities, that is, not subject to probability distributions. It will treat design variables as parameters, not as functions of time. These variables may take on a continuous spectrum of values within a practical range or may be restricted to discrete values within the range as might be the case for a number of stringers in a stiffened panel.

2.2. *Objective function*

The objective function, cost function, or measure of merit is a scalar function (a single value) of the design variables, $f(\mathbf{X})$, which must be optimized. This can be an explicit function or a more complex design model involving equations, tables or other systematic procedures. In a structural design problem it might be the weight or fabricated cost of the structure.

As noted the objective function must have a single value for a given set of design variables; only one merit can be optimized at any one time. With more than a single item of interest, a weighted combination of the items must be optimized. The weighting of these items is then at the designer's discretion.

2.3. *Inequality constraints*

Practical design problems are usually subject to a series of inequality constraints which can be expressed as

$$g(\boldsymbol{X}) \geq 0 \tag{2}$$

These constraints may be linear or non-linear in x_i. Stress values which depend on the design variables may be limited to some maximum. This form includes just $g(\mathbf{X}) > 0$ since we can always transform $g(\mathbf{X}) \leq 0$ in $-\, g(\mathbf{X}) \geq 0$.

Scaling and normalization of constraints is important; particularly in the method that will be discussed later which assign penalties depending how much any constraint is violated or how closely it is approached during the solution process. When the penalties associated with the various constraints are combined it is necessary to have 1 meter violation in 10 meters maximum length constraint weighted about the same as a 1 mm violation in 10 mm maximum plate thickness constraint. Therefore, some form of normalization is necessary

and it can be accomplished by the following formulation technique: if a particular function of the design variables is constrained to be within maximum and minimum values such as $g_{\min} \leq g(\boldsymbol{X}) \leq g_{\max}$, then a formulation approach using

$$1 - \frac{g(X)}{g_{\max}} \geq 0$$

$$\frac{g(X)}{g_{\min}} - 1 \geq 0 \tag{3}$$

will produce all constraints in the desired greater than zero form and will ensure that the satisfaction of all constraints are considered equally.

2.4. *Equality constraints*

Practical design problems may also be subject to a series of homogeneous or equality constraints

$$h(\boldsymbol{X}) = 0 \tag{4}$$

Again these may be linear or non linear in the x_i. Equality constraint may be used to eliminate one of the design variables since with the equality constraint, the x_i are not all independent. An equality constraint can also be expressed as two inequality constraints if desired, enforcing $|h(\boldsymbol{X})| \leq \epsilon$, ϵ being a small number.

The optimization problem can now be expressed as selecting the optimum design variable $\mathbf{X}_o$ vector which will minimize $f(\mathbf{X})$ subject to the constraints. This will yield the optimum objective function value $f_0(\mathbf{X}_o)$. The minimization of the objective function is used since $\max(f)$ can be always treated as $\min(-f)$. Methods to solve this standard problem will be concentrated on when $f(\mathbf{X})$, $g(\mathbf{X})$ and $h(\mathbf{X})$ are nonlinear functions.

2.5. *Local vs. Global solutions*

A minimum of a function $f(\mathbf{X})$ may be the global minimum, that is, the lowest value of $f(\mathbf{X})$ for any $\mathbf{X}$ which is allowed by the constraints or is feasible. A minimum may also be a local minimum; that is, the lowest value of $f(\mathbf{X})$ in some local region of the feasible $\mathbf{X}$. In some complex design problems, any feasible point would be a good design solution, even if this point is a local minimum.

3. THE SEARCH ALGORITHM

The method developed for this work (see also Augusto, 1996) is based on a non-linear constrained search originally proposed by Box (1965), which uses the same idea developed by Nelder and Mead (1965).

Originally, Nelder-Mead algorithm utilizes a simplex, an $n + 1$ cornered shape in n dimensions, to perform an unconstrained direct search of an objective function. The method involves constructing a simplex in the search region and then allowing this shape to move toward the minimum point by sequentially replacing the corner of the current simplex having the highest value of $f(\mathbf{X})$, with a new lower value.

The process involves four vector operations called reflection, expansion, contraction and reduction, which allow the simplex to change shape and size as it moves towards the minimum. The expansion allows the search to accelerate along successful directions. The reduction is used if the simplex is too large and surrounds a maximum in $f(\mathbf{X})$.

Since the Nelder-Mead algorithm was developed to minimize $f(\mathbf{X})$ without constraints, Box extended the same idea developing another algorithm, designated as *Complex Algorithm*, to solve problems with interior constraints, explicit in the decision variables, $x_{\text{imin}} \leq x_i \leq x_{\text{imax}}$, and implicit by functions constraints $g(\mathbf{X}) \geq 0$.

A group of points containing the search variables, all satisfying interior constraints are randomly chosen. The point whose objective function is the worst is substituted by another chosen by reflection from it through the centroid of the remaining points, as long as the objective function of the new point is better than at least one of the remaining points. These points are called vertices and they form the *complex*.

As with many other constrained algorithms, the one proposed by Box needs a trial set of variables or a vertex which is feasible, satisfying all the explicit and implicit constraints. A second vertex point is randomly chosen and if it is unfeasible then it is contracted halfway towards the first point. This procedure is repeated until a feasible point is found*. The remaining points of the complex are chosen in the same approach differing only by the reflection point which is the centrefold of anterior generated vertices. Since all vertices are interior of a feasible domain, the search is not allowed to extrapolate the boundaries of that domain.

Box's algorithm is interesting when the initial point is easy to find and no homogeneous constraints are present at the problem. Usually, when designing complicated structures such as those found in the ocean industry, searching for a feasible point to start the process may not be a trivial task.

When a feasible starting point is not promptly available, engineering design optimization has been widely treated with penalty functions. For the penalty search it is defined an effective objective function, $\phi(\boldsymbol{X},r_k)$, that considers the original objective function, $f(\mathbf{X})$, and includes a penalty term, $p(\boldsymbol{X},r_k)$, generally a term proportional to the sum of constraint violations scaled by a penalty parameter r_k

$$\phi(\boldsymbol{X},r_k) = f(\boldsymbol{X}) + p(\boldsymbol{X},r_k) \tag{5}$$

If no feasible region is encountered around the optimum found by the algorithm, the penalty factor is modified and a new search is performed. Again if no feasible region is encountered the preceding procedure is repeated. This method is referred by Fletcher (1973), as Sequential Unconstrained Minimization Technique (SUMT). The penalty factor is arbitrarily modified and if in two trials, k and $k + 1$, of optimum search, the algorithm finds the same optimum point, then it is said that the search has converged. If the point is feasible the problem is solved hopefully for a global optimum, otherwise with the constraints imposed, it has no solution. Sometimes, if the multiplier is set properly for a specific objective function, the need for corrective action can be eliminated.

In spite of wide usage, penalty function methods have some potentially serious difficulties associated with them. One can observe: the problems of scaling the objective function and constraints, selecting an appropriate initial penalty factor, and determining the amount to reduce the penalty factor between cycles, are problems for which general procedures have not yet been developed. Improper scaling between multiple constraints or between

*Sometimes if the search space is not too narrow an unfeasible start point is possible when the second point is sorted inside the feasible space. But it still remains a very particular case.

the objective function and the constraints can result in the penalty function method converging to a non-optimal solution.

Although the aforementioned problems associated with penalty functions are present, an algorithm based on this assumption was chosen to develop a powerful tool for design tasks. A variety of modifications have led us to an algorithm denoted in this paper as the Modified Box Algorithm (MBA). The modifications include incorporating penalty function search, elimination of sequential unconstrained minimum search by providing a dynamic modification of penalty factor, discrete, continuous variable handling capacity, and regeneration.

3.1. *Regeneration*

Original Box algorithm has no regeneration. That means the algorithm can fail for nonconvex problems if the centroid falls into non-feasible regions. Nelder-Mead uses a regeneration method which contracts the entire simplex towards the best vertex.

In MBA it was decided to set an alternative direction for search which will provide success if an unfeasible centroid is found. A search direction from the worst vertex to the best vertex is tried. The worst vertex is reflected through the best vertex. If the new point is better than one in the complex then the regeneration is successful, the worst vertex is replaced and the regeneration is terminated. If the regeneration fails, the reflected vertex is contracted halfway along the direction of the minimum vertex and this procedure is repeated until a new point is found. At least the new point will be replaced by another equal to the minimum point and the search will proceed.

3.2. *Dynamic scaling* r_k *factor*

It is possible to recognize that the optimum being approached is unfeasible before it is reached. In these situations the multiplier r_k can be modified dynamically to force the optimum back to a constraint boundary. If a trial point is calculated into the unfeasible decision space, its penalty term and transformed function value were saved. If at some later time, another move was made into the unfeasible region and its penalty term was greater than that of the previously saved value and its transformed function value was less than the previously saved value, the multiplier is modified instantaneously, modifying the penalty term to a more adequate level, and the search proceeds from that point. Many function evaluations can be eliminated using this approach because a complete restart is not required and the search is not allowed to move to an unfeasible optimum.

3.3. *Discrete variables*

Usually, discreteness in design optimization has been most often treated either explicitly by branch and bound or dynamic programming or implicitly by rounding to a nearby integer solution.

The simplest and most prevalent technique in design tasks for treating integer variables is some form of rounding of the continuous optimum. This is often combined with a search of neighborhood of the rounded solution. Of course, obtaining a continuous optimum is possible only if the discrete variables of the problem can satisfactorily be treated as continuous. For example, if the variable denotes the use of a certain structural arrangement with specific properties that affect the objective function and constraints (for example a number of stiffeners in a reinforced panel) then it may be impossible to set the variable

at a non-integer value. Also, the fact that rounded solution, even with neighborhood search may not be optimal is widely recognized. A major difficulty in constrained problems is that the rounded solution may not only be sub-optimal but may actually be unfeasible. Locating a feasible discrete point may itself be a nontrivial task.

Treating the discreteness of the variables explicitly in a search algorithm has the advantage of only requiring evaluation of the objective function and constraints on the allowable set of discrete points. In this work to each discrete variable is associated a precision, a step or an increment, that means, an integer variable can be dealt with a precision 1, a multiple of 0.2 variable with a precision 0.2, and so on. Other non recursive variables such as plate thickness in a group, can be mapped into an integer variable with precision 1.

During the search after each vector operationm, the variables are kept within their fixed precision. In this fashion the algorithm can mix integer and continuous variables explicitly during the search, without additional difficulties.

3.4. *Penalty function*

There are two forms of penalty term definition. One of the most used formulations of the transformed function is as follows:

$$p(\mathbf{X},r_k) = r_k \sum_{m=1}^{M} \frac{1}{g_m(\mathbf{X})} \tag{6}$$

With this function the surface response is minimized for a sequence of decreasing values of parameter r_k such that r_k decreases when k increases. It is readily seen that if any one of the constraint functions $g_m(\mathbf{X})$ approaches zero, the penalty term in equation increases very rapidly. If the original problem is convex, this is also true for $\phi(\mathbf{X},r_k)$ and the minimum of $\phi(\mathbf{X},r_k)$ must be inside the feasible region and any unconstrained minimization algorithm can be applied in the search for the minimum point of $\phi(\mathbf{X},r_k)$. One should observe, however, that to find this optimum it is necessary to begin the search from an interior point (a feasible point). The penalty function given by Equation (6) is thus called an *interior penalty function* and has the advantage that all intermediate designs during the search for the optimum solution are feasible. One may therefore stop the search at any time and end up with a feasible and, hopefully, usable design.

Another usage of penalty functions are called *exterior penalty functions*, usually of the types

$$p(\boldsymbol{X},r_k) = -\, r_k \sum_{m=1}^{M} < g_m(\boldsymbol{X}) >$$

$$< g_m(\boldsymbol{X}) > = \begin{cases} 0, \textit{ if } g_m(\boldsymbol{X}) \geq 0 \\ \alpha_m |g_m(\boldsymbol{X})|^{\beta_m}, \textit{ if } g_m(\boldsymbol{X}) < 0 \end{cases} \tag{7}$$

The coefficients α_m e β_m control the amount a function constraint violation is penalized. Suppose a structural design has a deflection constraint such that it should be not greater than $g_{\max}$. In practice there will be a level of tolerance (perhaps one would apply only a very small penalty) say p_1, to a displacement of $g_{\max} + \epsilon_1$, but a really significant penalty, say p_2, to a displacement of $g_{\max} + \epsilon_2$. In such conditions the pair (α_m,β_m) is calculated as

$$\beta_m = \frac{\ln \dfrac{P_2}{P_1}}{\ln \dfrac{g_{\max} + \epsilon_2}{g_{\max} + \epsilon_1}}, \alpha_m = \frac{P_2}{(g_{\max} + \epsilon_2)^{\beta_m}} \tag{8}$$

The usual values found in literature are $(\alpha_m,\beta_m) = (1,1)$, linear penalty, and $(\alpha_m,\beta_m) = (1,2)$, quadratic penalty.

Exterior penalty functions are also of special interest in connection with equality constraints, since there is no interior region with respect to the equality constraints, which are therefore usually included in the penalty function by an exterior type formulation.

4. WORKED EXAMPLES

Further insight into the method developed may be obtained by examination of the following examples.

4.1. *Grillage*

Figures 1 and 2 illustrates the appearance of several $\phi(\mathbf{X},r_k)$ functions for the grillage design problem shown in Fig. 1. The problem, first analyzed by Kavlie and Moe (1971), consists of selecting a minimal cross-sectional area for each beam leading the grillage withstanding a defined span load, w, and keeping the bending stress below a 138 MPa level. The beam second moment of area, I, and elastic section modulus, Z, were made continuous functions of beam cross-sectional area, A.

There follow the grillage design main features:

Second moment of area:

$$I(\text{mm}^4) = 4.191 \times 10^5 \left[\frac{A(\text{mm}^2)}{954.8368}\right]^{2.65}$$

Shear force:

$$Q(x) = Q_{\text{END}} - wx$$

Point of Maximum bending moment:

$$x_{\text{m}} = \frac{Q_{\text{END}}}{w}; [\leq x_{\text{m}} \leq \frac{\ell}{2}]$$

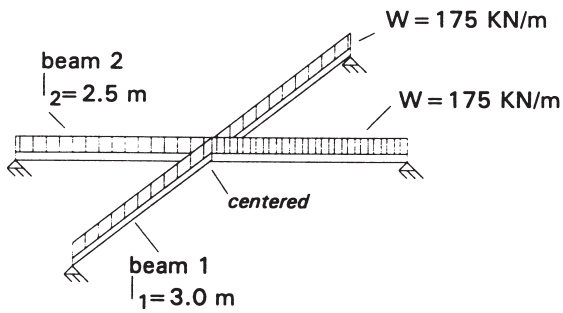


Fig. 1. Simple grillage design

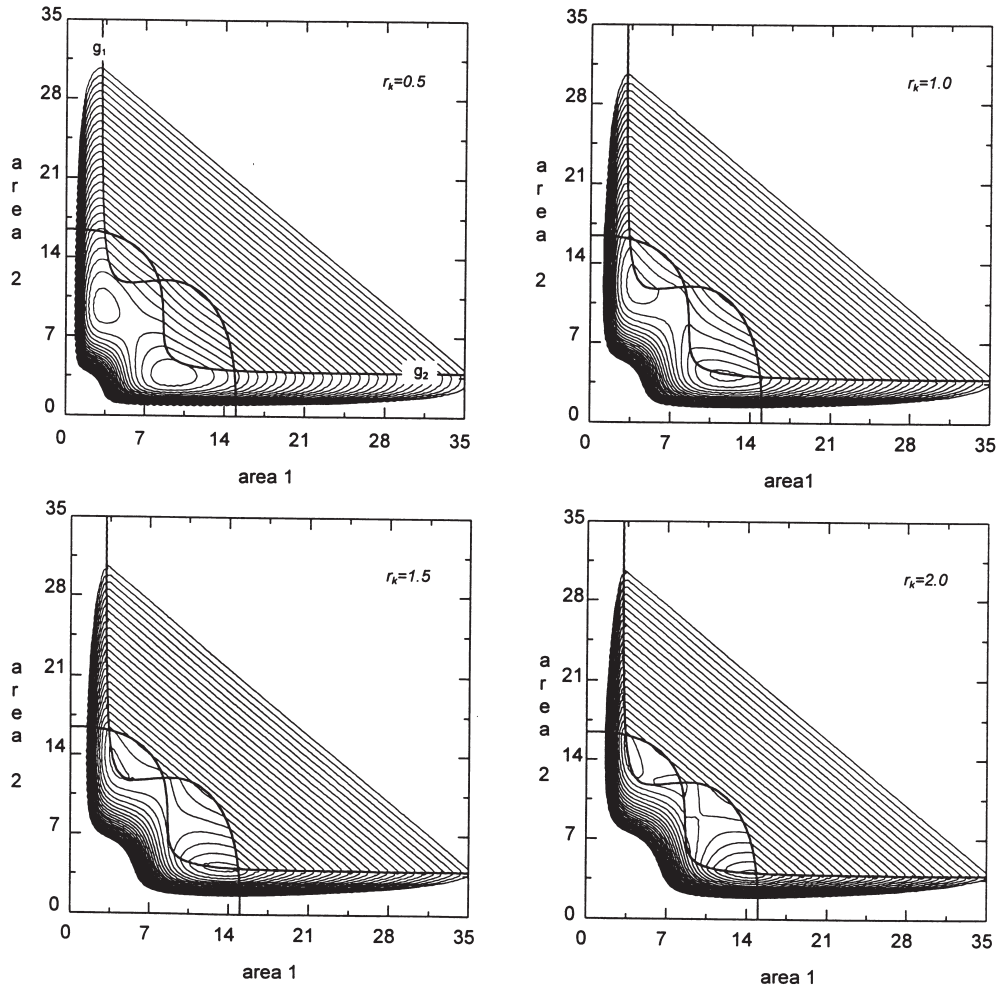


Fig. 2. External penalty surface response. Area (10^3mm^2)

Elastic section modulus:

$$Z(\text{mm}^3) = 1.6387 \times 104\left[\frac{A(\text{mm}^2)}{954.8368}\right]^{1.82}$$

Shear force at ends

$$\text{beam 1:} Q_{\text{END}} = \frac{w\ell - P}{2}$$

$$\text{beam 2:} Q_{\text{END}} = \frac{w\ell + P}{2}$$

Admissible stress:

$\sigma_{\text{adm}} = 138$ MPa

Displacements for a simply supported beam under:

uniform distributed load, w:

$$\delta = \frac{5w\ell^4}{384EI}$$

concentrated load, P:

$$\delta = \frac{P\ell^3}{48EI}$$

Constraints:

$$g_j(A_1,A_2) = 1 - \frac{M_{(j)\,\max}}{Z_j\sigma_{\text{adm}}}, \{\, j = 1,2\}$$

Bending moment:

$$M(x) = Q_{\text{END}}x - w\frac{x^2}{2}$$

Merit:

$$f(A_1,A_2) = 10^{-7}(\ell_1A_1 + \ell_2A_2)$$

In this non-convex problem there are three distinct relative minima. If the search method is not robust it can be trapped in a local minimum subspace, as shown in Table 1.

As can be seen in Fig. 2, the surface response is highly dependent on penalty parameter r_k and if an inappropriate r_k is chosen, the surface presents an unfeasible global minimum, requiring a larger penalty parameter to reshape the surface and move the minimum point into the feasible space.

As a performance idea of the proposed algorithm the features introduced in Box algorithm were incorporated in two other search algorithm: Nelder-Mead Simplex algorithm, and Hooke and Jeeves (1961) algorithm, as versions second Gabriele and Ragsdell (1984).

The MBA algorithm generates all points it needs to start the process. The other two need a starting vector, supplied by the user, and for the original Box algorithm the start vector should be feasible otherwise the search can fail. Where possible, all algorithms

Table 1. Grillage design results

Algorithm	Initial point A_{01}	A_{02}	Final results A_1	A_2	Objective function	Status	Number of simulations
MBA	15000	15000	14885.8	4004.8	4.923	s	148
	25000	25000	14885.1	4004.7	4.923	s	127
	35000	35000	14885.2	4004.7	4.923	s	158
	r	r	14885.1	4004.7	4.923	s	162
Box	15000	15000	9093.9	11111.5	5.607	f	182
	25000	25000	9093.9	11111.2	5.607	f	129
	35000	35000	9094.0	11111.2	5.607	f	128
Hooke-Jeeves	15000	15000	6127.9	14966.4	6.026	f	164
	25000	25000	14921.9	4218.7	4.949	s	193
	35000	35000	6127.9	14996.4	6.026	f	168
Nelder-Mead	15000	15000	14886.1	4004.7	4.923	s	274
	25000	25000	14885.2	4004.7	4.923	s	260
	35000	35000	14885.1	4004.7	4.923	s	264

Tolerance: 10^{-5}; 10000 units step in both variables for initial simpler in Nelder & Mead algorithm, and 10000 units initial step for the Hooke & Jeeves algorithm; search region (explicit constraints) $A_1 \subset [0,35000]$ and $A_2 \subset [0,35000]$; f: fail, s: success; r: pseudo-random starting points; linear penalty.

with the same initial conditions were submitted, as tolerance convergence, number of vectors in search, for Box and MBA and Nelder-Mead algorithms (2*n*,*n* number of variables), and the same initial step size for Nelder-Mead and Hooke-Jeves algorithm.

The grillage problem solution using these algorithms is shown in Table 1. The last column, number of simulations (number of designs executed), can be taken as a performance measurement of each algorithm.

As can be seen from Table 1, MBA shows robustness and good performance searching the optimum design.

4.2. *Cover hatch*

Another problem with two variables, proposed by Moe (1973) and found in literature, consists in designing an ideal box beam to cover a ship hull hatch opening. The box span length and width are shown in Fig. 3. The free design variables are flange thickness (t_b) and beam height (h). The cover hatch, subjected to a maximum load of 10 KPa, has to be designed in aluminum with modulus of elasticity E = 70,000 MPa, Poisson's ratio ν = 0.3 and admissible stress σ_{adm} = 70 MPa.

The following constraints must be met

Maximum displacement, $g_1(h,t_b)$:

$$\delta = \frac{5(pb)\ell^4}{384EI} \leq \delta_{\text{lim}} = \frac{\ell}{500} = 12.0 \text{ mm}$$

Useful properties:

Area: $A = 2(ht_h + bt_b)$

Inertia:

$$I = (t_h h + 3t_b b)\frac{h^2}{6}$$

Strength at maximum solicitation spots flange, external fiber, $g_2(h,t_b)$:

$$\sigma = \frac{(pb)\ell^2}{12\ \text{W}} \leq \sigma_{\text{adm}} = 70 \text{ MPa}$$

shear stress, web, neutral axis, $g_3(h,t_b)$:

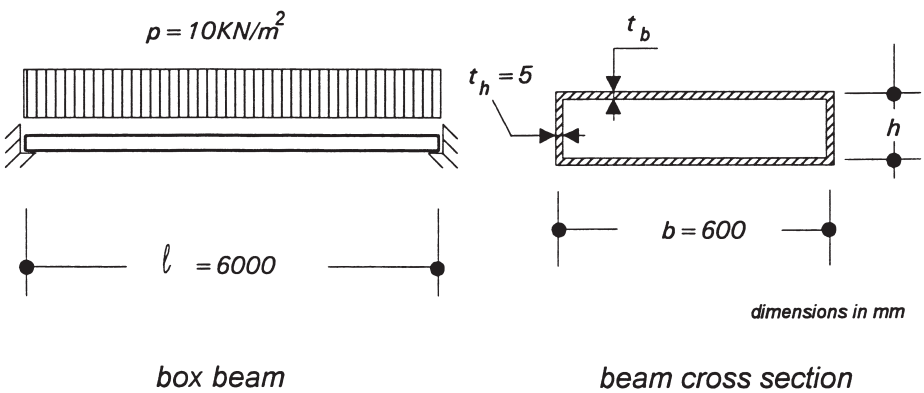


Fig. 3. Cover hatch box beam

$$\tau = \frac{\left(\frac{pb\ell}{2}\right)\mathrm{m}}{t_h I} \leq \tau_{\mathrm{adm}} = \frac{70}{\sqrt{3}}\ \mathrm{MPa}$$

Elastic modulus:

$$w = 2\,\frac{I}{h}$$

Moment of area at neutral axis:

$$m = \frac{1}{4}\,bt_b h + \frac{1}{8}\,h^2 t_h$$

flanges local stability, $g_4(h,t_b)$:

$$\frac{b}{t_b} \leq \frac{4\pi}{\sqrt{12(1-\mu^2)}}\,\sqrt{\frac{E}{\sigma}}$$

Merit:

$$f(h,t_b) = A$$

The response surface, with linear penalty for the cover hatch, shown in Fig. 4, were plotted using initial penalty parameter $r_1 = 1000$ and functions $g_j(\mathbf{X})$ normalized. In such conditions the surface response presents an unfeasible minimum point, forcing therefore, adjustments in the r_k factor during the search. The adjustments were made in power of ten and for the hatch problem three adjustments were sufficient to put the minimum point at a constraint boundary.

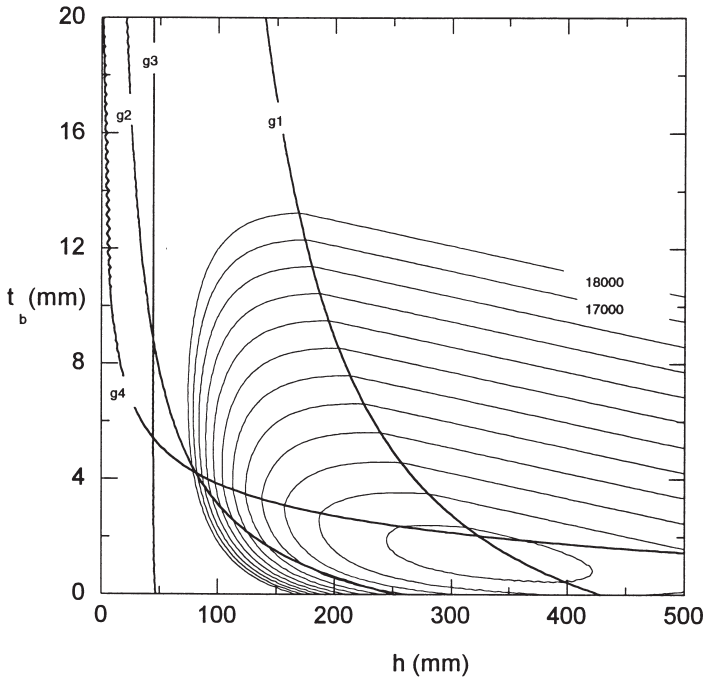


Fig. 4. Response Surface for the cover hatch

Table 2. Results for hatch cover design

Algorithm	Initial point h_0	t_{b0}	Final results h	t_b	Objective function	Status	Number of simulations
	300	2	323.816	2.932	6757	s	120
MBA	300	5	323.836	2.932	6757	s	113
	200	5	323.848	2.932	6757	s	127
	r	r	323.820	2.932	6757	s	109
	300	2	f	f	f	f	f
Box	300	5	323.933	2.932	6758	s	124
	200	5	f	f	f	f	f
	300	2	321.292	3.000	6813	s	172
Hooke-Jeeves	300	5	290.968	3.938	7635	f	155
	200	5	289.217	4.000	7692	f	153
	300	2	323.813	2.932	6757	s	283
Nelder-Mead	300	5	323.813	2.932	6757	s	301
	200	5	323.813	2.932	6757	s	315

Tolerance: 10^{-5}; 20 units step for*h* variable and 1 unit step for*t* variable for initial simplex in Nelder & Mead algorithm, and for initial steps in Hooke & Jeeves algorithm; search region (explicit constraints) h ⊂ [0,500] and t ⊂ [0,20]; initial point used in MBA to force the same initial conditions when compared with Box; f: fail, s: success; r:pseudo-ramdom starting points; linear penalty.

Table 2 shows the results for the cover hatch problem.

As can be noted in Table 2, MBA shows a good performance.

Finally, Table 3 shows the results of some trials with the hatch problem run with discrete variables. For each variable were fixed a precision or increment which keeps the variables in discrete values during search process, as mentioned in item 4. The optimum for continuous variables (see Table 2) is obtained with the pair $h = 323.8$ mm and $t_b = 2.932$ mm.

Some discrete cases were arbitrarily chosen and in the last row of Table 3 the beam height was permitted to assume non-recursive values {120,190,280,350,450,500} which were conveniently mapped into integers {1..6}. As can be seen, the results are as expected and the number of function evaluations decrease reasonably indicating an optional approach to reduce searching time.

4.3. *Pressure vessel*

A third example is a design optimization problem of a pressure vessel, shown in Fig. 5. This problem has been solved with different penalty methods (Fu *et al*., 1991; Li and Chou, 1994) and by two level genetic algorithms (Thierauf and Jianbo, 1997). The objective function is the combined costs of material, forming and welding of the pressure vessel.

Table 3. Hatch cover-MBA handling discrete variables

Increments δh	δt_b	Final results h	t_b	Objective function	Status	Number of simulations
30	5	270	5	8700	s	23
20	2	400	4	8800	s	17
20	1	340	3	7000	s	28
5	1	325	3	6850	s	35
{...}	1	350	3	7100	s	17

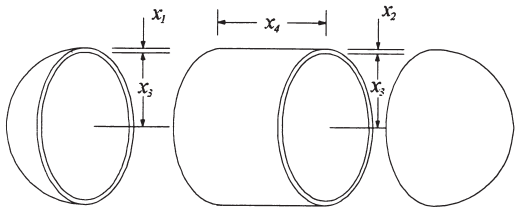


Fig. 5. A pressure vessel

The constraints are set in accordance with the respective ASME codes. The design optimization problem is formulated as follows:

Minimize
$f(\boldsymbol{X}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$
Subjected to

$$g_1(\boldsymbol{X}) = x_1 - 0.0193x_3 \geq 0$$

$$g_2(\boldsymbol{X}) = x_2 - 0.00954x_3 \geq 0$$

$$g_3(\boldsymbol{X}) = \pi x_3^2 x_4 + \frac{4}{3}\pi x_3^3 - 750.0{\cdot}1728.0 \geq 0$$

$$g_4(\boldsymbol{X}) = 240.0 - x_4 \geq 01.000 \leq x_1 \leq 1.3750.625 \leq x_2 \leq 1.000$$

The design variables x_3 and x_4 are continuous and the design variables x_1 and x_2 are discrete values with multiples of 0.0625 in.

Unfortunately the references do not show the number of simulations accomplished until the final results, which could be a performance measure of each method. The results shown in Table 4 confirm the robustness and the good performance of the proposed MBA search algorithm, which reached the final results performing 210 simulations.

Table 4. Results for the pressure vessel example

	Fu, Fenton and Cleghorn	Li, H. and Chou	Thierauf and Cai	MBA proposed algorithm
$f_{\min}$	8048.6	7127.3	7006.9	7006.8
x_1	1.125	1.000	1.000	1.000
x_2	0.625	0.625	0.625	0.625
x_3	48.380	51.250	51.812	51.813
x_4	111.745	90.991	84.591	84.583
g_1	0.191	1.011	0.000	0.000
g_2	0.163	0.136	0.131	0.131
g_3	75.875	18759.754	15.000	3.622
g_4	128.255	149.009	155.409	155.417

5. HULL SYNTHESIS OF A PATROL BOAT

Since the MBA algorithm had shown good performance it was decided to use it in the structural design of a patrol boat (Figs 6 and 7). The problem consists of synthesizing the structural panels which will form the hull mid-section. A typical panel with its design variables is shown in Fig. 6.

Details of this design can be found in Augusto (1996), but some features deserve to be mentioned. Starting a design process an engineer will have: the molded section (Fig. 8), the available material (Table 5) for light stiffeners, and plate thickness (in this project plate thicknesses ranged from 4–15 mm in 1 mm steps). The global and local loads for primary loads (Table 6) and for local loads (Table 7), and design constraints. It is common sense at first stages of hull design to divide the structural behavior into three categories, primary, secondary and tertiary, calculating the stresses in each one and then composing the effects to get the final state. Moreover it is usual to define stress limits for the strength as for the stiffness of each category composition, performing a total of at least six isolated safety factors. Restrictions about geometric appearance and limitations related with fabrication facilities must be also considered.

With all these data available the mid-section is then divided into panels, as shown in Fig. 9, and to synthesize the panels, a simulation program was specially written to perform the analysis of a hull structure (Augusto, 1996). The synthesis evolution and the final results are shown in Figs 10 and 11, respectively.

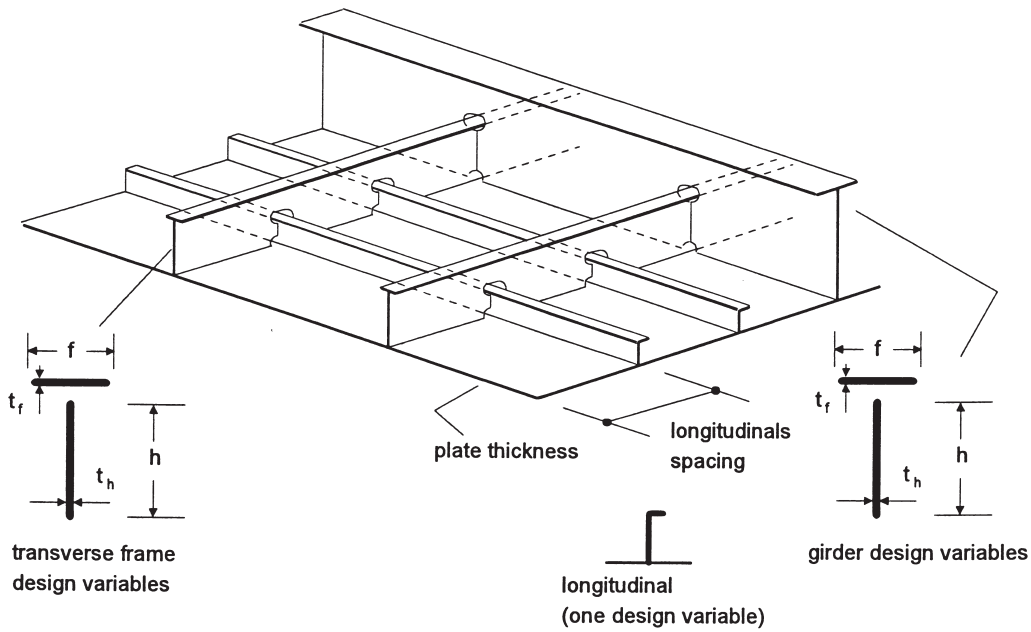


Fig. 6. Panel and girder design variables

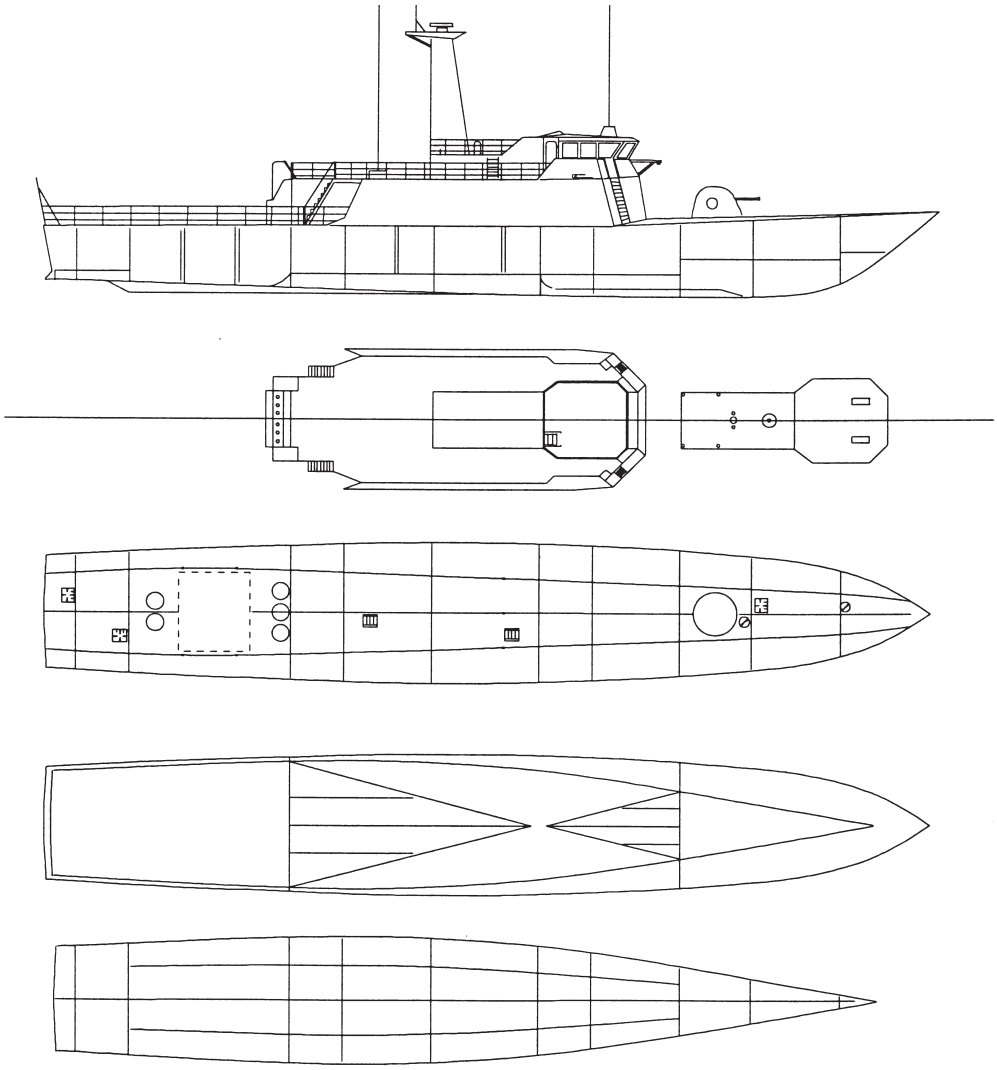


Fig. 7. The Patrol boat used for hull synthesis test

6. CONCLUSIONS

The algorithm presented has shown to be a powerful tool in helping design tasks where some kind of merit must be optimized. It was used as the heart of a computational system for designing ship structures, but it can be used in other complex engineering problems. Although the presented algorithm provides a design automation possibility freeing the engineer from many calculations tasks, it is still only a numerical tool and can never be a replacement for a sound engineering judgment.

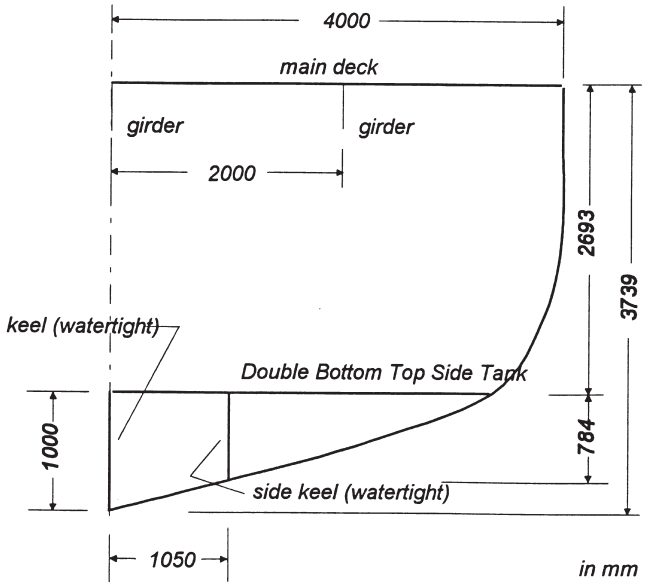


Fig. 8. Molded Hull Section

Table 5. Hull Girder Loads

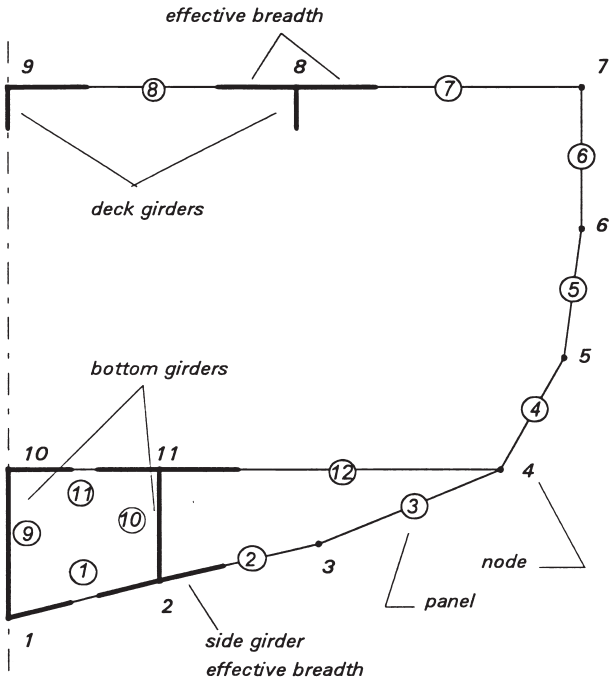
Condition	Bending moment (kN.m)	Shear force (kN)
Hogging	8222	562
Sagging	16737	1143

Table 6. Rolled T Sections Used for Longitudinals

Section number	Web h	th	Flange f	tf	Properties Htotal	Area	
1	70.000	4.450	25.400	6.350	76.350	473	
2	104.80	5.000	44.500	9.500	114.30	947	
3	113.70	6.350	63.500	13.360	127.06	1571	
4	138.00	6.860	76.200	14.220	152.22	2030	

Table 7. Panels Incidence and Local Loads

Panel Number	Incidence Node I	Node j	Lateral load (10^{-2}Pa) pi	pj
1	1	2	4.5	4.0
2	2	3	4.0	3.7
3	3	4	3.7	3.2
4	4	5	3.2	2.0
5	5	6	2.0	1.5
6	6	7	1.5	0.5
7	7	8	0.6	0.6
8	8	9	0.6	0.6
9	10	1	3.0	4.5
10	2	11	4.0	3.0
11	11	12	3.0	3.0
12	4	11	3.0	3.0



Node Coordinates (mm)

N	x	y
1	0.0	261
2	1050	523
3	2154	3428
4	3428	2091
5	3875	2091
6	4000	3000
7	4000	4000
8	2000	4000
9	0.0	4000
10	0.0	1307
11	1050	1307

Fig. 9. Panels discretization

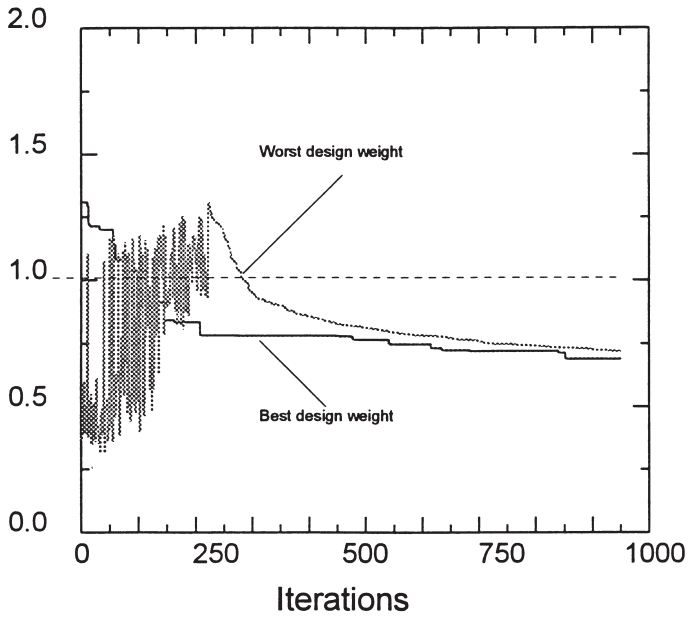


Fig. 10. Weights relative to a prototipe during synthesis evolution

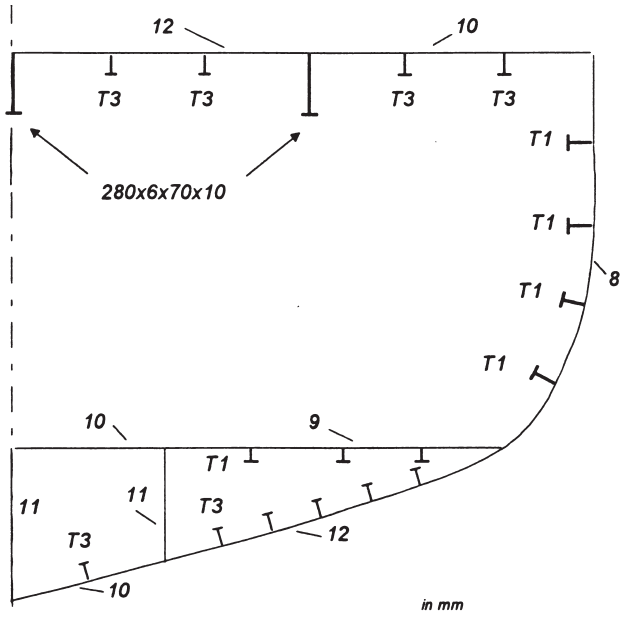


Fig. 11. Synthesis Results

Acknowledgements—The authors are grateful to FAPESP (Fundaçao de Amparo, a Pesquisa do Estado de Sao Paulo) for partial financial support of this research project.

REFERENCES

Augusto, O.B. (1996) *Structural Design of a Coastal Patrol Boat*. Technical Bulletin BT/PNV/26, Naval Architecture and Ocean Engineering Department, University of Sao Paulo (in Portuguese).

Box, M. J. (1965) A New Method of Constrained Optimization an Comparison with other Methods. *Computer Journal* **8**, 42–52.

Fletcher, R. (1973) Mathematical-Programming Methods-A Critical Review. In *Optimum Structural Design*, ed. R.H. Ghallagher and O.C. Zienkiewicz. John Wiley & Sons, Great Britain.

Fu, J., Fenton, R.G. and Cleghorn, W.L. (1991) A Mixed Integer-Discrete-Continuous Programming Method and its Application to Engineering Design Optimization. *Engineering Optimization*, 263–280

Gabriele, G.A. and Ragsdell, K.M. (1984) *OPTLIB, An Optimization Program Library*, University of Columbia.

Hooke, R. and Jeeves, T. A. (1961) Direct Search Solution of Numerical and Statistical Problems. *Journal of the Association of Computing Machinery* **8**, 212–229.

Kavlie, D. and Moe, J. (1971) Automated Design of Frame Structures. *Journal of The Structural Division ASCE* **97**, 33–62.

Li, H.L. and Chou, C.T. (1994) A Global Approach for Nonlinear Mixed Discrete Programming in Design Optimization. *Engineering Optimization*, 109–122

Moe, J. (1973) Penalty-function Methods. In *Optimum Structural Design-Theory and Applications*, ed. R.H. Gallagher and O.C. Zienkiewicz, pp. 143-177.

Nelder, J. A. and Mead, R. (1965) A Simplex Method for Function Minimization. *Computer Journal* **7**, 308–313.

Thierauf, G. and Jianbo, C. (1997) Parallel Evolution Strategy for Solving Structural Optimization. *Engineering Structures* **19**, 318–324.